Convolutional Prompting meets Language Models for Continual Learning

Anurag Roy[1]   Riddhiman Moulick[1]   Vinay K. Verma[2*]   Saptarshi Ghosh[1]   Abir Das[1]

[1]IIT Kharagpur, [2]IML Amazon India

{anurag_roy@,riddhimanmoulick@kgpian.,saptarshi@cse.,abir@cse.}iitkgp.ac.in, vinayugc@gmail.com

Abstract

Continual Learning (CL) enables machine learning models to learn from continuously shifting new training data in absence of data from old tasks. Recently, pretrained vision transformers combined with prompt tuning have shown promise for overcoming catastrophic forgetting in CL. These approaches rely on a pool of learnable prompts which can be inefficient in sharing knowledge across tasks leading to inferior performance. In addition, the lack of fine-grained layer specific prompts does not allow these to fully express the strength of the prompts for CL. We address these limitations by proposing ConvPrompt, *a novel convolutional prompt creation mechanism that maintains layer-wise shared embeddings, enabling both layer-specific learning and better concept transfer across tasks. The intelligent use of convolution enables us to maintain a low parameter overhead without compromising performance. We further leverage Large Language Models to generate fine-grained text descriptions of each category which are used to get task similarity and dynamically decide the number of prompts to be learned. Extensive experiments demonstrate the superiority of* ConvPrompt *and improves SOTA by* $\sim 3\%$ *with significantly less parameter overhead. We also perform strong ablation over various modules to disentangle the importance of different components.* [1]

1. Introduction

In this constantly changing world, computer vision models must adapt to the new and emerging concepts. However, such models often suffer from catastrophic forgetting [15, 31, 40], a phenomenon where previously learned concepts are forgotten when adapting to novel concepts. A trivial solution to this problem is to have separate models for each new task. However, this would require task identities to be available during inference which may not be very practical. Another way to tackle this will be to keep all the data that the model was trained on, add the new data to it, and then train the model again from the beginning. Naturally, this will incur increased storage and computation as well. A line of work attempting to maintain a balance, keeps a few samples from previous tasks and uses them for *rehearsing* the previous concepts while learning a new task [4, 6, 8, 16, 19, 29, 36, 38, 42, 56]. However, storing samples from older tasks may not always be feasible, especially where long-term storage of data is not permitted possibly due to privacy, security or legislative concerns [17]. Therefore, developing *rehearsal-free* CL approaches requiring no storage of old data has come up to be desirable.

Recently, there has been a surge of prompt tuning based approaches [9, 10, 20, 23, 34, 46, 47, 53, 54] that leverage on pre-trained transformers and show promising performance without using any rehearsal data. Prompt tuning, originally introduced in NLP [28] attaches small learnable vectors to a pre-trained frozen model to properly reuse the already learned representations. Although promising for continual learning, these approaches, however, suffer from the following drawbacks – (1) Learning task-specific and task-shared information in separate layers ignores possible interaction between task-specific and task-shared components [43, 49] and (2) Always learning a fixed number of prompts per task irrespective of the tasks' similarity with the previous ones. The redundant prompts can lead to overfitting specifically in cases where the tasks are highly similar usually seen in fine-grained datasets.

In this work, we propose ConvPrompt, which leverages task-specific prompts that is generated by convolution over task-shared parameters. We also exploit similarity between tasks to control the generation of prompts. Specifically, task shared knowledge is modeled by a learnable embedding matrix at each layer. Prompts, on the other hand, are task specific and are generated by convoluting on the task shared embeddings with learnable kernels. The shared embeddings are free to adapt with the different tasks. The prompt generating convolution kernels, on the other hand, are set aside once learned for a task and new set of kernels are employed for the next task. Such a design enables the shared embeddings to capture *common concepts* while allowing the convolution operations to capture the *task-specific concepts* from the common concepts. Moreover, we employ language

---

[1]Project page: https://cvir.github.io/projects/convprompt

models to get similarity between tasks in order to determine the number of trainable convolution kernels. Such an expansion strategy allows the model to be parameter efficient, introducing only the necessary number of new parameters as and when needed. The benefits of our approach are three-fold – (1) It facilitates knowledge transfer across tasks using task-shared embeddings. (2) Convolutional prompt generation promotes efficient adaptation to new tasks with low parameter overhead. (3) Exploiting similar tasks by large language models results in further reduction in parameters and superior performance.

Extensive experimentation across several rehearsal-free continual learning benchmarks shows that our `ConvPrompt` approach achieves significant performance gain compared to state-of-the-art approaches. On average, across the 3 benchmark datasets and different experimental settings, we outperform the state-of-the-art prompt based CL approaches (*e.g.*, CODA-Prompt [46]) by a margin of $\sim 3\%$ while requiring a significantly lower number of parameters. Our contribution can be summarized as follows:

- We propose a local prompt creation mechanism by applying convolution over task-invariant global parameters enabling an efficient transfer of local and global concepts across tasks that helps the new tasks to adapt better.
- We incorporate a novel language based task similarity prediction, for the first time in continual learning, which helps to reduce the model parameters significantly without sacrificing the performance and without adding significant pre-processing overhead as well.
- The extensive ablations and experimentation over the various standard datasets and experimental settings show the superiority of our approach by a significant margin.

2. Related Work

Continual Learning: CL approaches can be classified into 3 broad categories. (1) *Regularization-based methods* tackle catastrophic forgetting by applying regularizers that prioritize the preservation of important parameters associated with previously learned tasks. By minimizing interference between new and old knowledge through the introduction of penalty terms [1, 2, 24, 26, 58, 59] or constraining the direction of parameter update [14, 45], these methods encourage important parameters to remain in close proximity to previous solutions. While regularization based methods have shown promising results involving smaller number of tasks, their performance can be less satisfactory when confronted with challenging scenarios involving large number of tasks. (2) *Dynamic Architecture-based methods* learn new tasks by assigning distinct parameters for each task [13, 55–57]. While these approaches exhibit the capability to learn extended sequences of tasks, they may encounter substantial memory and computational overhead. Also, most approaches under this category require the information of

which task an image belongs to during inference which may be unrealistic. (3) *Rehearsal-based methods* [3, 4, 12, 20–22, 41, 52] store a few representative training samples from previous tasks in a buffer, which is then used for training alongside the current task. Though effective, these approaches are limited by the size of the buffer and the length of the task sequences. These are also not particularly suitable for scenarios with data privacy requirements. In contrast, `ConvPrompt` addresses rehearsal-free continual learning by intelligently utilizing prompts on pre-trained models.

Recently, vision transformers have performed very well in CL [12, 22, 43, 47, 52, 55]. Authors in [33] examined the impact of attention heads while Dytox [12] acquires new skills by expanding special task tokens. LVT [52] introduced inter-task attention in vision transformers. Both methods, however, require additional memory to store previous training instances. MEAT [55] uses a parameter isolation approach while learning new tasks. However, the model's expandability is limited, restricting the number of tasks it can learn and it requires task-ids to be known during inference. ContraCon [43] uses convolutional re-weighting of the self-attention weights. However, inference is costly as it uses augmentation-based entropic task identification.

Prompt Learning: Prompt-based CL offers robust protection against catastrophic forgetting by incorporating a small number of model instructions called prompts, rather than directly modifying encoder parameters [9, 10, 20, 23, 34, 46, 47, 53, 54]. Initial approaches like L2P [54] and Dual-Prompt [53] employ a prompt pool from which prompts are selected. These methods match input data to prompts without relying on task identification, using local clustering based optimization. Recently, S-Prompts [51] used prompts to continually learn in a domain incremental learning scenario, which involves learning the same set of classes under co-variate distribution shifts. CODA-Prompt builds on [53, 54] and applies soft attention on the prompts towards end-to-end continual learning. ProgressivePrompts [37] progressively learns new prompt tokens for each incoming tasks, but assumes the presence of task-id during inference. A contemporary work, LGCL [23], uses handcrafted prompts from outside and contrastively learns to bring the output representation of the transformer to the prompt. However, without indigenous prompt learning, this approach can act only as a plugin to existing CL approaches with incremental improvement in performance. Our approach, `ConvPrompt`, stands out from the rest in its proficiency in knowledge-sharing across tasks, its ability for on-the-fly prompt generation and its effective handling of the additional parameters required.

3. Preliminaries

Continual Learning: Continual Learning (CL) trains models on tasks arriving sequentially over time without forgetting the previous ones. Each task $t \in \{1, \ldots, T\}$ contains train-
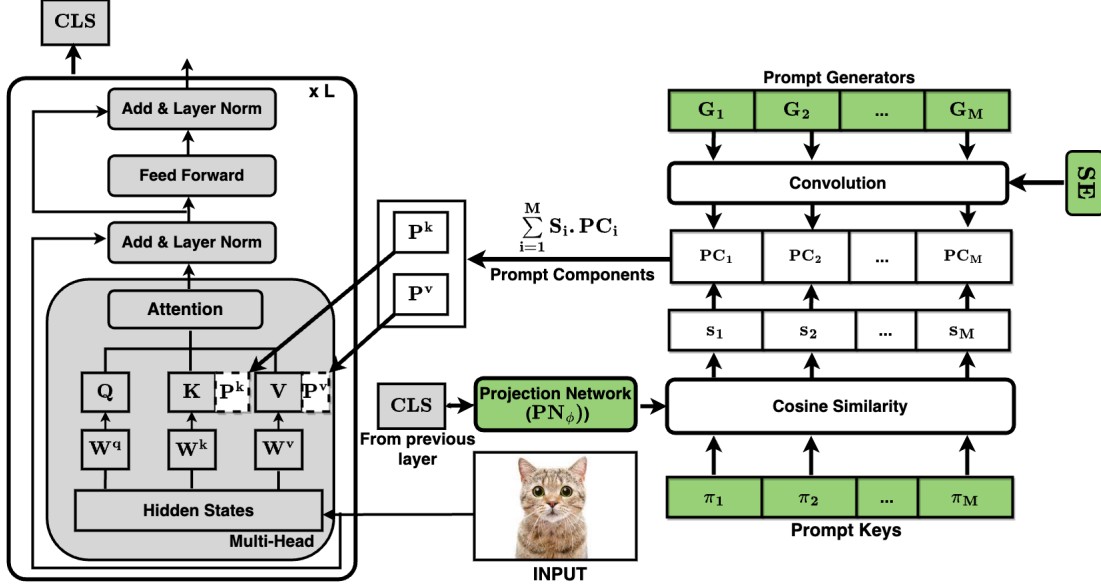
Figure 1. **The proposed `ConvPrompt` architecture.** The layerwise [cls] embedding is passed through the projection network to generate task-specific query representations which are then matched with the prompt-keys using cosine similarity to get the similarity values of prompt-generators. The prompt generators are applied over the shared embeddings to generate input specific prompt components. A weighted average of these prompt components are calculated with the corresponding cosine similarity value as weights to get the input specific key and value prompts which are applied to the pre-trained model. The learnable components are highlighted in **green**, the frozen components are highlighted in **grey**.

ing samples $\{(x_i^t, y_i^t)\}$ where x_i^t is the ith sample of the tth task and $y_i^t \in C^t$ is the corresponding label. The set of the class labels for different tasks are mutually exclusive, *i.e.*, $C^0 \cap C^1 \ldots \cap C^T = \phi$. We address the challenging *rehearsal-free* and *Class-Incremental Learning* (CIL) setting of CL where a trained model f needs to predict the label $y = f(x)$ for an unseen test sample x, regardless of its task and without access to training data from previous tasks.

Transformer Architecture: We build our approach on top of a pre-trained Vision Transformer (ViT) [11]. A transformer starts by dividing an input image x into a set of N fixed-sized patches $z_1 \in \mathbb{R}^{N \times d}$ which are then embedded into a d-dimensional space with positional encoding. A single encoder layer of ViT consists of stacks of Multi-Head Self-Attention (MHSA), layer normalization and Feed Forward Network (FFN) blocks with residual connections. Given the input z_l at the lth layer, the output z_{l+1} is generated that goes into the next layer as input for $l \in \{1, 2, \cdots L\}$ where L is the total number of encoder layers. At the lth layer, MHSA block computes self-attention on the input z_l by using H separate self-attention heads. Self-attention values from head $h \in \{1, 2, \cdots H\}$ at layer l is given by,

$$A(Q_{l,h}, K_{l,h}, V_{l,h}) = softmax\left(\frac{Q_{l,h}K_{l,h}^T}{\sqrt{d_k}}\right)V_{l,h} \quad (1)$$

where $Q_{l,h} = z_l W_{l,h}^Q$, $K_{l,h} = z_l W_{l,h}^K$ and $V_{l,h} = z_l W_{l,h}^V$ are query, key, and value with learnable weights $W_{l,h}^Q$, $W_{l,h}^K$ and $W_{l,h}^V \in \mathbb{R}^{d \times d_h}$ respectively. $d_h = d/H$ is the dimension

of key, query and value vectors. The activations from different attention heads are then concatenated and residually added to the input z_l before performing layer normalization. The resulting activations are passed through a FFN block consisting of two linear layers and an activation function (usually GELU). After another residual connection and layer normalization, the output z_{l+1} at the lth layer is generated.

Prompt and Prefix Tuning: Prefix or prompt tuning aims to learn continuous vectors where the pre-trained transformer is kept frozen. It prepends l_p learnable vectors to the original keys and values of the self-attention heads at every layer. Specifically, l_p length prefix vectors $P_{l,h}^{(K)}; P_{l,h}^{(V)} \in \mathbb{R}^{l_p \times d_h}$ are concatenated with the original key $K_{l,h}$ and value $V_{l,h}$ respectively. Then the self-attention values from head h at layer l is computed as $A(Q_{l,h}, [P_{l,h}^{(K)}, K_{l,h}], [P_{l,h}^{(V)}, V_{l,h}])$ following Eqn. 1. Unlike existing prompt based approaches learning directly the additional vectors [46, 53, 54], our work focuses on creating prompts by maintaining a balance between old knowledge of the system and new information to be put on an ad-hoc basis.

4. Methodology

In this work, we propose a prompt-based CL approach (`ConvPrompt`) by generating prompt vectors for each new task in combination with knowledge learned previously (ref. Fig. 1). Knowledge from previous tasks is modeled by a *learnable embedding matrix shared between all tasks*. The task-specific prompts are created by employing convolution

operation on the shared embedding matrix. While existing works [53, 54] have shown to perform well by prompting a pre-trained transformer, they rely on a single set of prompt vectors needing to compress all necessary information of an image into one single set. Rather than one set of l_p prompt vectors, following CODA-Prompt [46], we have M such sets which we call *prompt components*. However, unlike CODA-Prompt, the prompt components are *not* directly learned in our approach. Instead, they are generated from previous task knowledge by employing M convolution kernels (known as the *prompt generators*) in each head of each layer. A weighted combination of the prompt components provides the final l_p prompt vectors where the weights come from the [CLS] embedding at each layer of the ViT corresponding to the input image. The weighing not only allows us to optimize the model end-to-end but also provides a unique blend of previous task knowledge and the input image.

4.1. Prompt Generation

The prompt vectors in our approach are dynamically generated by small convolution operations. Convolution is performed between two learnable components – (i) Shared Embeddings and (ii) Prompt Generators. Corresponding to each head $h \in \{1, 2, \cdots H\}$ in each layer $l \in \{1, 2, \cdots L\}$ of the transformer, there are shared embedding matrices $SE_{l,h}^K$ and $SE_{l,h}^V$ respectively for the keys and the values. These matrices are shared across tasks. The prompt generators are single channel convolution kernels which are applied on the shared embeddings. We learn a set of M prompt generators for each head in each layer, for both keys and values. For head h and layer l, the prompt generators are denoted as $G_{l,h,m}^K$ and $G_{l,h,m}^V$, where $m \in \{1, 2, \cdots M\}$. The shared embeddings are of dimension $(l_p + k - 1) \times (d_h + k - 1)$ where k is the size of the convolution kernels of the prompt generators. The convolution operation results in prompt components $PC_{l,h,m}^K$ and $PC_{l,h,m}^V$ of size $l_p \times d_h$. The convolutional prompt generators not only enables us to maintain a good trade-off between performance and low parameter requirements per task but also makes use of the inherent inductive bias for structured prompt creation [48].

4.2. Prompt Weighting

Instead of compressing all the information into one set of prompt vectors we make use of M prompt components to get the final prompt vectors at each head of each layer. For each prompt component $PC_{l,h,m}^K$ (and $PC_{l,h,m}^V$), we generate weights (between -1 and 1) which can be interpreted as the relative importance of the particular prompt component in blending them together. We employ M learnable keys, referred to as *prompt keys* $\pi \in \mathbb{R}^{d_\pi}$ for this purpose. The prompt keys work on the image features expressed as a nonlinear function of the [CLS] token at each layer. The [CLS] token is passed through a Projec-

tion Network (PN_ϕ), parameterized by ϕ, consisting of a two-layer fully-connected neural network with a ReLU activation [35] in between. The output $PN_\phi(\text{[CLS]})$ maps the input image to the same space as the prompt keys where a cosine similarity is taken between $PN_\phi(\text{[CLS]})$ and the M prompt keys in each layer l resulting in the importance values $\{s_{l,1}, s_{l,2}, \cdots, s_{l,M}\}$. The prompt components obtained as a result of the convolution operation between the prompt generators and shared embeddings are weighed by the M similarity scores to get the final prompts $P_{l+1,h}^{(K)}$ and $P_{l+1,h}^{(V)}$ as follows,

$$P_{l+1,h}^{(K)} = \sum_{m=1}^{M} s_{l,m} PC_{l,h,m}^K \;;\; P_{l+1,h}^{(V)} = \sum_{m=1}^{M} s_{l,m} PC_{l,h,m}^V \quad (2)$$

While existing works have linearly combined prompt components to get the final prompt, the similarity scores were generated using the same final [CLS] token across all layers. However, this requires a full forward pass through the transformer solely for getting the similarity scores and then the final prompts at each layer for the final predictions resulting in a total of two passes. In contrast, we utilize the [cls] embedding from each layer enabling us to generate the final prompts for the subsequent layers in one single pass resulting in a significant reduction of computation. The similarity values generated from the image help different prompt components to focus on specific features towards the final prompt [46]. Building on this, we used a non-linearly learned projection network that better captures complex tasks as shown empirically.

4.3. Language Guided Prompting

To maintain a balance between learning new tasks and preserving knowledge accumulated from old tasks, we make use of both task-shared as well as task-specific parameters. In our work, PN_ϕ and SE act as shared parameters facilitating inter-task information sharing, while with new tasks coming, the previously learnt prompt generators and prompt-keys are frozen, thus making them task-specialized. A new set of these are freshly learned for every incoming task.

Let the total number of prompt generators per layer learnt till the $(t-1)^{th}$ task be M_{t-1}, while the number of prompt generators learnt for task i only is J_i. Naturally, $M_{t-1} = \sum_{i=1}^{t-1} J_i$ and all M_{t-1} prompt generators are kept frozen when the new set of J_t prompt generators are learned. Notwithstanding the prompt generators' role to mitigate catastrophic forgetting, it is also crucial to keep the increase in parameters in check with increasing number of tasks. Ideally, if a task is similar to a task seen earlier, then the prompt generators learnt previously can be reused. Thus, in contrast to previous works [46] which learn a fixed number of prompt components per task, we learn a dynamic

Tasks	Split CIFAR-100		Split CUB-200		$N_{param}(\downarrow)$
Method	$A_T(\uparrow)$	$F_T(\downarrow)$	$A_T(\uparrow)$	$F_T(\downarrow)$	**Train/Total**
Joint-FT (upper bound)	93.22 ± 0.16	–	88.00 ± 0.15	–	100/ 100
Seq-FT	8.6 ± 0.43	42.67 ± 0.13	23.87 ± 0.54	62.52 ± 0.57	100/ 100
ER (buffer size 5000)	82.30 ± 0.42	16.30 ± 0.24	60.73 ± 0.23	8.71 ± 0.65	100/ 100
LwF [27]	64.56 ± 1.23	25.27 ± 1.32	48.73 ± 1.46	25.18 ± 0.31	100/ 100
L2P [54]	82.76 ± 1.17	7.86 ± 0.39	62.21 ± 1.92	7.12 ± 0.33	**0.7/ 100.7**
L2P + LGCL [23]	84.33 ± 0.06	5.83 ± 0.23	–	–	**0.7/ 100.7**
DualPrompt [53]	85.07 ± 0.49	5.57 ± 0.20	66.00 ± 0.57	**4.4 ± 0.31**	1.3/ 101.3
DualPrompt + LGCL [23]	87.23 ± 0.21	5.10 ± 0.15	–	–	1.3/ 101.3
CODA-Prompt [46]	87.00 ± 0.38	4.78 ± 0.24	74.40 ± 0.74	6.40 ± 0.34	4.6/ 104.6
ConvPrompt	**88.87 ± 0.33**	**4.75 ± 0.15**	**80.2 ± 0.52**	$\underline{5.6 \pm 0.38}$	2.0/ 102.0

Table 1. **Results (%) on CIFAR-100 and CUB-200**. Reported results are for 10 tasks with a supervised ImageNet-21k pretrained ViT as the backbone. A_T denotes the average accuracy and F_T denotes the forgetting. N_{param} denotes the percentage of trainable/final parameters w.r.t that of the ViT model. The N_{param} values are dynamic for our approach and the reported value (2.0/102.0) is the average of the values in Split-CIFAR-100 (2.2/102.2) and Split CUB-200 (1.8/101.8).

number of prompt generators depending on the similarity of the task with the previous ones. The task similarity can be naively modeled by comparing the visual features of the images from different tasks. We present an alternative framework to get task similarity cheaply using Large Language Models (LLMs) such as GPT-3 [5] which show remarkable world knowledge on a variety of topics.

Our key insight is that we can use *language* as a tool to get descriptions of visual attributes of different classes and use these to find task similarity. Visual attributes are additional semantic knowledge that articulate the visual concepts associated with each categories. For example, some visual attributes of *bee* are 'black and yellow stripes', 'two pairs of wings', 'three body segments', *etc*. Instead of manually writing these, we queried GPT-3 to get the attributes for each set of classes in each task as they arrived. This is computationally cheap, requires no additional training, and is scalable to large number of classes. Inspired by works like [30, 32], the class attributes are generated by using the query – "*What are useful features for distinguishing a [class name] in a photo?*" for each class in a task. We generate the BERT embeddings of these attributes and store them in a pool for all seen tasks. For each attribute of the current task t, cosine similarity with all stored attribute embeddings till task $t-1$ is computed. The similarity sim_t of task t with the previous tasks, is the mean of such maximum similarity values across all attributes of task t. Let J_{max} denote the maximum prompt generators per task. Then J_t is given by $(1 - sim_t)J_{max}$ *i.e.*, higher similarity enables to have lower number of prompts. Leveraging linguistic knowledge, we reduce the learnable parameters if the classes in a task have a high overlap with the previously encountered ones.

4.4. Regularization and Final Objective

To prevent over-writing of concepts captured by previous tasks in global task-shared PN_ϕ and SE, we need to ensure that while learning the current task, these parameters deviate

less from the previous tasks. To achieve this, when learning for task t, we regularize the set of parameters ϕ_t of the projection network and the shared embeddings SE_t, to have low l_1 norm with that of the previous task as follows:

$$\mathcal{L}_r(\phi_t, \phi_{t-1}) = ||\phi_{t-1} - \phi_t||_1$$
$$\mathcal{L}_r(SE_t, SE_{t-1}) = ||SE_{t-1} - SE_t||_1 \quad (3)$$

SE_t denote the shared embedding parameters of task t for all heads and layers combined and for both keys and values. Note that suffix t used with the shared embeddings may indicate that SE is task specific, they are not. *Shared* signifies the same set of embeddings is used to learn the *shared* semantics among different tasks. We incrementally update the same SE for each task, using a copy of the SE from the immediately preceding task for regularization in Eqn. 3 that is discarded after training. ϕ_t denote the projection network parameters at the tth task. The final objective is:

$$\mathcal{L}_{cls}(f(x), y) + \mathbb{1}(t>1)\lambda[\mathcal{L}_r(\phi_t, \phi_{t-1}) + \mathcal{L}_r(SE_t, SE_{t-1})] \quad (4)$$

where $\mathcal{L}_{cls}$ denotes classification loss, t the task-id and $\lambda \in [0, 1]$ the hyper-parameter to weigh the loss components. The indicator function $\mathbb{1}(t>1)$ denotes the fact that regularization is applied after the first task.

5. Experiments

Datasets: We evaluate ConvPrompt on the benchmark datasets, ImageNet-R [18] and CIFAR-100 [25], as well as on the fine-grained dataset, CUB-200 [50] in the CIL setup. ImageNet-R [18] is formed from 200 subcategories of ImageNet [44] but with images from different domains such as cartoon, graffiti and origami. It also includes some hard examples from ImageNet that standard models fail to classify. It contains 24,000 training images and 6,000 test images. Following [53, 60], we split the 200 classes into 10 tasks with each task containing 20 classes respectively. CIFAR-100 [25], a widely used dataset in continual learning, contains 100 classes, with each having 500 training and

Tasks	5 Task		10 Task		20 Task		$N_{param}(\downarrow)$
Method	$A_T(\uparrow)$	$F_T(\downarrow)$	$A_T(\uparrow)$	$F_T(\downarrow)$	$A_T(\uparrow)$	$F_T(\downarrow)$	**Train/Total**
Joint-FT (upper bound)	79.6 ± 0.87	–	79.6 ± 0.87	–	79.6 ± 0.87	–	100/ 100
Seq-FT	21.82 ± 0.85	76.26 ± 0.37	11.42 ± 0.76	78.32 ± 0.64	8.75 ± 0.42	82.21 ± 0.96	100/ 100
ER (buffer size 5000)	70.53 ± 0.68	17.47 ± 0.35	64.32 ± 0.65	22.35 ± 0.97	53.26 ± 0.83	34.21 ± 0.82	100/ 100
LwF [27]	49.75 ± 0.65	41.36 ± 0.27	39.27 ± 1.92	51.23 ± 0.34	30.29 ± 1.82	60.32 ± 0.86	100/ 100
L2P [54]	67.43 ± 0.11	5.12 ± 0.62	63.49 ± 0.40	6.85 ± 0.42	59.38 ± 0.50	5.89 ± 0.36	**0.7**/ 100.7
L2P + LGCL [23]	–	–	62.51 ± 0.05	8.9 ± 0.17	–	–	**0.7**/ 100.7
DualPrompt [53]	70.42 ± 0.88	4.1 ± 0.33	68.50 ± 0.52	5.14 ± 0.18	63.21 ± 0.49	5.28 ± 0.45	1.3/ 101.3
DualPrompt + LGCL [23]	–	–	69.46 ± 0.04	4.2 ± 0.06	–	–	1.3/ 101.3
CODA-Prompt [46]	75.38 ± 0.34	6.08 ± 0.36	74.24 ± 0.56	4.92 ± 0.21	70.86 ± 0.42	6.87 ± 0.25	4.6/ 104.6
ConvPrompt	**79.10 ± 0.47**	**3.08 ± 0.11**	**77.86 ± 0.25**	**4.33 ± 0.24**	**75.1 ± 0.39**	**4.1 ± 0.29**	2.2/ 102.2

Table 2. **Results (%) on ImageNet-R**. Reported results are for 5, 10 and 20 tasks splits of ImageNet-R with a supervised ImageNet-21k pretrained ViT as the backbone. A_T denotes the average accuracy and F_T denotes the forgetting. N_{param} denotes the percentage of trainable/final parameters w.r.t that of the ViT model. The N_{param} values vary for ConvPrompt with varying number of tasks and the reported value is the average of the values for 5-tasks (1.64/101.64), 10-tasks (2.0/102.0) and 20 tasks (2.88/102.88).

100 test images. Following [46, 53, 54, 60], we use the 10 task setup of CIFAR-100 with each task containing 10 classes. CUB-200 [50] is a fine-grained dataset containing 200 classes of different bird species with 5994 training images and 5794 test images. Following [60], we use the 10 task setup of CUB-200 with each task containing 20 classes.

Training and Implementation Details: We use the ViT-B/16 [11] model pre-trained on the ImageNet-21k [39] as the backbone over which ConvPrompt is applied. Our projector network is a two-layer neural network having $d/2$ and $d/4$ neurons in these two layers respectively where the input ([CLS] token) is d-dimensional. A ReLU [35] activation function is applied between the two layers. Our approach applies prompts to 7 layers of the pre-trained ViT as our ablation experiments show that more layers with prompts does not help improve the performance although parameter overhead increases. We train each task in CIFAR-100, ImageNet-R and CUB-200 for 10, 10 and 60 epochs respectively. The hyperparameter J_{max} is set to 5, acting as an upper limit for the maximum number of prompt components needed per task. The hyperparameter λ, which is used to weigh the regularization terms in Eqn. 4 is set to 0.01. The experiments leading to our choice of λ and other additional experimental results can be found in the supplementary material. We present our results after conducting five random trials, where task orders were randomly selected for each run. The $mean \pm std$ values are reported.

Metrics Used: We report Average accuracy A_T and Forgetting F_T calculated over the T tasks for all our experiments. Specifically, after training on T tasks is completed, A_T and F_T are calculated as follows:

$$A_T = \frac{1}{T} \sum_{t=1}^{T} S_{t,T}$$

$$F_T = \frac{1}{T-1} \sum_{t=1}^{T-1} \max_{t' \in \{1,...,T-1\}} (S_{t,t'} - S_{t,T})$$
(5)

where $S_{t,T}$ is the test classification accuracy on task t after the model has been trained on task T. In other words, average accuracy measures the average accuracy of all the tasks after training on the last task and forgetting measures the average drop in accuracy of a task after training on the last task from its maximum accuracy attained.

Comparisons: We evaluated our approach against several rehearsal free approaches. These include Learning without Forgetting (LwF) [27], Learning to Prompt (L2P) [54], DualPrompt [53] and CODA-Prompt [46]. We also evaluated against LGCL [23], which uses language based prompts and acts as a plugin to L2P and DualPrompt. Additionally, we also compared our approach with a rehearsal-based approach, Experience Replay (ER) [7] with buffer size 5000. In addition, we report Joint-FT and Seq-FT performances as they serve as bounds of the performance in many situations. In Joint-FT, the ViT model is trained jointly on the training data of all the tasks combined, serving as an upper bound on the performance. Seq-FT represents fine-tuning the ViT model sequentially using only the new task's training data and thus is severely affected by catastrophic forgetting.

5.1. Results and Analysis

As shown in Table 1 and Table 2, ConvPrompt outperforms both rehearsal-free and rehearsal-based approaches significantly. On average, ConvPrompt outperforms the existing state-of-the-art CODA-Prompt [46] by $\sim 3\%$ while using only $\sim 40\%$ of the trainable parameters used by CODA-Prompt. Our approach shows slightly more forgetting in the 10-task setup of Split-CUB. However, even with more forgetting, our approach is able to outperform the existing approaches by atleast 3% margin thereby confirming that our design enables efficient adaptation to new tasks by preventing overfitting that leads to higher maximum accuracy by the tasks. This indicates that our convolutional prompt creation mechanism is able to utilize the shared inter-task concepts better than CODA-Prompt [46] and DualPrompt [53].

Dataset	SLCA	SLCA + `ConvPrompt`
Split CIFAR-100	91.53 ± 0.28	90.60 ± 0.35
Split ImageNet-R	77.00 ± 0.33	78.5 ± 0.37
Split CUB-200	84.71 ± 0.40	87.12 ± 0.31

Table 3. **Results with SLCA:** We report the A_T values for SLCA and the SLCA + `ConvPrompt`. `ConvPrompt` when applied on top of SLCA, improves its performance by $1 - 2\%$.

Method	$A_T(\uparrow)$	$F_T(\downarrow)$	$N_{param}(\downarrow)$
Upper Bound	79.7 ± 0.15	–	100/100
ViT (linear probing) (Lower Bound)	62.1 ± 0.52	5.74 ± 0.23	0.02/100
$+ SE$	67.3 ± 0.58	5.19 ± 0.21	0.7/100.7
$+ SE + $ NN	73.82 ± 0.84	9.92 ± 0.48	14.9/114.9
$+ SE + $ Conv	73.92 ± 0.65	9.43 ± 0.33	3.4/103.4
$+ SE + $ Conv $+ PN$ (1-layer)	76.28 ± 0.23	4.46 ± 0.17	3.5/103.5
$+ SE + $ Conv $+ PN$ (2-layer)	77.96 ± 0.54	4.75 ± 0.57	3.7/103.7
ViT $+ SE + $ Conv $+$ PN (2-layer) $+$ Task-Sim (`ConvPrompt`)	77.86 ± 0.25	4.33 ± 0.24	2.0/102.0

Table 4. **Ablation over ImageNet-R 10 tasks:** We report A_T and F_T averaged for 5 trials. We also report the number of trainable params of each of the variants. ViT denotes the ViT pre-trained on ImageNet-21k over which we apply the different components.

Comparison with SLCA: We also conducted a comparison with a recent SOTA approach *Slow Learner with Classifier Alignment* (SLCA) [60] that does not use prompts for CL. Instead, it finetunes the whole network with smaller learning rate for the represenation layers and larger learning rate for the classification layer. This approach has demonstrated superior performance compared to existing continual prompt-tuning methods. As it involves full network tuning, it is computationally expensive making it impractical in resource-constrained scenarios. In contrast, our approach is viable in situations where compute is limited. In this experiment, we show that our approach can also exploit differential learning rates and classifier alignment as SLCA and provide further improvement in performance. Specifically, in such cases, we learn the prompts along with the transformer weights during fine-tuning. The results in Table 3 demonstrate that our approach combined with SLCA, outperforms it in two out of the three datasets achieving the new state-of-the-art while in the CIFAR-100 dataset it is almost at par with SLCA.

5.2. Ablation Studies and Other Analysis

We perform all our ablation studies on the 10 task setup of the ImageNet-R dataset unless otherwise mentioned. Throughout the study, we consider the ImageNet-21k pre-trained ViT-B/16 model onto which we gradually integrate our modules to showcase their importance. Table 4 shows the results.
Exploiting Shared Embedding across Tasks: Adding shared embedding on top of the ViT-B/16 baseline, already gives better performance than L2P [54] that uses a pool of prompts (ref. row '$+SE$'). This shows that knowledge shared across tasks, when properly regularized, can itself be better than purely individual prompts.
Role of Prompt Generators: Next, we add the task-specific

prompt generators on top of the task-shared embeddings. We experimented with two functional forms of the prompt generators – a) Convolution kernels and b) Neural Networks. Both types show significant improvement (ref. rows '$+SE+NN$' and '$+SE+$Conv'). However, using a neural network is naturally heavier on compute compared to a convolutional prompt generator as shown in the rightmost column. Overall, the addition of task-specific prompt generators leads to a significant improvement of $\sim 6\%$ over the task-shared-only approach and an improvement of $\sim 3\%$ over DualPrompt [53] which also uses both task shared and task specific knowledge. This reaffirms our hypothesis that prompt generation over shared embeddings helps in the effective sharing of inter-task concepts to better adaptation of newer tasks.
Significance of Prompt Weighing: We then add our prompt weighting mechanism which further gives an improvement of $\sim 3\%$ (ref. two rows starting with '$+SE+$Conv$+PN$'). This signifies that our prompt weighting exploits important prompts better. To investigate if a non-linearity in projector network helps, we compare the performances of a linear projection network (single layer fully-connected neural net) with a non-linear one (two-layer fully-connected neural net with a ReLU in between) and observe that the non-linearity naturally is better with complex visual data albeit with a slight increase in parameter count.
Language helps reduce parameters: Finally, we add language driven task-similarity to dynamically determine the number of prompt generators (ref. last row of Table 4) which makes the full `ConvPrompt`. This leads to significant parameter reduction while maintaining the same performance. In datasets containing very similar tasks (*e.g.*, CUB-200) the performance also gets boosted ($\sim 1\%$) possibly due to less overfitting as a result of reduced number of parameters.
Effect of Prompt Length: We analyze the effect of the length (l_p) of prompt vectors on the performances of different prompt based CL approaches including ours. For this purpose, we ran experiments with different prompt lengths, increasing the length in multiples of 4 from 4 to 40. As can be seen in Fig. 2a, `ConvPrompt` performs the best across all values of l_p and the performance has an increasing trend till the prompt length is 20 after which it saturates. So, we used $l_p = 20$, unless otherwise mentioned.
Effect of Increasing Layers to Prompt: To analyze if prompting every layer helps we applied prompts to different layers of the pre-trained backbone for `ConvPrompt` and the closely related approaches (ref. Fig. 2b). Specifically, starting with prompts to the first 5 layers, we go on to apply prompts till the last layer (*i.e.*, 12th layer). As seen, `ConvPrompt`'s performance peaks at 7 layers before stagnating while for DualPrompt and CODA-Prompt performance decreases after 5 layers and then stagnates.
Effect of number of Prompt Generators: We analyze the effect of increasing the maximum number of prompt genera-
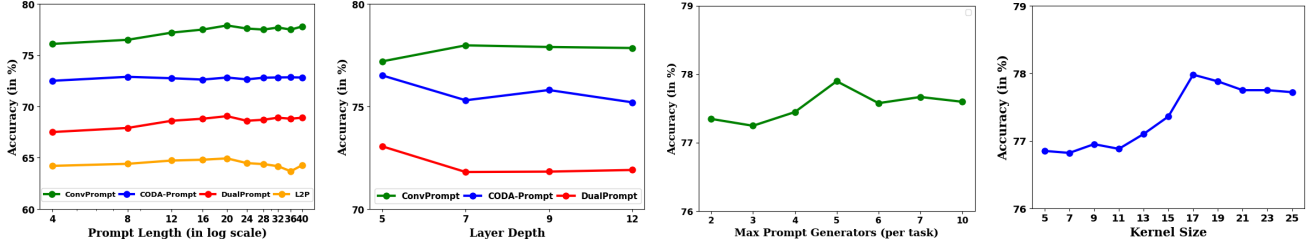
Figure 2. **Ablation over Split ImageNet-R 10 tasks:** (a) average accuracy vs prompt length: The performance peaks at 8 for CODA-P and at 20 for rest of the models (b) average accuracy vs number of layers prompts are applied to: The performance for `ConvPrompt`, peaks at 7 layers while it peaks at 5 for the other models. (c) Average accuracy vs maximum number of prompt components per task: The performance of `ConvPrompt` peaks at $J_{max} = 5$ prompts per task, with the final prompt count after 10 tasks reaching to 18 owing to task-similarity. (d) Average accuracy vs kernel size of the prompt creators: The performance for `ConvPrompt`, peaks at kernel size 17.

Method	$A_T(\uparrow)$	$F_T(\downarrow)$	$N_{param}(\downarrow)$
class-label based sim	76.93 ± 0.45	4.49 ± 0.57	$2.22/102.22$
image-based sim	77.18 ± 0.42	4.38 ± 0.47	$2.29/102.29$
Class Attribute based sim (`ConvPrompt`)	77.86 ± 0.25	4.33 ± 0.24	$2.00/102.00$

Table 5. **ImageNet-R 10 tasks:** Comparison of different task-similarity measures. We report A_T and F_T averaged for 5 trials.

tors J_{max} (ref. Fig. 2c). We observe that the performance peaks at $J_{max} = 5$ after which the performance decreases.
Effect of Kernel Size of Prompt Generators: To understand the impact of the prompt generator's kernel size, we varied the convolution kernel size k from 5 to 25 in steps of 2. As can be seen in Fig. 2d, convolution kernels of size 17 gives the best results while slightly decreasing and stagnating for higher values. We chose kernel size 17 as the default value since it leads to a better trade-off between the number of trainable parameters and performance.

	$P{\rightarrow}C$		$C{\rightarrow}P$	
Datasets	$A_T(\uparrow)$	$F_T(\downarrow)$	$A_T(\uparrow)$	$F_T(\downarrow)$
CIFAR-100	88.87 ± 0.33	4.75 ± 0.15	88.24 ± 0.31	3.86 ± 0.34
ImageNet-R	77.86 ± 0.25	4.33 ± 0.24	77.76 ± 0.28	3.65 ± 0.27
CUB-200	80.2 ± 0.52	5.6 ± 0.38	80.1 ± 0.45	5.7 ± 0.26

Table 6. Prefixes before and after projection on 10-task setup.

Effect of Prompting before or after Projection: We analyze the effect of concatenating the prompt vectors $P_{l,h}^{(K)}$ and $P_{l,h}^{(V)}$ with the original key $K_{l,h}$ and value $V_{l,h}$ respectively vis-a-vis concatenating them with the input z_l which is projected to get the key and values. Performance does not vary much if projection is performed after concatenation ($C{\rightarrow}P$) compared to the other way round ($P{\rightarrow}C$) (Table 6). Naturally, computation is more (by $\sim 0.2B$ MACs) in ($C{\rightarrow}P$) due to bigger matrix-vector multiplications. With low compute overhead and comparable performance, ($P \rightarrow C$) is advantageous and is used in our experiments.
Best way to measure Task Similarity: As similarity between tasks plays a crucial role in performance as well as additional parameters trained, we tried different avenues to measure task similarity. Specifically, we experimented with (i) class-label based task similarity and (ii) image-based task similarity. In the first approach, instead of taking GPT-3

Method	MACs
L2P [54]	$35.85B$
DualPrompt [53]	$33.72B$
CODA-Prompt [46]	$33.72B$
`ConvPrompt`	$17.98B$

Table 7. **Comparison of Inference Times:** MAC (multiply accumulate) operations required for the evaluation of tasks once the model has been fully trained on the 10 tasks of Split ImageNet-R.

generated attributes, we directly used the class labels and in the second we extracted visual features using pretrained ViT-B/16 and used them to measure similarity with previous tasks. Table 5 shows the performance and parameter requirements with these on 10-task ImageNet-R. It can be seen that the attribute-based class similarity leads to the introduction of least number of parameters as well as best performance.
Comparing Inference-Time: We compared the inference time with competing prompt tuning based approaches, namely L2P [54], DualPrompt [53] and CODA-Prompt [46]. Specifically, for a model, we compute the number of MACs (Multiply and Accumulate Operations) an input image consumes during inference after the model has been trained for all the 10 tasks on the ImageNet-R dataset. As can be seen in Table 7, `ConvPrompt` requires the least compute backed by the single pass for prompting through it.

6. Conclusion

In this paper, we proposed `ConvPrompt`, a novel convolutional prompt generation mechanism coupled with a task similarity based expansion strategy for rehearsal-free CL. Different from the existing approaches, our approach creates prompts in each layer by applying convolution over task shared embeddings causing better knowledge transfer across tasks. Moreover, our expansion strategy with LLM driven task similarity ensures that this performance boost is achieved without a significant increase in the number of learnable parameters. Extensive experimentation showed that `ConvPrompt` outperforms SOTA baselines significantly while requiring fewer additional parameters.

References

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

[2] Rahaf Aljundi, Klaas Kelchtermans, and Tinne Tuytelaars. Task-free continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11254–11263, 2019. 2

[3] Rahaf Aljundi, Min Lin, Baptiste Goujaud, and Yoshua Bengio. Gradient based sample selection for online continual learning. *Advances in neural information processing systems*, 32, 2019. 2

[4] Jihwan Bang, Heesu Kim, YoungJoon Yoo, Jung-Woo Ha, and Jonghyun Choi. Rainbow memory: Continual learning with a memory of diverse samples. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8218–8227, 2021. 1, 2

[5] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language Models are Few-Shot Learners. *Advances in neural information processing systems*, 33:1877–1901, 2020. 5

[6] Pietro Buzzega, Matteo Boschini, Angelo Porrello, and Simone Calderara. Rethinking experience replay: a bag of tricks for continual learning. *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 2180–2187, 2021. 1

[7] 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. 6

[8] Arslan Chaudhry, Marcus Rohrbach, Mohamed Elhoseiny, Thalaiyasingam Ajanthan, Puneet K Dokania, Philip HS Torr, and M Ranzato. Continual learning with tiny episodic memories. In *International Conference on Machine Learning*, 2019. 1

[9] Haoran Chen, Zuxuan Wu, Xintong Han, Menglin Jia, and Yu-Gang Jiang. Promptfusion: Decoupling stability and plasticity for continual learning. *arXiv preprint arXiv:2303.07223*, 2023. 1, 2

[10] Marco D'Alessandro, Alberto Alonso, Enrique Calabrés, and Mikel Galar. Multimodal parameter-efficient few-shot class incremental learning. *arXiv preprint arXiv:2303.04751*, 2023. 1, 2

[11] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021. 3, 6

[12] Arthur Douillard, Alexandre Ramé, Guillaume Couairon, and Matthieu Cord. Dytox: Transformers for continual learning with dynamic token expansion. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 2

[13] Verma et. al. Efficient feature transformations for discriminative and generative continual learning. In *CVPR*, 2021. 2

[14] Mehrdad Farajtabar, Navid Azizan, Alex Mott, and Ang Li. Orthogonal gradient descent for continual learning. In *International Conference on Artificial Intelligence and Statistics*, pages 3762–3773. PMLR, 2020. 2

[15] Ian J. Goodfellow, Mehdi Mirza, Da Xiao, Aaron Courville, and Yoshua Bengio. An empirical investigation of catastrophic forgetting in gradient-based neural networks, 2015. 1

[16] Tyler L Hayes, Kushal Kafle, Robik Shrestha, Manoj Acharya, and Christopher Kanan. Remind your Neural Network to Prevent Catastrophic Forgetting. In *European Conference on Computer Vision*, pages 466–483. Springer, 2020. 1

[17] Jiangpeng He and Fengqing Zhu. Exemplar-free Online Continual Learning. *arXiv preprint arXiv:2202.05491*, 2022. 1

[18] Dan Hendrycks, Steven Basart, Norman Mu, Saurav Kadavath, Frank Wang, Evan Dorundo, Rahul Desai, Tyler Zhu, Samyak Parajuli, Mike Guo, Dawn Song, Jacob Steinhardt, and Justin Gilmer. The many faces of robustness: A critical analysis of out-of-distribution generalization. *ICCV*, 2021. 5

[19] Saihui Hou, Xinyu Pan, Chen Change Loy, Zilei Wang, and Dahua Lin. Learning a unified classifier incrementally via rebalancing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 1

[20] Zhiyuan Hu, Jiancheng Lyu, Dashan Gao, and Nuno Vasconcelos. Pop: Prompt of prompts for continual learning, 2023. 1, 2

[21] David Isele and Akansel Cosgun. Selective experience replay for lifelong learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018.

[22] Kishaan Jeeveswaran, Prashant Bhat, Bahram Zonooz, and Elahe Arani. Birt: Bio-inspired replay in vision transformers for continual learning. *arXiv preprint arXiv:2305.04769*, 2023. 2

[23] Muhammad Gul Zain Ali Khan, Muhammad Ferjad Naeem, Luc Van Gool, Didier Stricker, Federico Tombari, and Muhammad Zeshan Afzal. Introducing language guidance in prompt-based continual learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 11463–11473, 2023. 1, 2, 5, 6

[24] 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. 2

[25] Alex Krizhevsky et al. Learning multiple layers of features from tiny images. *Citeseer*, 2009. 5

[26] 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. 2

[27] Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947, 2017. 5, 6

[28] 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. *ACM Computing Surveys*, 55(9): 1–35, 2023. 1

[29] David Lopez-Paz and Marc'Aurelio Ranzato. Gradient Episodic Memory for Continual Learning. *Advances in neural information processing systems*, 30, 2017. 1

[30] Chengzhi Mao, Revant Teotia, Amrutha Sundar, Sachit Menon, Junfeng Yang, Xin Wang, and Carl Vondrick. Doubly right object recognition: A why prompt for visual rationales. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2722–2732, 2023. 5

[31] Michael McCloskey and Neal J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. *Psychology of Learning and Motivation*, 24:109–165, 1989. 1

[32] Sachit Menon and Carl Vondrick. Visual classification via description from large language models. *arXiv preprint arXiv:2210.07183*, 2022. 5

[33] Seyed Iman Mirzadeh, Arslan Chaudhry, Dong Yin, Timothy Nguyen, Razvan Pascanu, Dilan Gorur, and Mehrdad Farajtabar. Architecture matters in continual learning. *arXiv*, 2022. 2

[34] Jun-Yeong Moon, Keon-Hee Park, Jung Uk Kim, and Gyeong-Moon Park. Online class incremental learning on stochastic blurry task boundary via mask and visual prompt tuning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 11731–11741, 2023. 1, 2

[35] Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. In *International Conference on Machine Learning*, page 807–814, Madison, WI, USA, 2010. Omnipress. 4, 6

[36] Ameya Prabhu, Philip HS Torr, and Puneet K Dokania. GDumb: A Simple Approach that Questions our Progress in Continual Learning. In *European conference on computer vision*, pages 524–540. Springer, 2020. 1

[37] Anastasia Razdaibiedina, Yuning Mao, Rui Hou, Madian Khabsa, Mike Lewis, and Amjad Almahairi. Progressive prompts: Continual learning for language models. In *International Conference on Learning Representations*, 2023. 2

[38] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 2001–2010, 2017. 1

[39] Tal Ridnik, Emanuel Ben-Baruch, Asaf Noy, and Lihi Zelnik-Manor. Imagenet-21k pretraining for the masses. *arXiv*, 2021. 6

[40] Anthony V. Robins. Catastrophic forgetting, rehearsal and pseudorehearsal. *Connect. Sci.*, 7:123–146, 1995. 1

[41] David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy Lillicrap, and Gregory Wayne. Experience replay for continual learning. *Advances in Neural Information Processing Systems*, 32, 2019. 2

[42] David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy Lillicrap, and Gregory Wayne. Experience replay for continual

learning. In *Advances in Neural Information Processing Systems*, 2019. 1

[43] Anurag Roy, Vinay Verma, Sravan Voonna, Kripabandhu Ghosh, Saptarshi Ghosh, and Abir Das. Exemplar-free continual transformer with convolutions. In *International Conference on Computer Vision (ICCV)*, 2023. 1, 2

[44] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3), 2015. 5

[45] Gobinda Saha, Isha Garg, and Kaushik Roy. Gradient projection memory for continual learning. In *International Conference on Learning Representations*, 2021. 2

[46] James Seale Smith, Leonid Karlinsky, Vyshnavi Gutta, Paola Cascante-Bonilla, Donghyun Kim, Assaf Arbelle, Rameswar Panda, Rogerio Feris, and Zsolt Kira. Coda-prompt: Continual decomposed attention-based prompting for rehearsal-free continual learning. *arXiv preprint arXiv:2211.13218*, 2022. Accepted for publication at CVPR 2023. 1, 2, 3, 4, 5, 6, 8

[47] Yu-Ming Tang, Yi-Xing Peng, and Wei-Shi Zheng. When prompt-based incremental learning does not meet strong pretraining. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1706–1716, 2023. 1, 2

[48] Yun-Yun Tsai, Chengzhi Mao, and Junfeng Yang. Convolutional Visual Prompt for Robust Visual Perception. In *Neural Information Processing Systems*, 2023. 4

[49] Vinay Kumar Verma, Kevin J Liang, Nikhil Mehta, Piyush Rai, and Lawrence Carin. Efficient feature transformations for discriminative and generative continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13865–13875, 2021. 1

[50] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The caltech-ucsd birds-200-2011 dataset. In *California Institute of Technology*, 2011. 5, 6

[51] Yabin Wang, Zhiwu Huang, and Xiaopeng Hong. S-prompts learning with pre-trained transformers: An occam's razor for domain incremental learning. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2022. 2

[52] Zhen Wang, Liu Liu, Yiqun Duan, Yajing Kong, and Dacheng Tao. Continual learning with lifelong vision transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 171–181, 2022. 2

[53] Zifeng Wang, Zizhao Zhang, Sayna Ebrahimi, Ruoxi Sun, Han Zhang, Chen-Yu Lee, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, et al. DualPrompt: Complementary Prompting for Rehearsal-free Continual Learning. In *European Conference on Computer Vision*, 2022. 1, 2, 3, 4, 5, 6, 7, 8

[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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 139–149, 2022. 1, 2, 3, 4, 5, 6, 7, 8

[55] Mengqi Xue, Haofei Zhang, Jie Song, and Mingli Song. Meta-attention for vit-backed continual learning. In *2022*

IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2022. 2

[56] Shipeng Yan, Jiangwei Xie, and Xuming He. Der: Dynamically expandable representation for class incremental learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 1

[57] Jaehong Yoon, Eunho Yang, Jeongtae Lee, and Sung Ju Hwang. Lifelong learning with dynamically expandable networks. In *International Conference on Learning Representations*. ICLR, 2018. 2

[58] Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *International conference on machine learning*, pages 3987–3995. PMLR, 2017. 2

[59] Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *International Conference on Machine Learning*, pages 3987–3995. PMLR, 2017. 2

[60] Gengwei Zhang, Liyuan Wang, Guoliang Kang, Ling Chen, and Yunchao Wei. Slca: Slow learner with classifier alignment for continual learning on a pre-trained model. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023. 5, 6, 7