DualPrompt: Complementary Prompting for Rehearsal-free Continual Learning

Zifeng Wang[1][⋆], Zizhao Zhang[2], Sayna Ebrahimi[2], Ruoxi Sun[2],
Han Zhang[3], Chen-Yu Lee[2], Xiaoqi Ren[2], Guolong Su[3],
Vincent Perot[3], Jennifer Dy[1], and Tomas Pfister[2]

[1]Northeastern University   [2]Google Cloud AI   [3]Google Research

Abstract. Continual learning aims to enable a single model to learn a sequence of tasks without catastrophic forgetting. Top-performing methods usually require a rehearsal buffer to store past pristine examples for experience replay, which, however, limits their practical value due to privacy and memory constraints. In this work, we present a simple yet effective framework, DualPrompt, which learns a tiny set of parameters, called *prompts*, to properly instruct a pre-trained model to learn tasks arriving sequentially without buffering past examples. DualPrompt presents a novel approach to attach complementary prompts to the pre-trained backbone, and then formulates the objective as learning task-invariant and task-specific "instructions". With extensive experimental validation, DualPrompt consistently sets state-of-the-art performance under the challenging class-incremental setting. In particular, DualPrompt outperforms recent advanced continual learning methods with relatively large buffer sizes. We also introduce a more challenging benchmark, Split ImageNet-R, to help generalize *rehearsal-free* continual learning research. Source code is available at https://github.com/google-research/l2p.

Keywords: Continual learning, rehearsal-free, prompt-based learning

1 Introduction

The central goal of continual learning (CL) is to learn a sequence of tasks with a single model without suffering from *catastrophic forgetting* [39] – a significant deterioration in performance on previously seen data. Many existing methods aim at preserving and extending the acquired knowledge during the continual learning process [12,34]. Architecture-based methods assign isolated parameters to encode learned knowledge from different tasks [26,31,36,53,58]. However, they often introduce a substantial number of additional parameters and sometimes involve simplified assumption like known test time task identity [11,36,35], which falls into the setting of task-incremental learning. However, the task-incremental setting is usually considered over-simplified [3,34,37], since task identity is not
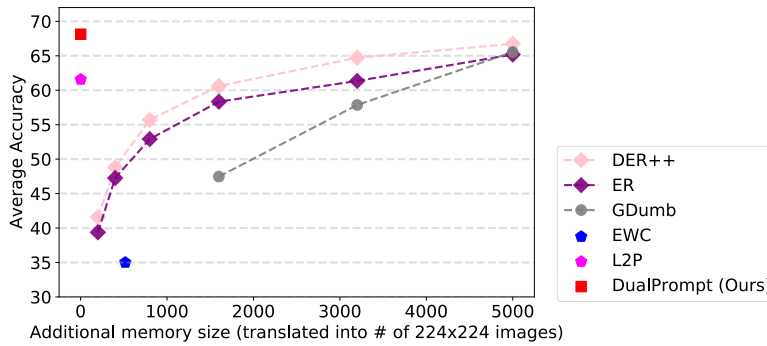

Fig. 1: The average accuracy comparison on Split ImageNet-R, suggesting that the accuracy degradation is significant for representative rehearsal-based methods like DER++ [3] and ER [7] when buffer size shrinks. Notably, they require a large rehearsal buffer (5000 images ≈ 20% of the whole training set) to close the gap to our method. In contrast, GDumb [46] matches their result by training only on the i.i.d sampled buffer, without continual learning. The additional parameters size required by DualPrompt is only about the bytes of one 224×224 RGB image. See Experiments section for discussion on compared methods.

known at test time in the real world. Our work focuses on more difficult class-incremental setting with unknown test-time task identity. Another line of work, rehearsal-based CL methods, preserve past knowledge directly by keeping data from prior tasks in a rehearsal buffer [3,4,44]. Due to their conceptual simplicity, generalizability to various settings, and superior ability to mitigate catastrophic forgetting, rehearsal-based methods have been widely recognized as the reigning state-of-the-art [5,3] in the challenging class-incremental setting. Nevertheless, the dependence on rehearsal buffer has been criticised in the community [53,12,46,30]. While the performance of these methods is sensitive to the size of the buffer, GDumb [46] argues that performing supervised training directly on a relatively large buffer already surpasses most recent CL methods. Critically, these methods cannot be used in applications with privacy concerns [54] or when memory budget is highly constrained [55]. Thus, it is desirable to develop a parsimonious, rehearsal-free continual learning method that can achieve similar or higher level of performance.

A recent method, Learning to Prompt (L2P) [59] approaches this problem from a brand-new perspective – it proposes to leverage learnable *prompt* parameters to encode knowledge in a much more succinct way (i.e. prompt pool) than buffer, thus a rehearsal buffer is no longer necessary. Prompt techniques are originally introduced in natural language processing (NLP) for task adaptation [28] of large-scale pre-trained models by attaching fixed or learnable "instructions", since prompts are designed to instruct the model to properly reuse learned representations, instead of learning new representations from scratch. L2P successfully formulates the problem of learning new tasks as training small prompt

parameters attached to a pre-trained frozen model. L2P takes an exciting step towards rehearsal-free continual learning, although the performance is still lower than rehearsal-based methods.

In L2P, one single prompt pool is designed to transfer knowledge from one task to another without distinguishing between the common features among all tasks versus the features that are unique to each task. We argue such a design could be sub-optimal from the perspective of theory of Complementary Learning Systems (CLS) [38,22], an intuition that many recent advanced CL methods are based on [7,44,3]. CLS suggests that humans learn continually via the synergy between two learning systems: the hippocampus focuses on learning pattern-separated representation on specific experiences, and the neocortex focuses on learning more general and transferable representation from past experience sequences. Thus, they are able to learn task-specific knowledge separately without interference while leveraging task-invariant knowledge to have greater learning capacity to learn future tasks better. However, previous CLS-driven methods still decouple or expand the backbone parameters learn the two kinds of knowledge [11,44,45]. Thus, they still rely on constructing a rehearsal buffer repeatedly to consolidate decoupled knowledge to prevent catastrophic forgetting.

In this paper, we present DualPrompt, a rehearsal-free continual learning approach to explicitly learn two sets of disjoint prompt spaces, *G(eneral)-Prompt* and *E(xpert)-Prompt*, that encode task-invariant and task-specific instructions, respectively. DualPrompt directly decouples the higher-level prompt space, which turns out to be more effective and memory efficient that conventional methods which focus on the lower-level latent representation space. We further explore where and how to attach both types of prompts is crucial to steer the backbone model to learn with less forgetting and to achieve effective knowledge sharing, thus significantly enhancing the effectiveness of continual learning.

Moreover, we introduce Split ImageNet-R, a new CL benchmark based on ImageNet-R [15] to the community. The intra-class diversity for each task in Split ImageNet-R is large (see Appendix E for representative examples), thus a small buffer is not sufficient to represent past experiences. Figure 1 showcases that the size of rehearsal buffer needed is non-trivial for even advanced methods to perform well. While rehearsal-based methods require a large buffer (up to 20% of total training data) to achieve a competitive average accuracy, our method DualPrompt shows superior performance despite not using any rehearsal buffer.

In summary, our work makes the following contributions:

- We propose DualPrompt, a simple and effective rehearsal-free CL method, comprised of G-Prompt and E-Prompt for learning task-invariant and task-specific knowledge, respectively. The method is fairly simple to apply without data or memory access concerns which is favorable for real-world CL scenarios.
- DualPrompt explores various design choices to incorporate these two types of prompts into the pre-trained models. For the first time, we empirically discover that properly attaching prompts to the backbone model is crucial to the effectiveness of continual learning.

– We introduce a new CL benchmark, Split ImageNet-R to help validate the method. DualPrompt sets new state-of-the-art performance on multiple benchmarks under the challenging class-incremental setting, and beats rehearsal-based methods with relatively large buffer size.

2   Related work

Continual learning. We discuss three related categories of continual learning methods: regularization-based, rehearsal-based and architecture-based methods.

Regularization-based methods [19,66,27,1] address catastrophic forgetting by regularizing important parameters for learned tasks. Although these methods mitigate forgetting under simpler task-incremental setting, their performance under more challenging class-incremental setting [34], or more challenging datasets [61] is not satisfactory.

Architecture-based methods assign isolated parameters for each task. These methods can be further categorized as expanding the model [52,64,26,31,68], or dividing the model [36,53,58,17,11]. However, a major part of the work is limited to the task-incremental setting [53,36,35,17], while other work only considers specific convolutional-based architectures [60,45,11]. However, DualPrompt aims at more challenging class-incremental setting, and focus on pre-trained transformer-based models. Moreover, architecture-based method generally require substantially large amount of additional parameters to assist model separation [58,17,63]. On the contrary, DualPrompt is much lightweight and only require negligible amount of parameters ($0.2\% - 0.6\%$ of full model size).

Rehearsal-based methods save data from learned tasks in a rehearsal buffer to train with the current task. Although these methods share this quite simple idea, they are very effective even in the class-incremental setting. Several advanced rehearsal-base methods achieve state-of-the-art performance [3,4]. However, rehearsal-based methods deteriorates when buffer size [4] decreases, and are eventually not applicable to data privacy sensitive scenarios [54]. Some recent methods are inspired from the Complementary Learning Systems (CLS). However, ACL [11] is limited to the task-incremental setting, DualNet [44] requires specific architecture design, and both methods still rely on a rehearsal buffer to work well. Our DualPrompt tackles continual learning from a rehearsal-free perspective, standing upon a wise utilization of pre-trained models, thus getting rid of the shortcomings of rehearsal-based methods.

Prompt-based learning. As an emerging transfer learning technique in natural language processing (NLP), prompt-based learning (or prompting), applies a fixed function to condition the model, so that the language model gets additional instructions to perform the downstream task. However, the design of a prompting function is challenging and requires heuristics. To this end, recent work propose to apply prompts as learnable parameters, achieving outstanding performance on transfer learning [24,25]. Prompts capture task-specific knowledge with much smaller additional parameters, than its competitors, such as Adapter [57,43] and LoRA [16]. As discussed above, L2P [59] is the only work that connects

prompting and continual learning. Differently, DualPrompt takes inspiration from CLS and presents a different approach to attach complementary prompts to the pre-trained backbone to learn task-invariant and task-specific instructions. We show DualPrompt outperforms L2P consistently.

3 Prerequisites

3.1 Continual learning problem setting

Continual learning is defined as training machine learning models on a continuum of data from a sequence of tasks. We denote the sequence of tasks as $\mathcal{D} = \{\mathcal{D}_1, \cdots, \mathcal{D}_T\}$, where the t-th task $\mathcal{D}_t = \{(\boldsymbol{x}_{i,t}, y_{i,t})\}_{i=1}^{n_t}$ contains tuples of the input sample $\boldsymbol{x}_{i,t} \in \mathcal{X}$ and its corresponding label $y_{i,t} \in \mathcal{Y}$. The model $f_\theta : \mathcal{X} \to \mathcal{Y}$ is parameterized by θ, such that it predicts the label $y = f_\theta(\boldsymbol{x}) \in \mathcal{Y}$ given an unseen test sample $\boldsymbol{x}$ from arbitrary tasks. Data from the previous tasks is not available when training future tasks.

We use the widely-adopted assumption that the task boundaries are clear and the task switch is sudden at training time [6,44]. Moreover, we consider the more challenging class-incremental learning [6] setting, *i.e.*, task identity is unknown for each example at test time. Also, following the settings in prior work [59], we assume a pre-trained sequence model, *e.g.*, a vision transformer (ViT) [9] on ImageNet, is available, a wide-used assumption in recent literature of the computer vision community. Unlike many rehearsal-based methods [7,3], we do not assume any form of rehearsal buffer as a prerequisite.

3.2 Prompt-based learning

Prompt-based learning (or prompting) was first proposed in NLP for transfer learning. The main idea of prompting is to add extra instruction for pre-trained models to perform downstream tasks conditionally [28]. Prompt Tuning [24], one of the recent emerging techniques, proposes to attach a set of prompt parameters to frozen transformer-based language models [47] to perform downstream NLP tasks. The prompts are usually prepended to the input sequence to instruct the model prediction. We briefly illustrate the idea of Prompt Tuning below.

As we mainly focus on vision-related continual learning setting, here we introduce the definition of Prompt Tuning using the vision transformer (ViT) based sequence models [10,56]. In ViT, the input embedding layer transforms the input image into a sequence-like output feature $\boldsymbol{h} \in \mathbb{R}^{L \times D}$, where L is the sequence length and D is the embedding dimension. When solving downstream tasks, the pre-trained backbone is kept frozen as a general feature extractor, and the prompt parameters $\boldsymbol{p} \in \mathbb{R}^{L_{\boldsymbol{p}} \times D}$ with sequence length $L_{\boldsymbol{p}}$ and embedding dimension D are prepended to the embedding feature along the sequence length dimension to form the extended embedding feature. Finally, the extended feature is sent to the rest of the model for performing classification tasks. Prompt serves as a lightweight module to encode high-level instruction to instruct the backbone to leverage pre-trained representations for downstream tasks.
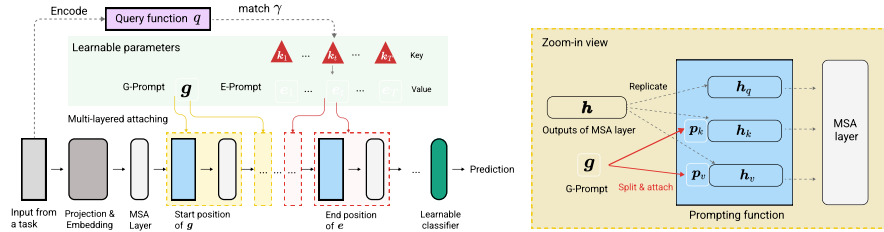
Fig. 2: Overview of DualPrompt. **Left**: At test time, an input is transformed by a query function to match the closest task key $\boldsymbol{k}_t$ and the corresponding E-Prompt $\boldsymbol{e}_t$. Then the shared G(eneral)-Prompt $\boldsymbol{g}$ and the matched E(xpert)-Prompt $\boldsymbol{e}_t$ are attached to multiple MSA layers of a pre-trained transformer. At training time, the E-Prompt is selected by task identity and the selected E-Prompt and G-Prompt are trained together with the classifier. **Right**: A prompting function is illustrated where the G-prompt is split equally and attached to the key and value replicas of the hidden feature (see Section 4.2) before passing them to the preceding MSA layer.

4   DualPrompt

Our proposed method, DualPrompt is illustrated in Figure 2. We first introduce the complementary learning components, G- and E-prompts, in Section 4.1 by showcasing how they work with a single multi-head self-attention (MSA) layer. We then explore design choices of attaching prompts to the backbone Section 4.2. We finally present the overall objective for DualPrompt in Section 4.3.

4.1   Complementary G-Prompt and E-Prompt

Given a pre-trained ViT f with N consecutive MSA layers, we further extend the notations introduced in 3.2 by denoting the input embedding feature of the i-th MSA layer as $\boldsymbol{h}^{(i)}, i = 1, 2, \cdots, N$.

G-Prompt: $\boldsymbol{g} \in \mathbb{R}^{L_g \times D}$ with sequence length L_g and embedding dimension D, is a shared parameter for all tasks. Suppose we would like to attach G-Prompt to the i-th MSA layer, G-Prompt transforms $\boldsymbol{h}^{(i)}$ via a *prompting function*:

$$\boldsymbol{h}_g^{(i)} = f_{\text{prompt}}\left(\boldsymbol{g}, \boldsymbol{h}^{(i)}\right), \tag{1}$$

where f_{prompt} defines the approach how to attach the prompts to the hidden embeddings. Section 4.2 discusses the details.

E-Prompt: $\mathbf{E} = \{\boldsymbol{e}_t\}_{t=1}^T$ is a set of task-dependent parameters, where $\boldsymbol{e}_t \in \mathbb{R}^{L_e \times D}$ has a sequence length of L_e and the same embedding dimension D as the G-Prompt, and T is the total number of tasks. Different from the shared G-Prompt, each $\boldsymbol{e}_t$ is associated with a task-specific key $\boldsymbol{k}_t \in \mathbb{R}^D$, which is also a learnable parameter that aims to capture representative features of a task. For an input example from the t-th task, to attach E-Prompt to the j-th MSA layer, we apply the prompting function in a similar way:

$$\boldsymbol{h}_e^{(j)} = f_{\text{prompt}}\left(\boldsymbol{e}_t, \boldsymbol{h}^{(j)}\right). \tag{2}$$

Moreover, we update the corresponding $\boldsymbol{k}_t$ to match the feature of the input instance via a matching loss $\mathcal{L}_{\text{match}}$, such that $\boldsymbol{k}_t$ becomes "closer" to examples from the t-th task than other keys. At test time, inspired by the strategy proposed in [59], we propose to adopt a query function q on the test sample to search for the best match from the task keys, and select the corresponding E-Prompt to use. Although it is interesting to design various matching and query strategies by introducing additional components, it actually violates the principle of parsimony in continual learning [12,58]. Fortunately, as suggested in [59], we can directly use the whole pre-trained model as the query function: $q(\boldsymbol{x}) = f(\boldsymbol{x})[0]$ (the feature vector corresponding to [class] token [10]), and cosine similarity as γ. Thus, the matching loss takes the following form:

$$\mathcal{L}_{\text{match}}(\boldsymbol{x}, \boldsymbol{k}_t) = \gamma(q(\boldsymbol{x}), \boldsymbol{k}_t), \quad \boldsymbol{x} \in \mathcal{D}_t. \tag{3}$$

For a test example $\boldsymbol{x}$, we simply choose the best matched task key index via $\operatorname{argmin}_t \gamma(q(\boldsymbol{x}), \boldsymbol{k}_t)$. We show the relationship between query accuracy and final performance in Appendix I. We empirically discover this matching loss and the corresponding query mechanism works fairly well for all benchmarks.

4.2   Prompt attaching: where and how?

G- and E-prompts encode respective type of instructions during training with the backbone and cooperatively instruct the model to make predictions at inference. We have showcased how to attach them to a single MSA layer in Section 4.1. Most existing prompt-related work simply place prompts only at the first MSA [59,24], or at every MSA layer [25,29]. However, we argue that it is crucial to explore *where* and *how* to attach both types of prompts.

Where: Decoupled prompt positions. Intuitively, different layers of the backbone have different levels of feature abstraction [48]. Therefore, when learning tasks sequentially, some layers of representations can have higher responses to task-specific knowledge than others, vise versa for task-invariant knowledge. This motivates us to give the two types of prompts more flexibility to attach to the most proper positions in a decoupled way, thus different instructions can interact with the corresponding representations more effectively.

With a slight abuse of notation, we introduce the multi-layered extension of both types of prompts: $\boldsymbol{g} = \{\boldsymbol{g}^{(l)}\}_{l=\texttt{start}_g}^{\texttt{end}_g}$, where $\boldsymbol{g}^{(l)} \in \mathbb{R}^{L_g \times D}$ is the G-Prompt to be attached to the l-th MSA layer. We also define $\boldsymbol{e}_t = \{\boldsymbol{e}_t^{(l)}\}_{l=\texttt{start}_e}^{\texttt{end}_e}$ similarly. In this way, we are able to attach the G-Prompt $\boldsymbol{g}^{(l)}$ from the $\texttt{start}_g$-th to the $\texttt{end}_g$-th MSA layers, and attach the E-Prompt $\boldsymbol{e}_t^{(l)}$ from the $\texttt{start}_e$-th to the $\texttt{end}_e$-th MSA layers. And most importantly, $(\texttt{start}_g, \texttt{end}_g)$ and $(\texttt{start}_e, \texttt{end}_e)$ could be totally different or non-overlapping. In our experiments, we empirically search for a certain set of $\texttt{start}_g, \texttt{end}_g, \texttt{start}_e, \texttt{end}_e$ on a validation set and discover that it performs consistently well across different benchmarks. Note that we make a simplified assumption that the chosen indices of MSA layers to attach prompts are contiguous, which already achieves state-of-the-art performance

in our empirical evaluation. However, there could be more advanced ways to auto-search the configuration, which we treat as valuable future work.

How: Configurable prompting function. The prompting function f_{prompt} controls the way we combine prompts with the embedding features. From another perspective, f_{prompt} directly affects how the high-level instructions in prompts interact with low-level representations. Thus, we believe a well-designed prompting function is also vital for the overall continual learning performance. Although DualPrompt is compatible with various prompting functions, here we exemplify and study two mainstream realizations in the NLP community - Prompt Tuning (Pro-T) [24] and Prefix Tuning (Pre-T) [25].

Specifically, applying a prompting function can be viewed as modifying the inputs of the MSA layers [56]. Let the input to the MSA layer be $\boldsymbol{h} \in \mathbb{R}^{L \times D}$, and we further denote the input query, key, and values for the MSA layer to be $\boldsymbol{h}_Q, \boldsymbol{h}_K, \boldsymbol{h}_V$, respectively. Recall that the MSA layer is proposed by [56]:

$$\text{MSA}(\boldsymbol{h}_Q, \boldsymbol{h}_K, \boldsymbol{h}_V) = \text{Concat}\left(\text{h}_1, \ldots, \text{h}_\text{m}\right) W^O$$
$$\text{where}\, \text{h}_\text{i} = \text{Attention}\left(\boldsymbol{h}_Q W_i^Q, \boldsymbol{h}_K W_i^K, \boldsymbol{h}_V W_i^V\right),$$

where W^O, W_i^Q, W_i^K, and W_i^V are projection matrices. m is the number of heads. In ViT, $\boldsymbol{h}_Q = \boldsymbol{h}_K = \boldsymbol{h}_V$. For simplicity, we define a unified prompt parameter $\boldsymbol{p} \in \mathbb{R}^{L_p \times D}$ ($\boldsymbol{p}$ could be either single-layered G or E-Prompt).

Prompt Tuning (Pro-T) prepends prompts to the input tokens, which is equivalent to concatenate the same prompt parameter $\boldsymbol{p}$ to $\boldsymbol{h}_Q$, $\boldsymbol{h}_K$, and $\boldsymbol{h}_V$,

$$f_{\text{prompt}}^{\text{Pro-T}}(\boldsymbol{p}, \boldsymbol{h}) = \text{MSA}([\boldsymbol{p}; \boldsymbol{h}_Q], [\boldsymbol{p}; \boldsymbol{h}_K], [\boldsymbol{p}; \boldsymbol{h}_V]), \tag{4}$$

where $[\cdot; \cdot]$ defines the concatenation operation along the sequence length dimension. The output length increases, resulting the output dimension as $\mathbb{R}^{(L+L_p) \times D}$. The operation is equivalent to how `[class]` is added [10] at the first MSA layer.

Prefix Tuning (Pre-T) splits $\boldsymbol{p}$ into $\boldsymbol{p}_K, \boldsymbol{p}_V \in \mathbb{R}^{L_p/2 \times D}$, and prepends them to to $\boldsymbol{h}_K$ and $\boldsymbol{h}_V$ respectively, while keep $\boldsymbol{h}_Q$ as-is:

$$f_{\text{prompt}}^{\text{Pre-T}}(\boldsymbol{p}, \boldsymbol{h}) = \text{MSA}(\boldsymbol{h}_Q, [\boldsymbol{p}_k; \boldsymbol{h}_K], [\boldsymbol{p}_v; \boldsymbol{h}_V]). \tag{5}$$

Compared with Pro-T, the output sequence length remains the same as input $\boldsymbol{h} \in \mathbb{R}^{L \times D}$. Section 5.4 studies both versions empirically and discusses the intuition behind their difference in performance from a continual learning perspective.

4.3   Overall objective for DualPrompt

The full picture of DualPrompt at training and test time is described in Algorithm 1 and 2, respectively, in Appendix A. Following the design patterns discussed in Section 4.2, we denote the architecture with prompts attached by $f_{\boldsymbol{g}, \boldsymbol{e}_t}$. Then we transform our input $\boldsymbol{x}$ from the t-th task via $f_{\boldsymbol{g}, \boldsymbol{e}_t}$ and send it to the classification head f_ϕ parametrized by ϕ for prediction. Finally, we train both

types of prompts, the task keys, as well as the newly-initialized classification head in an end-to-end fashion:

$$\min_{\boldsymbol{g}, \boldsymbol{e}_t, \boldsymbol{k}_t, \phi} \mathcal{L}(f_\phi(f_{\boldsymbol{g}, \boldsymbol{e}_t}(\boldsymbol{x})), y) + \lambda \mathcal{L}_{\text{match}}(\boldsymbol{x}, \boldsymbol{k}_t), \quad \boldsymbol{x} \in \mathcal{D}_t, \tag{6}$$

where $\mathcal{L}$ is the cross-entropy loss, $\mathcal{L}_{\text{match}}$ is the matching loss defined in equation 3, and λ is a scalar balancing factor.

5 Experiments

5.1 Evaluation benchmarks

Split ImageNet-R. The Split ImageNet-R benchmark is build upon ImageNet-R [15] by dividing the 200 classes randomly into 10 tasks with 20 classes per task. We split the dataset into training and test set with 24,000 and 6,000 images respectively. We further sample 20% from the training set as validation data for prompt attaching design search. The original ImageNet-R includes newly collected data of different styles, such as cartoon, graffiti and origami, as well as hard examples from ImageNet [8] that standard models, *e.g.*, ResNet [14], fail to classify. We believe the Split ImageNet-R is of great importance to the continual learning community, for the following reasons: 1) Split ImageNet-R contains classes with different styles, which is closer to the complicated real-world problems. 2) The significant intra-class diversity (see Appendix E) poses a great challenge for rehearsal-based methods to work effectively with a small buffer size (see Figure 1), thus encouraging the development of more practical, rehearsal-free methods. 3) Pre-trained vision models are useful in practice for many fields [51,20], including continual learning. However, their training set usually includes ImageNet. Thus, Split ImageNet-R serves as a relative fair and challenging benchmark, and an alternative to ImageNet-based benchmarks [50,61] for continual learning that uses pre-trained models.

Split CIFAR-100. Split CIFAR-100 is a widely-used benchmark in continual learning literature. It splits the original CIFAR-100 [21] into 10 disjoint tasks, with 10 classes per task. Although it is a relatively simple task for image classification under the i.i.d. setting, it sufficiently makes advanced CL methods expose large forgetting rate in class-incremental learning.

We use Split ImageNet-R and Split CIFAR-100 to demonstrate our main results in Section 5.2, and additionally conduct experiments on 5-datasets for completeness in the Appendix H.

5.2 Comparison with state-of-the-arts

We compare DualPrompt against representative baselines and state-of-the-art methods. Please refer to Appendix B for experimental details. We use the widely-used *Average accuracy* (higher is better) and *Forgetting* (lower is better) [32,6,34] as our evaluation metrics. The definitions of both metrics are in Appendix C.

Table 1: Results on class-incremental learning (i.e., task identity is unknown at test time). We compare and group methods by buffer sizes. 0 means no rehearsal is used, when most SOTA methods are not applicable anymore. Note that the chosen buffer sizes here are considered sufficiently large sizes that are used in prior works for Split CIFAR-100 [6,3]. They are large enough even for training a supervised counterpart – e.g. GDumb [46] trains on the i.i.d sampled buffer with this size and demonstrated competitive results, making continual training unnecessary.

Method	Buffer size	Split CIFAR-100		Buffer size	Split ImageNet-R	
		Avg. Acc ($\uparrow$)	Forgetting ($\downarrow$)		Avg. Acc ($\uparrow$)	Forgetting ($\downarrow$)
ER [7]	1000	$67.87_{\pm 0.57}$	$33.33_{\pm 1.28}$	1000	$55.13_{\pm 1.29}$	$35.38_{\pm 0.52}$
BiC [61]		$66.11_{\pm 1.76}$	$35.24_{\pm 1.64}$		$52.14_{\pm 1.08}$	$36.70_{\pm 1.05}$
GDumb [46]		$67.14_{\pm 0.37}$	-		$38.32_{\pm 0.55}$	-
DER++ [3]		$61.06_{\pm 0.87}$	$39.87_{\pm 0.99}$		$55.47_{\pm 1.31}$	$34.64_{\pm 1.50}$
Co^{2}L [4]		$72.15_{\pm 1.32}$	$28.55_{\pm 1.56}$		$53.45_{\pm 1.55}$	$37.30_{\pm 1.81}$
ER [7]	5000	$82.53_{\pm 0.17}$	$16.46_{\pm 0.25}$	5000	$65.18_{\pm 0.40}$	$23.31_{\pm 0.89}$
BiC [61]		$81.42_{\pm 0.85}$	$17.31_{\pm 1.02}$		$64.63_{\pm 1.27}$	$22.25_{\pm 1.73}$
GDumb [46]		$81.67_{\pm 0.02}$	-		$65.90_{\pm 0.28}$	-
DER++ [3]		$83.94_{\pm 0.34}$	$14.55_{\pm 0.73}$		$66.73_{\pm 0.87}$	$20.67_{\pm 1.24}$
Co^{2}L [4]		$82.49_{\pm 0.89}$	$17.48_{\pm 1.80}$		$65.90_{\pm 0.14}$	$23.36_{\pm 0.71}$
FT-seq	0	$33.61_{\pm 0.85}$	$86.87_{\pm 0.20}$	0	$28.87_{\pm 1.36}$	$63.80_{\pm 1.50}$
EWC [19]		$47.01_{\pm 0.29}$	$33.27_{\pm 1.17}$		$35.00_{\pm 0.43}$	$56.16_{\pm 0.88}$
LwF [27]		$60.69_{\pm 0.63}$	$27.77_{\pm 2.17}$		$38.54_{\pm 1.23}$	$52.37_{\pm 0.64}$
L2P [59]		$83.86_{\pm 0.28}$	$7.35_{\pm 0.38}$		$61.57_{\pm 0.66}$	$9.73_{\pm 0.47}$
DualPrompt		$\mathbf{86.51_{\pm 0.33}}$	$\mathbf{5.16_{\pm 0.09}}$		$\mathbf{68.13_{\pm 0.49}}$	$\mathbf{4.68_{\pm 0.20}}$
Upper-bound	-	$90.85_{\pm 0.12}$	-	-	$79.13_{\pm 0.18}$	-

To make the comparison fair and precise, we first compare DualPrompt with regularization-, rehearsal- and prompt-based methods, which are compatible with transformer-based models, in Table 1. We then compare DualPrompt with architecture-based methods, which are mostly compatible with ConvNets, using a different protocol in Table 2.

- **Comparing methods.** We select representative methods including EWC [19], LwF [27], ER [7,13], GDumb [46], BiC [61], DER++ [3], Co^{2}L [4] and L2P [59], from all categories. Please see Appendix D for details.
- **Naive baselines.** For better demonstration of the relative effectiveness of all methods, we also include: FT-seq, the naive sequential training, and Upper-bound, the usual supervised finetuning on the i.i.d. data of all tasks.

Table 1 reports the performance of all comparing methods on Split CIFAR-100 and Split ImageNet-R. Our proposed method, DualPrompt, outperforms all methods consistently, including non-rehearsal based methods and rehearsal-based methods with a large buffer size. When the buffer size is 5000 (10% of the CIFAR-100 training set and >20% of the ImageNet-R training set), all rehearsal-based methods are fairly close to GDumb, indicating that performing rehearsal-based continual learning likely provide no performance gain than supervised training

Table 2: Comparison with architecture-based methods on Split CIFAR-100. We use `Diff = Upper-Bound Acc - Method Acc` (lower is better), to measure how close the performance to the upper-bound of the used backbone.

Method	Backbone	Avg. Acc (↑)	Diff (↓)	Buffer size	Additional Parameters	
					MB	%
Upper-bound	ResNet18	$80.41^{\dagger}$	-	-	-	-
SupSup [60]		$28.34_{\pm 2.45}^{\ddagger}$	52.07	0	3.0	6.5%
DualNet [44]		$40.14_{\pm 1.64}^{\ddagger}$	40.27	1000	5.04	10.9%
RPSNet [49]		$68.60^{\dagger}$	11.81	2000	181	404%
DynaER [63]		$74.64^{\dagger}$	5.77	2000	19.8	43.8%
Upper-bound	ResNet152	$88.54^{\dagger}$	-	-	-	-
DynaER [63]		$71.01_{\pm 0.58}^{\ddagger}$	17.53	2000	159	68.5%
Upper-bound	ViT-B/16	$90.85_{\pm 0.12}^{\ddagger}$	-	-	-	-
L2P [59]		$83.86_{\pm 0.28}^{\ddagger}$	6.99	0	1.94	0.56%
DualPrompt		$\mathbf{86.51_{\pm 0.33}}$	**4.34**	**0**	**1.90**	**0.55%**

†Reported by the original papers. ‡ Reproduced using their original codebases.

on the buffered data as GDumb does. *DualPrompt achieves better performance without any buffered data.* Moreover, from Table 1, as well as Figure 1, we can observe the performance of rehearsal-based methods drops sharply when the buffer size shrinks. This again suggests the clear advantage of DualPrompt as a rehearsal-free method. For the non-rehearsal based methods, only L2P performs close to our methods. Nevertheless, DualPrompt still beats L2P significantly by a 3%-7% margin on Average accuracy, thanks to our novel design of the two complementary prompts, which successfully reduces catastrophic forgetting.

Architecture-based methods. We compare against representative class-incremental learning methods, including DualNet [44], SupSup [60], DynaER [63] and RPSNet [49]. Please see Appendix D for details.

Prior architecture-based methods, which are based on ConvNet, are not trivial to migrate to transformer-based models. Moreover, different architecture-based methods usually add different amount of additional parameters. To enable a relative fair comparison between these methods, we introduce a metric to measure how close the performance of a certain method is to the upper-bound performance, *i.e.* trained under the i.i.d. setting, of a given architecture. Table 2 shows the results on Split-CIFAR100. DualPrompt achieves the best accuracy and its difference to upper-bound is only 4.34%, with minimal additional parameters and no buffer. The strongest competitor on ResNet, DynaER, on the contrary, requires a buffer of 2,000 images and include 43.8% additional parameters.

5.3   Does stronger backbones naively improve CL?

Our method builds upon more advanced yet bigger backbones than many previous methods. We think understand this question is very important for fair comparison and future research. Although pre-trained ViT is a stronger backbone than common ConvNets, it is not necessarily translate to continual learning performance. The observations we shown here are similar to what reported in a very recent
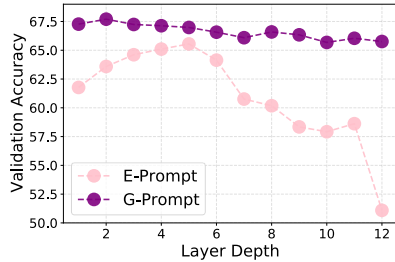
Fig. 3: Effects of position to attach prompts on Split ImageNet-R validation set. We empirically observe that attaching G- and E-Prompts to the 2nd and 5th MSA layer results in the best performance.
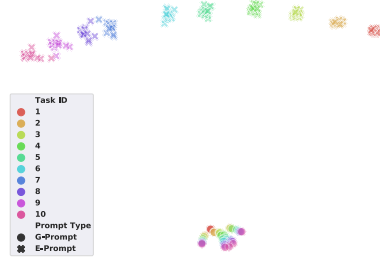
Fig. 4: t-SNE visualization of G- and E-prompts. Each point represents a prompt vector of dimension 768. E-Prompts are taken from the final model, while G-Prompts are taken from model snapshots after trained on each task.

study about how large architecture help continual learning [41]. First, this fact can be seen from Table 1, where well-known general methods still suffer large forgetting rate given this backbone. We have tried to use ImageNet pre-trained ResNet for competing methods in Table 2, which leads to no improvement upon the reported numbers in Table 2. This further indicates that a pre-trained model is not a "panacea" for continual learning without being leveraged properly. We also equip best-performing DynaER [63] with ImageNet pre-trained ResNet152 (60M parameters), which has close upper-bound performance to ViT-B/16 (86M). DynaER learns weight masks, a popular strategy for architecture-based methods [60], to dynamically expand architectures. However, results in Table 2 show worse performance (we sweep their suggested hyper-parameters to report the best performance), a similar observation is shown in their original paper when scaling DynaER to ResNet32. That being said, how to effectively utilize large models under traditional architecture-based methods remains an open question. DualPrompt is novel at wisely leveraging the state-of-the-art vision backbones to solve challenges in continual learning.

5.4   Exploration of where and how to attach prompts

We have shown the best performing model of DualPrompt in Section 5.2. In this section, we explore *where and how* to attach prompts and enhance their influences to the overall performance. We also present critical empirical observations that lead to interesting future research.

Position of prompts. To explore the most proper position to insert the G-Prompt and E-Prompt, we use a heuristic search strategy on the validation set of Split ImageNet-R. We first set $\mathtt{start}_e = \mathtt{end}_e$, *i.e.*, only insert E-Prompt at a single MSA layer. The lower line of Figure 3 shows that placing E-Prompt at the 5th MSA layer leads to the best performance. We then search extend E-Prompt to multi-layer and found that $\mathtt{start}_e = 3, \mathtt{end}_e = 5$ performs the best

Table 3: Comparison of different prompting functions: Prompt Tuning (Pro-T) v.s. Prefix Tuning (Pre-T).

Prompting function		Pro-T	Pre-T
Split CIFAR-100	Avg. Acc ($\uparrow$)	83.81	**86.51**
	Forgetting ($\downarrow$)	5.94	**5.16**
Split ImageNet-R	Avg. Acc ($\uparrow$)	64.99	**68.13**
	Forgetting ($\downarrow$)	6.81	**4.68**

Table 4: Ablation study on Split ImageNet-R. ML means multi-layered.

G-P	E-P	ML	Split ImageNet-R	
			Avg. Acc ($\uparrow$)	Forgetting ($\downarrow$)
			27.01	7.57
$\checkmark$			63.41	6.52
	$\checkmark$		65.10	5.52
$\checkmark$	$\checkmark$		66.77	5.74
$\checkmark$		$\checkmark$	63.85	7.50
	$\checkmark$	$\checkmark$	66.91	4.77
$\checkmark$	$\checkmark$	$\checkmark$	**68.13**	**4.68**

(see Appendix F). We then study the case of $\texttt{start}_g = \texttt{end}_g$ based on the optimal setting of E-Prompt. Interestingly, the upper line of Figure 3 shows that placing G-Prompt at the 2nd MSA layer leads to the best performance. We also extend G-Prompt into its multi-layered counterparts and conduct searching experiments and find the best choice to be $\texttt{start}_g = 1, \texttt{end}_g = 2$ (see Appendix F).

Interestingly, we observe that the best depth are different, and the final layers to attach G- and E-Prompts are non-overlapping. In particular, $\texttt{start}_g > \texttt{start}_e$, which suggests that G-Prompt captures task-invariant knowledge better at shallower layer, while E-Prompt captures task-specific knowledge better at deeper layer. This observation also fits the intuition that different layers in deep learning models capture different types of knowledge [48,65], and thus naturally fit different prompts. This also justifies decoupling positions of G- and E-Prompts as a reasonable option. Moreover, when attaching them to top layers, both E-Prompt and G-Prompt exhibit the worst performance. We speculate prompts need to be attached to shallower layers in order to condition more layers of the pre-trained model and thereby offer effective instructions.

Prompting function: Prompt v.s. Prefix. We further study the role of prompting function on Split CIFAR-100 and Split ImageNet-R. In prior prompt-based CL work, L2P, only Pro-T is applied without further investigation. In Table 3, we observe that Pre-T version leads to a better performance on both datasets. Besides its empirically better performance, Pre-T is actually more scalable and efficient when attached to multiple layers, since it results in unchanged sequence length. Nevertheless, prompting function is a flexible component of our method, and designing better prompting function is also an open research question, so we can easily plug-in any newly proposed prompting function to DualPrompt and evaluate its effectiveness on given continual learning tasks.

5.5   Ablation study

Based on the optimal parameters searched in the previous section, we present the ablation study results in Table 4 to show the importance of each component of DualPrompt on Split ImageNet-R. Note that G-P (G-Prompt) and E-P (E-Prompt) alone represent the optimal single-layered version for each type of prompts ($\texttt{start}_g = \texttt{end}_g = 2, \texttt{start}_e = \texttt{end}_e = 5$), while ML represents the

optimal multi-layered version ($\mathtt{start}_g = 1, \mathtt{end}_g = 2, \mathtt{start}_e = 3, \mathtt{end}_e = 5$). When all components are absent, we simply have a naive baseline with a frozen pre-trained backbone and trainable classification head.

In general, all components contribute to the final performance. Interestingly, adding a single-layered G-Prompt alone brings substantial improvement upon the baseline, indicating that the task-invariant knowledge obtained by G-Prompt generalizes pretty well across tasks. However, simply sharing knowledge between tasks introduces inevitable forgetting, due to the fact that task-specific knowledge is not properly decoupled. Thus, E-Prompt alone consistently outperforms G-Prompt alone, since E-Prompt mitigates forgetting by separating knowledge learned from different tasks. However, only applying E-Prompt ignores the task-invariant knowledge, which helps to learn future tasks. Thus, when adding G-Prompt and E-Prompt together to the backbone, it further enhances the overall performance by selectively decoupling the task-invariant knowledge into G-Prompt and task-specific knowledge into E-Prompt. We also observe that extending both prompts to its multi-layered counterparts helps consistently in all cases, due to the fact that properly adding more prompt parameters through different layers offers more representation power.

Visualization of G- and E-prompts. To further understand different types of instructions learned within G- and E-prompts, we visualize these two types of prompts using t-SNE [33] in Figure 4. For a prompt with shape $L \times D$, we treat it as L prompts with dimension D. E-Prompts are taken from the final model after trained on the sequence of all tasks, while the G-Prompts are taken from different model snapshots after training on each task. We can observe that E-Prompts are well-separated, indicating they are learning task-specific knowledge. Meanwhile, the G-Prompts are quite centered and only differ slightly between tasks, which suggests they are learning task-invariant knowledge.

6   Conclusion

In this paper, we present a novel method, DualPrompt, that achieves rehearsal-free continual learning under the challenging class-incremental setting. DualPrompt presents a novel way to attach complementary prompts to a pre-trained model to learn decoupled knowledge. To comprehensively validate the proposed method, we propose a new continual learning benchmark, Split ImageNet-R, besides study on the widely-used benchmarks. DualPrompt sets state-of-the-art performance in all metrics, surprisingly needs much lower additional memory compared with previous architecture-based and rehearsal-based methods. Empirical investigations are conducted to understand the inner-workings. Since large-scale pre-trained models are widely used in practice for their great representation power, we believe DualPrompt serves as a starting point for real-world rehearsal-free continual learning systems. Moreover, we recommend DualPrompt as a unified framework for future prompt-based continual learning research, for its simplicity, flexibility, and strong performance.

References

1. Aljundi, R., Babiloni, F., Elhoseiny, M., Rohrbach, M., Tuytelaars, T.: Memory aware synapses: Learning what (not) to forget. In: ECCV (2018) 4
2. Bulatov, Y.: notmnist dataset (2011), http://yaroslavvb.blogspot.com/2011/09/notmnist-dataset.html 22
3. Buzzega, P., Boschini, M., Porrello, A., Abati, D., Calderara, S.: Dark experience for general continual learning: a strong, simple baseline. In: NeurIPS (2020) 1, 2, 3, 4, 5, 10, 20
4. Cha, H., Lee, J., Shin, J.: Co2l: Contrastive continual learning. In: ICCV (2021) 2, 4, 10, 20
5. Chaudhry, A., Gordo, A., Dokania, P.K., Torr, P., Lopez-Paz, D.: Using hindsight to anchor past knowledge in continual learning. arXiv preprint arXiv:2002.08165 **2**(7) (2020) 2
6. Chaudhry, A., Ranzato, M., Rohrbach, M., Elhoseiny, M.: Efficient lifelong learning with a-gem. arXiv preprint arXiv:1812.00420 (2018) 5, 9, 10
7. Chaudhry, A., Rohrbach, M., Elhoseiny, M., Ajanthan, T., Dokania, P.K., Torr, P.H., Ranzato, M.: On tiny episodic memories in continual learning. arXiv preprint arXiv:1902.10486 (2019) 2, 3, 5, 10, 20
8. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: CVPR. pp. 248–255. Ieee (2009) 9
9. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., Houlsby, N.: An image is worth 16x16 words: Transformers for image recognition at scale. In: ICLR. OpenReview.net (2021), https://openreview.net/forum?id=YicbFdNTTy 5, 20
10. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al.: An image is worth 16x16 words: Transformers for image recognition at scale. ICLR (2021) 5, 7, 8
11. Ebrahimi, S., Meier, F., Calandra, R., Darrell, T., Rohrbach, M.: Adversarial continual learning. In: ECCV. pp. 386–402. Springer (2020) 1, 3, 4, 21, 22, 23
12. Hadsell, R., Rao, D., Rusu, A.A., Pascanu, R.: Embracing change: Continual learning in deep neural networks. Trends in cognitive sciences (2020) 1, 2, 7
13. Hayes, T.L., Cahill, N.D., Kanan, C.: Memory efficient experience replay for streaming learning. In: ICRA (2019) 10, 20
14. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR. pp. 770–778 (2016) 9
15. Hendrycks, D., Basart, S., Mu, N., Kadavath, S., Wang, F., Dorundo, E., Desai, R., Zhu, T., Parajuli, S., Guo, M., Song, D., Steinhardt, J., Gilmer, J.: The many faces of robustness: A critical analysis of out-of-distribution generalization. arXiv preprint arXiv:2006.16241 (2020) 3, 9
16. Hu, E.J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., Chen, W.: Lora: Low-rank adaptation of large language models. arXiv preprint arXiv:2106.09685 (2021) 4
17. Ke, Z., Liu, B., Huang, X.: Continual learning of a mixed sequence of similar and dissimilar tasks. NeurIPS **33** (2020) 4, 21
18. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014) 19
19. Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A.A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., et al.: Overcoming

catastrophic forgetting in neural networks. PNAS **114**(13), 3521–3526 (2017) 4, 10, 20

20. Kolesnikov, A., Beyer, L., Zhai, X., Puigcerver, J., Yung, J., Gelly, S., Houlsby, N.: Big transfer (bit): General visual representation learning. In: ECCV. pp. 491–507. Springer (2020) 9

21. Krizhevsky, A., Hinton, G., et al.: Learning multiple layers of features from tiny images (2009) 9, 22

22. Kumaran, D., Hassabis, D., McClelland, J.L.: What learning systems do intelligent agents need? complementary learning systems theory updated. Trends in cognitive sciences **20**(7), 512–534 (2016) 3

23. LeCun, Y.: The mnist database of handwritten digits. http://yann. lecun. com/exdb/mnist/ (1998) 22

24. Lester, B., Al-Rfou, R., Constant, N.: The power of scale for parameter-efficient prompt tuning. arXiv preprint arXiv:2104.08691 (2021) 4, 5, 7, 8

25. Li, X.L., Liang, P.: Prefix-tuning: Optimizing continuous prompts for generation. arXiv preprint arXiv:2101.00190 (2021) 4, 7, 8

26. Li, X., Zhou, Y., Wu, T., Socher, R., Xiong, C.: Learn to grow: A continual structure learning framework for overcoming catastrophic forgetting. In: ICML. pp. 3925–3934. PMLR (2019) 1, 4

27. Li, Z., Hoiem, D.: Learning without forgetting. TPAMI **40**(12), 2935–2947 (2017) 4, 10, 20

28. Liu, P., Yuan, W., Fu, J., Jiang, Z., Hayashi, H., Neubig, G.: Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. arXiv preprint arXiv:2107.13586 (2021) 2, 5

29. Liu, X., Ji, K., Fu, Y., Du, Z., Yang, Z., Tang, J.: P-tuning v2: Prompt tuning can be comparable to fine-tuning universally across scales and tasks. arXiv preprint arXiv:2110.07602 (2021) 7

30. Lomonaco, V., Maltoni, D., Pellegrini, L.: Rehearsal-free continual learning over small non-iid batches. In: CVPR Workshops. pp. 989–998 (2020) 2

31. Loo, N., Swaroop, S., Turner, R.E.: Generalized variational continual learning. arXiv preprint arXiv:2011.12328 (2020) 1, 4

32. Lopez-Paz, D., Ranzato, M.: Gradient episodic memory for continual learning. NeurIPS (2017) 9

33. Van der Maaten, L., Hinton, G.: Visualizing data using t-sne. JMLR **9**(11) (2008) 14

34. Mai, Z., Li, R., Jeong, J., Quispe, D., Kim, H., Sanner, S.: Online continual learning in image classification: An empirical survey. arXiv preprint arXiv:2101.10423 (2021) 1, 4, 9, 20, 24

35. Mallya, A., Davis, D., Lazebnik, S.: Piggyback: Adapting a single network to multiple tasks by learning to mask weights. In: ECCV. pp. 67–82 (2018) 1, 4

36. Mallya, A., Lazebnik, S.: Packnet: Adding multiple tasks to a single network by iterative pruning. In: CVPR (2018) 1, 4

37. Masana, M., Liu, X., Twardowski, B., Menta, M., Bagdanov, A.D., van de Weijer, J.: Class-incremental learning: survey and performance evaluation on image classification. arXiv preprint arXiv:2010.15277 (2020) 1

38. McClelland, J.L., McNaughton, B.L., O'Reilly, R.C.: Why there are complementary learning systems in the hippocampus and neocortex: insights from the successes and failures of connectionist models of learning and memory. Psychological review **102**(3), 419 (1995) 3

39. McCloskey, M., Cohen, N.J.: Catastrophic interference in connectionist networks: The sequential learning problem. In: Psychology of learning and motivation, vol. 24, pp. 109–165. Elsevier (1989) 1

40. Mehta, S.V., Patil, D., Chandar, S., Strubell, E.: An empirical investigation of the role of pre-training in lifelong learning. ICML Workshop (2021) 20, 23

41. Mirzadeh, S.I., Chaudhry, A., Yin, D., Nguyen, T., Pascanu, R., Gorur, D., Farajtabar, M.: Architecture matters in continual learning. arXiv preprint arXiv:2202.00275 (2022) 12

42. Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., Ng, A.Y.: Reading digits in natural images with unsupervised feature learning. In: NIPS (2011) 22

43. Pfeiffer, J., Kamath, A., Rücklé, A., Cho, K., Gurevych, I.: Adapterfusion: Non-destructive task composition for transfer learning. arXiv preprint arXiv:2005.00247 (2020) 4

44. Pham, Q., Liu, C., Hoi, S.: Dualnet: Continual learning, fast and slow. NeurIPS **34** (2021) 2, 3, 4, 5, 11

45. Pham, Q., Liu, C., Sahoo, D., et al.: Contextual transformation networks for online continual learning. In: ICLR (2020) 3, 4, 21

46. Prabhu, A., Torr, P.H., Dokania, P.K.: Gdumb: A simple approach that questions our progress in continual learning. In: ECCV (2020) 2, 10, 20, 24

47. Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., Liu, P.J.: Exploring the limits of transfer learning with a unified text-to-text transformer. JMLR **21**, 1–67 (2020) 5

48. Raghu, M., Unterthiner, T., Kornblith, S., Zhang, C., Dosovitskiy, A.: Do vision transformers see like convolutional neural networks? NeurIPS **34** (2021) 7, 13

49. Rajasegaran, J., Hayat, M., Khan, S.H., Khan, F.S., Shao, L.: Random path selection for continual learning. NeurIPS **32** (2019) 11

50. Rebuffi, S.A., Kolesnikov, A., Sperl, G., Lampert, C.H.: icarl: Incremental classifier and representation learning. In: CVPR. pp. 2001–2010 (2017) 9

51. Ridnik, T., Ben-Baruch, E., Noy, A., Zelnik-Manor, L.: Imagenet-21k pretraining for the masses. arXiv preprint arXiv:2104.10972 (2021) 9

52. Rusu, A.A., Rabinowitz, N.C., Desjardins, G., Soyer, H., Kirkpatrick, J., Kavukcuoglu, K., Pascanu, R., Hadsell, R.: Progressive neural networks. arXiv preprint arXiv:1606.04671 (2016) 4

53. Serra, J., Suris, D., Miron, M., Karatzoglou, A.: Overcoming catastrophic forgetting with hard attention to the task. In: ICML. pp. 4548–4557 (2018) 1, 2, 4

54. Shokri, R., Shmatikov, V.: Privacy-preserving deep learning. In: Proc SIGSAC conference on computer and communications security (2015) 2, 4

55. Smith, J., Balloch, J., Hsu, Y.C., Kira, Z.: Memory-efficient semi-supervised continual learning: The world is its own replay buffer. arXiv preprint arXiv:2101.09536 (2021) 2

56. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need. NeurIPS (2017) 5, 8

57. Wang, R., Tang, D., Duan, N., Wei, Z., Huang, X., Cao, G., Jiang, D., Zhou, M., et al.: K-adapter: Infusing knowledge into pre-trained models with adapters. arXiv preprint arXiv:2002.01808 (2020) 4

58. Wang, Z., Jian, T., Chowdhury, K., Wang, Y., Dy, J., Ioannidis, S.: Learn-prune-share for lifelong learning. In: ICDM (2020) 1, 4, 7

59. Wang, Z., Zhang, Z., Lee, C.Y., Zhang, H., Sun, R., Ren, X., Su, G., Perot, V., Dy, J., Pfister, T.: Learning to prompt for continual learning. CVPR (2022) 2, 4, 5, 7, 10, 11, 20, 21, 22

60. Wortsman, M., Ramanujan, V., Liu, R., Kembhavi, A., Rastegari, M., Yosinski, J., Farhadi, A.: Supermasks in superposition. arXiv preprint arXiv:2006.14769 (2020) 4, 11, 12
61. Wu, Y., Chen, Y., Wang, L., Ye, Y., Liu, Z., Guo, Y., Fu, Y.: Large scale incremental learning. In: CVPR. pp. 374–382 (2019) 4, 9, 10, 20
62. Xiao, H., Rasul, K., Vollgraf, R.: Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. arXiv preprint arXiv:1708.07747 (2017) 22
63. Yan, S., Xie, J., He, X.: Der: Dynamically expandable representation for class incremental learning. In: CVPR. pp. 3014–3023 (2021) 4, 11, 12
64. Yoon, J., Yang, E., Lee, J., Hwang, S.J.: Lifelong learning with dynamically expandable networks. arXiv preprint arXiv:1708.01547 (2017) 4
65. Zeiler, M.D., Fergus, R.: Visualizing and understanding convolutional networks. In: ECCV. pp. 818–833. Springer (2014) 13
66. Zenke, F., Poole, B., Ganguli, S.: Continual learning through synaptic intelligence. In: ICML (2017) 4
67. Zeno, C., Golan, I., Hoffer, E., Soudry, D.: Task agnostic continual learning using online variational bayes. arXiv preprint arXiv:1803.10123 (2018) 24
68. Zhao, T., Wang, Z., Masoomi, A., Dy, J.: Deep bayesian unsupervised lifelong learning. Neural Networks 149, 95–106 (2022) 4