CODA-Prompt: COntinual Decomposed Attention-based Prompting for Rehearsal-Free Continual Learning

James Seale Smith[1,2] **Leonid Karlinsky**[2,4] **Vyshnavi Gutta**[1] **Paola Cascante-Bonilla**[2,3]
Donghyun Kim[2,4] **Assaf Arbelle**[4] **Rameswar Panda**[2,4] **Rogerio Feris**[2,4] **Zsolt Kira**[1]
[1]Georgia Institute of Technology [2]MIT-IBM Watson AI Lab [3]Rice University [4]IBM Research

Abstract

Computer vision models suffer from a phenomenon known as catastrophic forgetting when learning novel concepts from continuously shifting training data. Typical solutions for this continual learning problem require extensive rehearsal of previously seen data, which increases memory costs and may violate data privacy. Recently, the emergence of large-scale pre-trained vision transformer models has enabled prompting approaches as an alternative to data-rehearsal. These approaches rely on a key-query mechanism to generate prompts and have been found to be highly resistant to catastrophic forgetting in the well-established rehearsal-free continual learning setting. However, the key mechanism of these methods is not trained end-to-end with the task sequence. Our experiments show that this leads to a reduction in their plasticity, hence sacrificing new task accuracy, and inability to benefit from expanded parameter capacity. We instead propose to learn a set of prompt components which are assembled with input-conditioned weights to produce input-conditioned prompts, resulting in a novel attention-based end-to-end key-query scheme. Our experiments show that we outperform the current SOTA method DualPrompt on established benchmarks by as much as 4.5% in average final accuracy. We also outperform the state of art by as much as 4.4% accuracy on a continual learning benchmark which contains both class-incremental and domain-incremental task shifts, corresponding to many practical settings. Our code is available at https://github.com/GT-RIPL/CODA-Prompt

1. Introduction

For a computer vision model to succeed in the real world, it must overcome brittle assumptions that the concepts it will encounter after deployment will match those learned *a-priori* during training. The real world is indeed *dynamic* and contains continuously emerging objects and categories. Once deployed, models will encounter a range of differences from their training sets, requiring us to continuously update them to avoid stale and decaying performance.

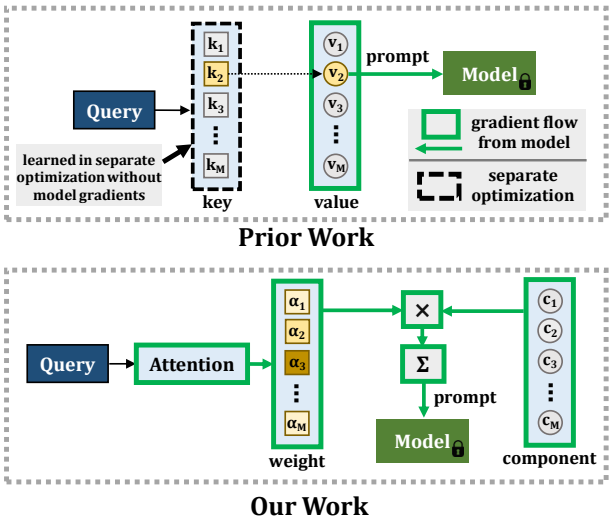

Figure 1. **Prior work** [65] learns a pool of key-value pairs to select learnable prompts which are inserted into several layers of a pre-trained ViT (the prompting parameters are unique to each layer). **Our work** introduces a **decomposed prompt** which consists of learnable prompt *components* that assemble to produce *attention-conditioned* prompts. Unlike prior work, ours is **optimized in an end-to-end fashion** (denoted with thick, green lines).

One way to update a model is to collect additional training data, combine this new training data with the old training data, and then retrain the model from scratch. While this will guarantee high performance, it is not practical for large-scale applications which may require lengthy training times and aggressive replacement timelines, such as models for self-driving cars or social media platforms. This could incur *high financial [26] and environmental [33] costs* if the replacement is frequent. We could instead update the model by only training on the new training data, but this leads to a phenomenon known as *catastrophic forgetting* [46] where the model overwrites existing knowledge when learning the new data, a problem known as *continual learning*.

The most effective approaches proposed by the continual

learning community involve saving [50] or generating [55] a subset of past training data and mixing it with future task data, a strategy referred to as **rehearsal**. Yet many important applications are unable to store this data because they work with *private user data that cannot be stored* long term.

In this paper, we instead consider the highly-impactful and well-established setting of *rehearsal-free* continual learning [38, 56, 57, 65, 66][1], limiting our scope to strategies for continual learning which do not store training data. While rehearsal-free approaches have classically under-performed rehearsal-based approaches in this challenging setting by wide-margins [57], recent *prompt-based approaches* [65, 66], leveraging pre-trained Vision transformers (ViT), have had tremendous success and can even outperform state-of-the-art (SOTA) rehearsal-based methods. These prompting approaches boast a strong protection against catastrophic forgetting by learning a small pool of insert-able model embeddings (prompts) rather than modifying vision encoder parameters directly. One drawback of these approaches, however, is that they cannot be optimized in an *end-to-end* fashion as they use a key and query to select a prompt *index* from the pool of prompts, and thus rely on a second, localized optimization to learn the keys because the model gradient cannot backpropagate through the key/query index selection. Furthermore, these methods reduce forgetting by *sacrificing new task accuracy* (i.e., they lack sufficient *plasticity*). We show that expanding the prompt size does not increase the plasticity, motivating us to grow learning capacity from a new perspective.

We replace the prompt pool with a *decomposed prompt* that consists of a weighted sum of learnable prompt components (Figure 1). This decomposition enables higher prompting capacity by expanding in a new dimension (the number of components). Furthermore, this inherently encourages knowledge re-use, as future task prompts will include contributions from prior task components. We also introduce a novel attention-based component-weighting scheme, which allows for our entire method to be *optimized in an end-to-end fashion* unlike the existing works, increasing our plasticity to better learn future tasks. We boast significant performance gains on existing rehearsal-free continual learning benchmarks w.r.t. SOTA prompting baselines in addition to a challenging *dual-shift* benchmark containing both incremental concept shifts *and* covariate domain shifts, highlighting our method's impact and generality. Our method improves results even under equivalent parameter counts, but importantly can scale performance by increasing capacity, unlike prior methods which quickly plateau. *In summary, we make the following contributions:*

[1]We focus on continual learning over a single, expanding classification head called *class-incremental continual learning*. This is different from the multi-task continual learning setting, known as *task-incremental* continual learning, where we learn separate classification heads for each task and the task label is provided during inference. [25, 61]

1. We introduce a *decomposed* attention-based prompting for rehearsal-free continual learning, characterized by an *expanded learning capacity* compared to existing continual prompting approaches. Importantly, our approach can be optimized in an end-to-end fashion, unlike the prior SOTA.

2. We establish a new SOTA for rehearsal-free continual learning on the well-established ImageNet-R [21, 65] and CIFAR-100 [32] benchmarks, beating the previous SOTA method DualPrompt [65] by as much as 4.5%.

3. We evaluate on a challenging benchmark with dual distribution shifts (semantic *and* covariate) using the ImageNet-R dataset, and again outperform the state of art, highlighting the real-world impact and generality of our approach.

2. Background and Related Work

Continual Learning: Continual learning approaches can be organized into a few broad categories which are all useful depending on the problem setting and constraints. One group of approaches expands a model's architecture as new tasks are encountered; these are highly effective for applications where a model growing with tasks is practical [16, 35, 39, 43, 54]. Another approach is to regularize the model with respect to past task knowledge while training the new task. This can either be done by regularizing the model in the weight space (i.e., penalize changes to model parameters) [2, 15, 30, 58, 71] or the prediction space (i.e., penalize changes to model predictions) [1, 8, 23, 34, 38]. Regularizing knowledge in the prediction space is done using *knowledge distillation* [22] and it has been found to perform better than model regularization-based methods for continual learning when task labels are not given [36, 60].

Rehearsal with stored data [3–5, 7, 9, 10, 18, 19, 24, 29, 41, 50–52, 63, 67] or samples from a generative model [27, 28, 47, 55, 59] is highly effective when storing training data or training/saving a generative model is possible. Unfortunately, for many machine learning applications long-term storage of training data will violate data privacy, as well as incur a large memory cost. With respect to the generative model, this training process is much more computationally and memory intensive compared to a classification model and additionally may violate data legality concerns because using a generative model increases the chance of memorizing potentially sensitive data [45]. This motivates us to work on the important setting of *rehearsal-free* approaches to mitigate catastrophic forgetting.

Rehearsal-Free Continual Learning: Recent works propose producing images for rehearsal using deep-model inversion [12, 17, 56, 68]. While these methods perform well compared to generative modeling approaches and simply

rehearse from a small number of stored images, model-inversion is a slow process associated with high computational costs in the continual learning setting, and furthermore these methods under-perform rehearsal-based approaches by significant margins [56]. Other works have looked at rehearsal-free continual learning from an online "streaming" learning perspective using a frozen, pre-trained model [20, 40]. Because we allow our models to train to convergence on task data (as is common for continual learning [67]), our setting is very different.

Continual Learning in Vision Transformers: Recent work [6, 37, 69] has proven transformers to generalize well to unseen domains. For example, one study varies the number of attention heads in Vision transformers and conclude that ViTs offer more robustness to forgetting with respect to CNN-based equivalents [44]. Another [70] shows that the vanilla counterparts of ViTs are more prone to forgetting when trained from scratch. Finally, DyTox [14] learns a unified model with a parameter-isolation approach and dynamically expands the tokens processed by the last layer to mitigate forgetting. For each task, they learn a new task-specific token per head using task-attention based decoder blocks. However, the above works either rely on exemplars to defy forgetting or they need to train a new transformer model from scratch, a costly operation, hence differing from our objective of exemplar-free CL using pre-trained ViTs.

Prompting for Continual Learning: Prompt-based approaches for continual learning boast a strong protection against catastrophic forgetting by learning a small number of insert-able model instructions (prompts) rather than modifying encoder parameters directly. The current state-of-the-art approaches for our setting, DualPrompt [65] and L2P [66], create a pool of prompts to be selected for insertion into the model, matching input data to prompts *without task id* with a local *clustering-like* optimization. We build upon the foundation of these methods, as discussed later in this paper. We note that the recent S-Prompts [64] method is similar to these approaches but designed for *domain-incremental learning* (i.e., learning the same set of classes under covariate distribution task shifts), which is different than the class-incremental continual learning setting of our paper (i.e., learn emerging object classes in new tasks).

3. Preliminaries

3.1. Continual Learning

In our continual learning setting, a model is sequentially shown N tasks corresponding to non-overlapping subsets of semantic object classes. Each class appears in only a single task, and the goal is to incrementally learn to classify new object classes as they are introduced while retaining performance on previously learned classes. To describe our models, we denote θ as a pre-trained vision encoder and ϕ_n as our classifier head with logits corresponding to task n classes. In this paper, we deal with the *class-incremental continual learning setting* rather than the *task-incremental continual learning* setting. Class-incremental continual learning is challenging because the learner must support classification across all classes seen up to task n [25] (i.e., *no task labels are provided to the learner during inference*). Task-incremental continual learning is a simpler *multi-task* setting where the task labels are given during both training and inference.

We also include experiments that contain both *class-incremental* ***and*** *domain-incremental continual learning at the same time*. The difference between this setting and the former is that we also inject *covariate distribution shifts* (e.g., task 1 might include real images and clip-art, whereas task 2 might include corrupted photos and artistic paintings). The motivation is to make continual learning more practical - as in the real world we might encounter these dual types of domain shifts as well.

3.2. Prompting with Prefix-Tuning

In our work, we do not change the technical foundations of *prompting* from the prior state-of-the-art DualPrompt [65]. We instead focus on the *selection and formation* of the prompt (including enabling an end-to-end optimization of all prompting components), and the resulting prompt is used by the vision transformer encoder in an identical fashion to DualPrompt. This allows us to make a fair comparison with respect to our novel contributions.

As done in DualPrompt, we pass prompts to *several* multi-head self-attention (MSA) layers in a pre-trained ViT transformer [13, 62] and use *prefix-tuning over prompt-tuning*[2], which prepends prompts to the keys and values of an MSA layer rather than prepending prompts to the input tokens. The layers in which prompting occurs is set as a hyperparameter, and the prompting parameters are unique between layers. We define a prompt parameter as $\boldsymbol{p} \in \mathbb{R}^{L_p \times D}$ where L_p is the prompt length (chosen as a hyperparameter) and D is the embedding dimension (768). Consider an MSA layer with input $\boldsymbol{h} \in \mathbb{R}^{L \times D}$, and query, key, and values given as $\boldsymbol{h}_Q, \boldsymbol{h}_K, \boldsymbol{h}_V$, respectively. In the ViT model, $\boldsymbol{h}_Q = \boldsymbol{h}_K = \boldsymbol{h}_V = \boldsymbol{h}$. The output of this layer is given as:

$$\begin{aligned} &\mathrm{MSA}(\boldsymbol{h}_Q, \boldsymbol{h}_K, \boldsymbol{h}_V) = \mathrm{Concat}\left(\mathrm{h}_1, \ldots, \mathrm{h}_\mathrm{m}\right) W^O \\ &\text{where } \mathrm{h}_\mathrm{i} = \mathrm{Attention}\left(\boldsymbol{h}_Q W_i^Q, \boldsymbol{h}_K W_i^K, \boldsymbol{h}_V W_i^V\right) \end{aligned} \tag{1}$$

where W^O, W_i^Q, W_i^K, and W_i^V are projection matrices and m is the number of heads. We split our prompt $\boldsymbol{p}$ into $\{\boldsymbol{p}_K, \boldsymbol{p}_V\} \in \mathbb{R}^{\frac{L_p}{2} \times D}$ and prepend them to $\boldsymbol{h}_K$ and $\boldsymbol{h}_V$ with

[2]We refer the reader to sections 4.2 and 5.4 of the the DualPrompt [65] paper for a discussion on this design choice.

prefix-tuning (P-T) as:

$$f_{P-T}(\boldsymbol{p}, \boldsymbol{h}) = \text{MSA}(\boldsymbol{h}_Q, [\boldsymbol{p}_K; \boldsymbol{h}_K], [\boldsymbol{p}_V; \boldsymbol{h}_V]) \quad (2)$$

The result is that we now only train a small number of parameter (the prompts) while leaving the rest of the transformer encoder unmodified. The critical question that now remains is *how to select and update prompts in a continuous fashion?* The next subsection will discuss how prior works select and update prompts.

3.3. L2P and DualPrompt

L2P [66] and DualPrompt [65] select prompts from a pool using an image-wise prompt query. Specifically, these methods use a key-value pair based query strategy[3] to dynamically select instance-specific prompts from a pool of candidate prompts. Each prompt $\boldsymbol{p}_m$ is selected from a pool of learnable keys $\boldsymbol{k}_m \in \mathbb{R}^D$ where M is the size of the prompt pool, based on cosine similarity to an input-conditioned query. Queries are produced as: $q(\boldsymbol{x}) \in \mathbb{R}^D = \theta(\boldsymbol{x})$ where θ is the pretrained ViT encoder and x is the input image[4]. A query $q(\boldsymbol{x})$ is matched to a key $\boldsymbol{k}_m$ by selecting the key with the *highest cosine similarity*, denoted as: $\gamma(q(\boldsymbol{x}), \boldsymbol{k}_m)$. The keys are learned with a separate optimization from the task classification loss by simply maximizing the cosine similarity between each matched $q(\boldsymbol{x})$ and $\boldsymbol{k}_m$, acting as a *clustering-like* approach. However, this approach does not directly update the key with gradients from the task loss, which is critical in our experiments.

4. Method

4.1. Prompt Formation

While prompting has been shown to perform exceptionally well in terms of mitigating catastrophic forgetting, the existing state-of-the-art approach DualPrompt lacks the ability to expand learning capacity within a given task. Specifically, DualPrompt learns a *single prompt* for each new task - regardless if the task is easy (such as learning a handful of new object classes) or complex (such as learning a large number of new object classes). One might try to increase the length of the prompt, but we show in our experiments (Section 5.3) that increasing the prompt length has saturated returns. Intuitively, we instead desire a method where the learning capacity is correlated to the underlying complexity of task data, rather than a single prompt. Additionally, we desire an approach that is end-to-end differentiable, which we conjecture increases the ability to learn a new task with higher accuracy.

We grow our learning capacity by introducing a new axis: *a set of prompt components*. Rather than pick and choose prompts from a pool, we learn a set of prompt components which, via a weighted summation, form a *decomposed*[5] prompt that is passed to the corresponding MSA layer. This allows us to grow our prompting capacity to arbitrary depth and capture the rich underlying complexity of any task while maintaining a fixed prompt length. In addition, prompting in new tasks will inherently reuse the previously acquired knowledge of past tasks rather than initializing a new task prompt from scratch.

Specifically, we replace learnable prompt parameter $\boldsymbol{p}$ with a weighted summation over the prompt components:

$$\boldsymbol{p} = \sum_m \alpha_m \boldsymbol{P_m} \quad (3)$$

where $\boldsymbol{P} \in \mathbb{R}^{L_p \times D \times M}$ is our set of prompt components, M is the length of our set (i.e., the introduced axis of additional capacity), and α is a weighting vector that determines the contribution of each component which will be discussed in the next subsection.

A distinct difference between our method and L2P/DualPrompt is that *all of our learnable parameters are optimized using the classification loss* rather than splitting into two separate loss optimizations. This allows for better end-to-end learning in our method, and we argue this is a key reason why our method performs better in the benchmarks described in Section 5.

4.2. Prompt-Component Weighting

Rather than a pool of prompts, we have a set of prompt components and desire to produce a *weight vector* α given a query $\theta(x)$, rather than a prompt index. We compute the cosine similarity between a query and keys to produce our weighting vector as:

$$\begin{aligned} \alpha &= \gamma(q(\boldsymbol{x}), \boldsymbol{K}) \\ &= \{\gamma(q(\boldsymbol{x}), \boldsymbol{K}_1), \gamma(q(\boldsymbol{x}), \boldsymbol{K}_2), \ldots, \gamma(q(\boldsymbol{x}), \boldsymbol{K}_M)\} \end{aligned} \quad (4)$$

where $\boldsymbol{K} \in \mathbb{R}^{D \times M}$ contains keys corresponding to our prompt components. The intuition here is that the contributing magnitude of each prompt component $\boldsymbol{P_m}$ to the final prompt $\boldsymbol{p}$ is determined by the similarity between the query $q(\boldsymbol{x})$ and corresponding key $\boldsymbol{K_m}$.

The prompt-query matching can be thought of as clustering in a high dimension space D which is a well-known and difficult problem [31]. To mitigate this drawback, we introduce another component to our key-query matching: *attention*. Each $\boldsymbol{P_m}$ has a corresponding attention vector $\boldsymbol{A_m}$ in addition to the key $\boldsymbol{K_m}$. This allows the the query to focus on certain features from the high dimensional query

[3] We note that the difference between L2P and DualPrompt w.r.t. prompt querying is that the size of the prompt pool in L2P is chosen as a hyperparameter, but in DualPrompt it is equal to the number of tasks and, during training, DualPrompt selects the prompt using task-id (and selects the closest key-query match during inference).

[4] The output is read from the class token.

[5] Decomposed in the sense that our prompt is a weighted summation of components.

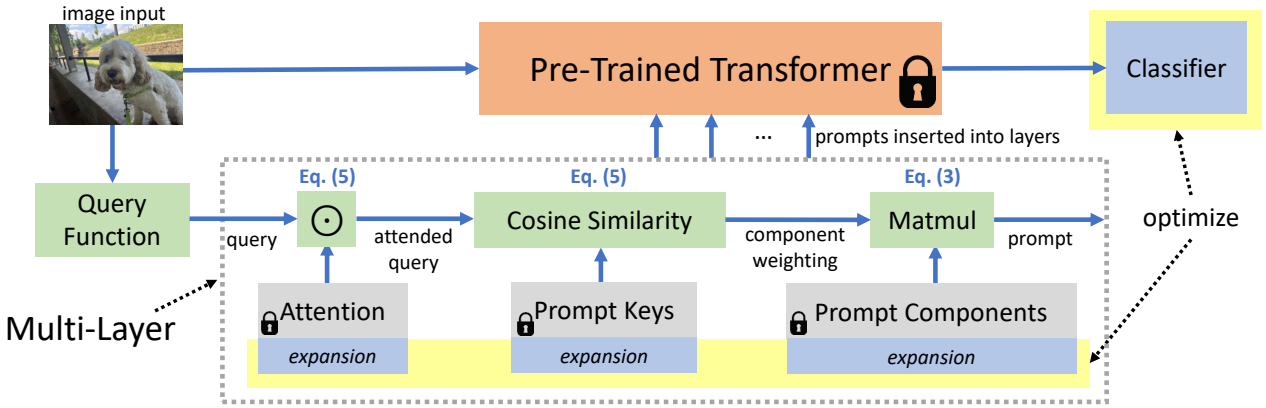

Figure 2. **Our CODA-Prompt approach to rehearsal-free continual learning**. An input (e.g., image) is passed through a query function (we use the pretrained ViT encoder) and used for a novel **attention-based prompt-selection scheme** to produce prompts which are passed to multiple layers of a large-scale, pre-trained transformer. Our method is parameterized by an *expanding* set of small prompt components, with each prompt having a corresponding key and attention vector. Both the input and produced prompts are passed through the transformer and sent to the task head. Only the prompting and task parameters are learned which is *parameter efficient* and no training data is stored for replay which is *memory-efficient and privacy preserving. Importantly, our prompt selection scheme can be optimized end-to-end, unlike prior works which have a local similarity-based optimization for key-query matching.*

$q(\boldsymbol{x})$ output - for example, a prompt component used for mammalian face features to classify dogs and cats would be able to focus on relevant query features such as eye placement and skin texture, and furthermore ignore features of unrelated semantics, such as features useful for automobile classification. We use a simple feature-selection *attention* scheme with element-wise multiplication between the query vector and attention vector to create an *attended-query* which is then used for the key-similarity matching. Specifically, our updated approach to producing our weighting vector is:

$$\begin{aligned}\alpha &= \gamma(q(\boldsymbol{x}) \odot \boldsymbol{A}, \boldsymbol{K}) \\ &= \{\gamma(q(\boldsymbol{x}) \odot \boldsymbol{A}_1, \boldsymbol{K}_1), \ldots, \gamma(q(\boldsymbol{x}) \odot \boldsymbol{A}_M, \boldsymbol{K}_M)\}\end{aligned} \tag{5}$$

where $\boldsymbol{A} \in \mathbb{R}^{D \times M}$ contains learnable parameters (attention vectors) corresponding to our prompt components and $\odot$ is the element-wise product (also known as the Hadamard product). Notice that our attention vectors are learnable feature weightings rather than input-conditioned modules - we found that this fixed representation was less vulnerable to forgetting by its simple design, similar to the our prompt component keys.

4.3. Expansion & Orthogonality

Crucial to mitigating catastrophic forgetting is the avoidance of *overwriting* knowledge acquired in prior tasks. When we visit a new task we freeze our current components and expand the set, only updating the new components. This is visualized in the bottom of Figure 2 where the existing parameters are *locked* and only the expanded parameters are optimized. Specifically, in task t we learn $\frac{M}{N}$ components where N is the number of tasks and M is a chosen hyperparameter, keeping the previous $\frac{(t-1)\cdot M}{N}$ components *frozen*. This expansion is enabled by our attention-based component-weighting scheme, in that expanding our parameters do *not* alter the computation of weights α corresponding previously learned components. For example, expanding the capacity of a nonlinear model like an MLP module, transformer attention head [13], or hypernetwork [63] would alter the weight outputs of previously learned components.

While the prior heuristic is focused on reducing catastrophic forgetting (not destroying previously acquired knowledge), we also want to reduce interference between existing and new knowledge. To do this, we add an orthogonality constraint to $\boldsymbol{P}$, $\boldsymbol{K}$, and $\boldsymbol{A}$. The intuition is that orthogonal vectors have less interference with each other. For example, we would want a key and prompt learned in task 2 to not attract task 1 data, otherwise the encoder output for task 1 data will change, resulting in misclassified data. We introduce a simple orthogonality penalty loss to our method as:

$$\mathcal{L}_{ortho}(B) = ||BB^\top - I||_2 \tag{6}$$

where B represents any arbitrary matrix.

4.4. Full Optimization

Given task classification loss $\mathcal{L}$, our full optimization is:

$$\begin{aligned}\min_{\boldsymbol{P}^{\boldsymbol{n}}, \boldsymbol{K}^{\boldsymbol{n}}, \boldsymbol{A}^{\boldsymbol{n}}, \phi_n} \; & \mathcal{L}(f_\phi(f_{\theta, \boldsymbol{P}, \boldsymbol{K}, \boldsymbol{A}}(\boldsymbol{x})), y) \quad + \\ & \lambda(\mathcal{L}_{ortho}(\boldsymbol{P}) + \mathcal{L}_{ortho}(\boldsymbol{K}) + \mathcal{L}_{ortho}(\boldsymbol{A}))\end{aligned} \tag{7}$$

where $\boldsymbol{P}^{\boldsymbol{n}}, \boldsymbol{K}^{\boldsymbol{n}}, \boldsymbol{A}^{\boldsymbol{n}}$ refer to the prompt components and corresponding keys/attention vectors that are unfrozen and

trained during task n (see Section 4.3) and λ is a hyperparameter balancing the orthogonality loss. Note that we do not update the logits of past task classes, consistent with L2P and DualPrompt. We formally refer to our method as **CO**ntinual **D**ecomposed **A**ttention-based **P**rompting, or simply **CODA-P**.

5. Experiments

We benchmark our approach with several image datasets in the class-incremental continual learning setting. We implemented baselines which do not store training data for rehearsal: Learning without Forgetting (LwF) [38], Learning to Prompt (L2P) [66], and DualPrompt [65]. Additionally, we report the upper bound performance (i.e., trained offline) and performance for a neural network trained only on classification loss using the new task training data (we refer to this as FT), and include an improved version of FT which uses the same classifier implementation[6] as L2P/DualPrompt (referred to as FT++). We also compare to Experience Replay [11] to provide additional context to our results. We implement our method and all baselines in PyTorch [48] using the ViT-B/16 backbone [13] pre-trained on ImageNet-1K [53]. Our contribution includes faithful PyTorch implementations of the popular prompting baselines L2P and DualPrompt, which were released in JAX [65, 66]. Our implementations of the competing methods actually achieve slight boosts in the performance of DualPrompt in most benchmarks, as well as significant performance boosts in L2P due to the improved prompting type[7] (which we refer to as L2P++).

DualPrompt uses length 5 prompts in layers 1-2 (referred to as *general* prompts) and length 20 prompts in layers 3-5 (referred to as *task-expert*) prompts. We insert prompts into the same layers as DualPrompt (layers 1-5) for CODA-P and use a prompt length of 8 and 100 prompt components, which were chosen with a hyperparameter sweep on validation data to have the best trade-off between performance and parameter efficiency. Because our approach introduces more learnable parameters compared to DualPrompt, we include a variant of out method **CODA-P-S** which uses a smaller number of parameters equal to DualPrompt for additional comparison. We show that our method outperforms other methods even under this condition, while still retaining the capability to scale if desired.

We report results on the test dataset, but emphasize that all hyperparameters and design decisions (including for the baselines) were made using validation data (20% of the training data). Unlike DualPrompt, we run our benchmarks for several different shuffles of the task class order and report the mean and standard deviation of these runs. We do this with a consistent seed (different for each trial) so that results can be directly compared. This is crucial because the results with different shuffling order can be different due to differences in task difficulty and their appearance order. *We include additional implementation details and results in the Supplementary Material (SM).*

Evaluation Metrics: We evaluate methods using (1) average final accuracy A_N, or the final accuracy with respect to all past classes averaged over N tasks, and (2) average forgetting [9, 41, 42] F_N, or the drop in task performance averaged over N tasks. The reader is referred to our SM for additional details about these metrics. We emphasize that A_N is the more important metric and encompasses both method plasticity *and* forgetting, whereas F_N simply provides additional context.

5.1. CODA-P sets the SOTA on existing benchmarks

We first evaluate our method and state of art on several well-established benchmarks. Table 1 provides results for ImageNet-R [21, 65] which is composed of 200 object classes with a wide collection of image styles, including cartoon, graffiti, and hard examples from the original ImageNet dataset [53]. This benchmark is attractive because the distribution of training data has significant distance to the pre-training data (ImageNet), thus providing a fair and challenging problem setting. In addition to the original 10-task benchmark, we also provide results with a smaller number of large tasks (5-task) and a larger number of small tasks (20 task). We first notice that our reported performance for DualPrompt is a few percentage points higher compared to the original DualPrompt [65] paper. We re-implemented the method from scratch in a different framework (PyTorch [48]) and suspect the difference has to do with our averaging over different class-order shuffling. We note that our implementation of L2P (L2P++) performs significantly better than originally reported because we use the same form of prompting as DualPrompt (for fair comparison). Furthermore, our "Deep L2P ++", which prompts at the same 5 layers as DualPrompt (rather than only at layer 1), actually performs similarly to DualPrompt. See supplementary for discussion and figures illustrating this.

Importantly, we see that our method has strong gains in average accuracy across all three task lengths, with as much as **4.5% improvement in average accuracy over DualPrompt**. We remind the reader that CODA-P is our proposed method with tuned prompt length and component set size, whereas CODA-P-S is downsized (see SM for exact details) to exactly match the number of learnable parameters as DualPrompt. We notice that our method often has slightly higher average forgetting compared to DualPrompt. Given that average accuracy is the crucial metric that captures practical performance, we

[6] See the SM for additional details

[7] As discussed in Section 3.2, we use pre-fix tuning over prompt-tuning for all implementations as is shown to work best for continual learning [65].

Table 1. **Results (%) on ImageNet-R**. Results are included for 5 tasks (40 classes per task), 10 tasks (20 classes per task), and 20 tasks (10 classes per task). A_N gives the accuracy averaged over tasks and F_N gives the average forgetting. We report results over 5 trials.

Tasks	5		10		20	
Method	A_N (↑)	F_N (↓)	A_N (↑)	F_N (↓)	A_N (↑)	F_N (↓)
UB	77.13	-	77.13	-	77.13	-
ER (5000)	71.72 ± 0.71	13.70 ± 0.26	64.43 ± 1.16	10.30 ± 0.05	52.43 ± 0.87	7.70 ± 0.13
FT	18.74 ± 0.44	41.49 ± 0.52	10.12 ± 0.51	25.69 ± 0.23	4.75 ± 0.40	16.34 ± 0.19
FT++	60.42 ± 0.87	14.66 ± 0.24	48.93 ± 1.15	9.81 ± 0.31	35.98 ± 1.38	6.63 ± 0.11
LwF.MC	74.56 ± 0.59	4.98 ± 0.37	66.73 ± 1.25	3.52 ± 0.39	54.05 ± 2.66	2.86 ± 0.26
L2P++	70.83 ± 0.58	3.36 ± 0.18	69.29 ± 0.73	2.03 ± 0.19	65.89 ± 1.30	1.24 ± 0.14
Deep L2P++	73.93 ± 0.37	2.69 ± 0.10	71.66 ± 0.64	1.78 ± 0.16	68.42 ± 1.20	1.12 ± 0.13
DualPrompt	73.05 ± 0.50	$\mathbf{2.64 \pm 0.17}$	71.32 ± 0.62	1.71 ± 0.24	67.87 ± 1.39	1.07 ± 0.14
CODA-P-S	75.19 ± 0.47	2.65 ± 0.15	73.93 ± 0.49	1.60 ± 0.20	70.53 ± 1.24	1.00 ± 0.15
CODA-P	$\mathbf{76.51 \pm 0.38}$	2.99 ± 0.19	$\mathbf{75.45 \pm 0.56}$	$\mathbf{1.64 \pm 0.10}$	$\mathbf{72.37 \pm 1.19}$	$\mathbf{0.96 \pm 0.15}$

Table 2. **Results (%) on CIFAR-100 and DomainNet**. A_N gives the accuracy averaged over tasks and F_N gives the average forgetting. We report results over 5 trials for CIFAR-100 and 3 trials for DomainNet (due to dataset size).

(a) 10-task CIFAR-100 (10 classes per task)

Method	A_N (↑)	F_N (↓)
Upper-Bound	89.30	-
ER (5000)	76.20 ± 1.04	8.50 ± 0.37
FT	9.92 ± 0.27	29.21 ± 0.18
FT++	49.91 ± 0.42	12.30 ± 0.23
LwF.MC	64.83 ± 1.03	5.27 ± 0.39
L2P++	82.50 ± 1.10	1.75 ± 0.42
Deep L2P++	84.30 ± 1.03	$\mathbf{1.53 \pm 0.40}$
DualPrompt	83.05 ± 1.16	1.72 ± 0.40
CODA-P-S	84.59 ± 0.87	1.76 ± 0.28
CODA-P	$\mathbf{86.25 \pm 0.74}$	1.67 ± 0.26

(b) 5-task DomainNet (69 classes per task)

Method	A_N (↑)	F_N (↓)
Upper-Bound	79.65	-
ER (5000)	58.32 ± 0.47	26.25 ± 0.24
FT	18.00 ± 0.26	43.55 ± 0.27
FT++	39.28 ± 0.21	44.39 ± 0.31
LwF.MC	$\mathbf{74.78 \pm 0.43}$	5.01 ± 0.14
L2P++	69.58 ± 0.39	2.25 ± 0.08
Deep L2P++	70.54 ± 0.51	2.05 ± 0.07
DualPrompt	70.73 ± 0.49	$\mathbf{2.03 \pm 0.22}$
CODA-P-S	71.80 ± 0.57	2.54 ± 0.10
CODA-P	73.24 ± 0.59	3.46 ± 0.09

are not worried by the slight uptick in forgetting. In fact, we argue it is very reasonable and reflects the strength of our approach: our method has a larger capacity to learn new tasks, and thus we can sacrifice marginally higher forgetting. Crucially, we see that as the task sequence length grows, the forgetting metric of our method versus DualPrompt begin to converge to a similar value.

We provide results for experience replay to provide additional context of our results. While the gap between rehearsal with a substantial coreset size (i.e., the buffer size of replay data) and our approach is small for the short task sequence, we see that our method strongly outperforms rehearsal in the longer task sequence.

Tables 2a and 2b provide results on additional[8] benchmarks ten-task CIFAR-100 [32] and five-task DomainNet [49]. While neither dataset is as impactful as ImageNet-R in terms of challenge and distance from the pre-trained distribution, these tables provide additional context to our method's performance. These tables illustrate a similar story to the ImageNet-R benchmarks, with improvements of +3.2% and +2.5%, respectively. We do note that, surprisingly, LwF [38] slightly outperforms prompting on this benchmark.

[8]Note that the hyperparameters for both DualPrompt and CODA-P were tuned on ImageNet-R, which was intentionally done to evaluate robustness of all methods.

5.2. CODA-P sets the SOTA for a new dual-shift benchmark

We also evaluate on a challenging *dual-shift* benchmark using the ImageNet-R dataset. Our motivation is to show robustness to two different types of continual distribution shifts: semantic *and* covariate. We accomplish this by randomly selecting image types from the ImageNet-R dataset to include in each task's training data (while keeping the evaluation data unmodified). The reader is refereed to our SM for more details. Results for this benchmark are provided in Table 3. Compared to 5-task ImageNet-R where our method improves SOTA by 4.5% average accuracy, we see a 4.4% improvement on this 5-task benchmark with similar forgetting margins. Our results indicate that CODA-P better generalizes to real-world type shifts.

Table 3. **Results (%) on ImageNet-R *with covariate domain shifts***. Results are included for 5 tasks (40 classes per task). We simulate domain shifts by randomly removing 50% of the dataset's domains (e.g., clipart, paintings, and cartoon) for the training data of each task (see SM for more details). A_N gives the accuracy averaged over tasks and F_N gives the average forgetting. We report results over 5 trials.

Method	A_N (↑)	F_N (↓)
Upper-Bound	77.13	2.66
ER (5000)	67.39 ± 0.37	11.94 ± 0.17
FT	17.93 ± 0.27	37.49 ± 0.28
FT++	54.51 ± 0.68	14.41 ± 0.33
LwF.MC	64.02 ± 1.55	7.05 ± 0.27
L2P++	65.08 ± 0.29	2.79 ± 0.32
Deep L2P++	65.74 ± 0.12	2.48 ± 0.30
DualPrompt	66.98 ± 0.08	$\mathbf{2.21 \pm 0.28}$
CODA-P-S	69.73 ± 0.18	2.35 ± 0.19
CODA-P	$\mathbf{71.35 \pm 0.08}$	2.56 ± 0.26

Table 4. **Ablation Results (%) on 10-task ImageNet-R**. A_N gives the accuracy averaged over tasks and F_N gives the average forgetting. We report results over 5 trials.

Method	A_N (↑)	F_N (↓)
CODA-P	75.45 ± 0.56	1.64 ± 0.10
Ablate Attention	74.52 ± 0.65	1.67 ± 0.13
Ablate Freezing	74.60 ± 0.64	2.29 ± 0.10
Ablate Orthogonality	70.66 ± 0.60	1.74 ± 0.25

5.3. Ablations and Additional Analysis

We take a closer look at our method with ablation studies and additional analysis. In Table 4 we show the effect of removing a few key components of our method: attention keys, freezing of past task components, and our orthogonality regularization. We show that removing attention slightly reduces forgetting degrades performance on average accuracy (the more important metric). This makes sense because removing the attention keys makes our query processing more aligned with the L2P/DualPrompt methods which boast low forgetting but lack sufficient learning capacity. We see larger drops when removing freezing and orthogonality. This indicates that these aspects are crucial to our approach. Our intuition is is that, without these, our prompt formation is similar to a shallow MLP module which, unregularized, should suffer from high forgetting.

We also look at our ability to grow in model capacity along our newly introduced prompt component dimension using the 5-task ImageNet-R benchmark (with validation data). We show in Figure 3 that, with a number of prompt components equal to the number of prompts learned in DualPrompt, we achieve higher performance. Importantly, when we increase the number of components, we are able to leverage the scaled parameters for significantly higher performance, finding that our method is closer to upper bound performance than it is to DualPrompt in average accuracy. Because the prompt pool in L2P can be increased to arbitrary size as well, we include results in this analysis as well. However, we show that the L2P method peaks with a pool size equal to twice the task sequence length, and then drops in performance. This reflects that our *prompt components* benefit from scale, whereas the existing *prompt pool* actually suffer.

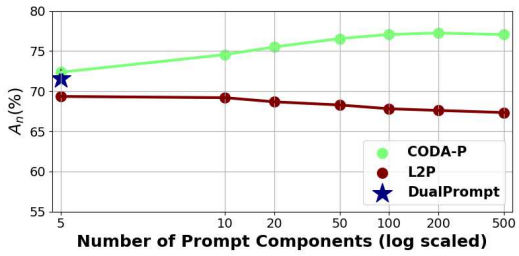

Figure 3. **Analysis of average accuracy A_N vs prompt components (or pool size)** for CODA-P, L2P, and DualPrompt. The setting is 5 task ImageNet-R, and we report the mean over 3 trials.

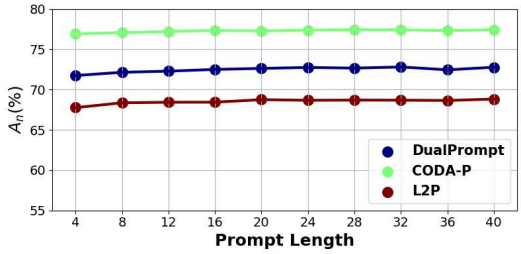

Figure 4. **Analysis of average accuracy A_N vs prompt length** for CODA-P, L2P, and DualPrompt. The setting is 5 task ImageNet-R, and we report the mean over 3 trials.

Finally, we show average accuracy versus prompt *length* in Figure 4 using the same setting as above. The purpose of this experiment is to emphasize that *accuracy saturates with prompt length*, thus motivating the need to expand in our introduced *component* dimension. *Additional analysis and hyperparameter sweeps is available in our SM.*

6. Conclusion

We present *COntinual decomposed attention-based prompting* (**CODA-Prompt**) for *rehearsal-free continual learning*. Our approach assembles learnable prompt *components* which are then inserted into a pre-trained ViT encoder for image classification. Importantly, CODA-Prompt is optimized in an end-to-end fashion (unlike prior SOTA methods which involve two, separate optimizations). Furthermore, CODA-Prompt can *scale prompting capacity to arbitrary sizes*. We set a new SOTA on both well-established benchmarks and a dual-distribution shift benchmark (containing semantic *and* covariate shifts), highlighting the potential real world impact and generality of our approach.

References

[1] Hongjoon Ahn, Jihwan Kwak, Subin Lim, Hyeonsu Bang, Hyojun Kim, and Taesup Moon. Ss-il: Separated softmax for incremental learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 844–853, October 2021. 2

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

[3] Rahaf Aljundi, Eugene Belilovsky, Tinne Tuytelaars, Laurent Charlin, Massimo Caccia, Min Lin, and Lucas Page-Caccia. Online continual learning with maximal interfered retrieval. In *Advances in Neural Information Processing Systems*, pages 11849–11860, 2019. 2

[4] Rahaf Aljundi, Min Lin, Baptiste Goujaud, and Yoshua Bengio. Gradient based sample selection for online continual learning. In *Advances in Neural Information Processing Systems*, pages 11816–11825, 2019. 2

[5] 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*, pages 8218–8227, 2021. 2

[6] 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. 3

[7] Pietro Buzzega, Matteo Boschini, Angelo Porrello, Davide Abati, and Simone Calderara. Dark experience for general continual learning: a strong, simple baseline. *Advances in neural information processing systems*, 33:15920–15930, 2020. 2

[8] Francisco M Castro, Manuel J Marín-Jiménez, Nicolás Guil, Cordelia Schmid, and Karteek Alahari. End-to-end incremental learning. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 233–248, 2018. 2

[9] Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with a-GEM. In *International Conference on Learning Representations*, 2019. 2, 6

[10] Arslan Chaudhry, Marcus Rohrbach, Mohamed Elhoseiny, Thalaiyasingam Ajanthan, Puneet K Dokania, Philip HS Torr, and Marc'Aurelio Ranzato. Continual learning with tiny episodic memories. *arXiv preprint arXiv:1902.10486*, 2019. 2

[11] 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

[12] Yoojin Choi, Mostafa El-Khamy, and Jungwon Lee. Dual-teacher class-incremental learning with data-free generative replay. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3543–3552, 2021. 2

[13] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 3, 5, 6

[14] Arthur Douillard, Alexandre Ramé, Guillaume Couairon, and Matthieu Cord. Dytox: Transformers for continual learning with dynamic token expansion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9285–9295, 2022. 3

[15] Sayna Ebrahimi, Mohamed Elhoseiny, Trevor Darrell, and Marcus Rohrbach. Uncertainty-guided continual learning with bayesian neural networks. *arXiv preprint arXiv:1906.02425*, 2019. 2

[16] Sayna Ebrahimi, Franziska Meier, Roberto Calandra, Trevor Darrell, and Marcus Rohrbach. Adversarial continual learning. *arXiv preprint arXiv:2003.09553*, 2020. 2

[17] Qiankun Gao, Chen Zhao, Bernard Ghanem, and Jian Zhang. R-dfcil: Relation-guided representation learning for data-free class incremental learning. *arXiv preprint arXiv:2203.13104*, 2022. 2

[18] Alexander Gepperth and Cem Karaoguz. Incremental learning with self-organizing maps. *2017 12th International Workshop on Self-Organizing Maps and Learning Vector Quantization, Clustering and Data Visualization (WSOM)*, pages 1–8, 2017. 2

[19] Tyler L Hayes, Nathan D Cahill, and Christopher Kanan. Memory efficient experience replay for streaming learning. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 9769–9776. IEEE, 2019. 2

[20] Tyler L Hayes and Christopher Kanan. Lifelong machine learning with deep streaming linear discriminant analysis. *arXiv preprint arXiv:1909.01520*, 2019. 3

[21] 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. 2, 6

[22] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015. 2

[23] Saihui Hou, Xinyu Pan, Chen Change Loy, Zilei Wang, and Dahua Lin. Lifelong learning via progressive distillation and retrospection. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 437–452, 2018. 2

[24] Saihui Hou, Xinyu Pan, Chen Change Loy, Zilei Wang, and Dahua Lin. Learning a unified classifier incrementally via rebalancing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 831–839, 2019. 2

[25] Yen-Chang Hsu, Yen-Cheng Liu, Anita Ramasamy, and Zsolt Kira. Re-evaluating continual learning scenarios: A categorization and case for strong baselines. *arXiv preprint arXiv:1810.12488*, 2018. 2, 3

[26] Daniel Justus, John Brennan, Stephen Bonner, and Andrew Stephen McGough. Predicting the computational cost

of deep learning models. In *2018 IEEE international conference on big data (Big Data)*, pages 3873–3882. IEEE, 2018. 1

[27] Nitin Kamra, Umang Gupta, and Yan Liu. Deep generative dual memory network for continual learning. *arXiv preprint arXiv:1710.10368*, 2017. 2

[28] Ronald Kemker and Christopher Kanan. Fearnet: Brain-inspired model for incremental learning. *International Conference on Learning Representations (ICLR)*, 2018. 2

[29] Ronald Kemker, Marc McClure, Angelina Abitino, Tyler Hayes, and Christopher Kanan. Measuring catastrophic forgetting in neural networks. *AAAI Conference on Artificial Intelligence*, 2018. 2

[30] 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*, 2017. 2

[31] Mario Köppen. The curse of dimensionality. In *5th online world conference on soft computing in industrial applications (WSC5)*, volume 1, pages 4–8, 2000. 4

[32] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. *Tech Report*, 2009. 2, 7

[33] Alexandre Lacoste, Alexandra Luccioni, Victor Schmidt, and Thomas Dandres. Quantifying the carbon emissions of machine learning. *arXiv preprint arXiv:1910.09700*, 2019. 1

[34] Kibok Lee, Kimin Lee, Jinwoo Shin, and Honglak Lee. Overcoming catastrophic forgetting with unlabeled data in the wild. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 312–321, 2019. 2

[35] Soochan Lee, Junsoo Ha, Dongsu Zhang, and Gunhee Kim. A neural dirichlet process mixture model for task-free continual learning. *arXiv preprint arXiv:2001.00689*, 2020. 2

[36] Timothée Lesort, Hugo Caselles-Dupré, Michael Garcia-Ortiz, Andrei Stoian, and David Filliat. Generative models from the perspective of continual learning. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2019. 2

[37] Duo Li, Guimei Cao, Yunlu Xu, Zhanzhan Cheng, and Yi Niu. Technical report for iccv 2021 challenge sslad-track3b: Transformers are better continual learners. *arXiv preprint arXiv:2201.04924*, 2022. 3

[38] Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947, 2017. 2, 6, 7

[39] Vincenzo Lomonaco and Davide Maltoni. Core50: a new dataset and benchmark for continuous object recognition. *arXiv preprint arXiv:1705.03550*, 2017. 2

[40] Vincenzo Lomonaco, Davide Maltoni, and Lorenzo Pellegrini. Rehearsal-free continual learning over small non-iid batches. In *CVPR Workshops*, pages 989–998, 2020. 3

[41] David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pages 6470–6479, USA, 2017. Curran Associates Inc. 2, 6

[42] Zheda Mai, Ruiwen Li, Jihwan Jeong, David Quispe, Hyunwoo Kim, and Scott Sanner. Online continual learning in image classification: An empirical survey. *Neurocomputing*, 469:28–51, 2022. 6

[43] Davide Maltoni and Vincenzo Lomonaco. Continuous learning in single-incremental-task scenarios. *Neural Networks*, 116:56–73, 2019. 2

[44] Seyed Iman Mirzadeh, Arslan Chaudhry, Dong Yin, Timothy Nguyen, Razvan Pascanu, Dilan Gorur, and Mehrdad Farajtabar. Architecture matters in continual learning. *arXiv preprint arXiv:2202.00275*, 2022. 3

[45] Vaishnavh Nagarajan, Colin Raffel, and Ian J Goodfellow. Theoretical insights into memorization in gans. In *Neural Information Processing Systems Workshop*, 2018. 2

[46] Cuong V Nguyen, Alessandro Achille, Michael Lam, Tal Hassner, Vijay Mahadevan, and Stefano Soatto. Toward understanding catastrophic forgetting in continual learning. *arXiv preprint arXiv:1908.01091*, 2019. 1

[47] Oleksiy Ostapenko, Mihai Puscas, Tassilo Klein, Patrick Jahnichen, and Moin Nabi. Learning to remember: A synaptic plasticity driven framework for continual learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 11321–11329, 2019. 2

[48] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019. 6

[49] Xingchao Peng, Qinxun Bai, Xide Xia, Zijun Huang, Kate Saenko, and Bo Wang. Moment matching for multi-source domain adaptation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 1406–1415, 2019. 7

[50] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H. Lampert. icarl: Incremental classifier and representation learning. In *2017 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR'17, pages 5533–5542, 2017. 2

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

[52] David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy Lillicrap, and Gregory Wayne. Experience replay for continual learning. In *Advances in Neural Information Processing Systems*, pages 348–358, 2019. 2

[53] 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):211–252, 2015. 6

[54] Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016. 2

[55] Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual learning with deep generative replay. In I. Guyon,

U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 2990–2999. Curran Associates, Inc., 2017. 2

[56] James Smith, Yen-Chang Hsu, Jonathan Balloch, Yilin Shen, Hongxia Jin, and Zsolt Kira. Always be dreaming: A new approach for data-free class-incremental learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9374–9384, October 2021. 2, 3

[57] James Seale Smith, Junjiao Tian, Yen-Chang Hsu, and Zsolt Kira. A closer look at rehearsal-free continual learning. *arXiv preprint arXiv:2203.17269*, 2022. 2

[58] Michalis K Titsias, Jonathan Schwarz, Alexander G de G Matthews, Razvan Pascanu, and Yee Whye Teh. Functional regularisation for continual learning with gaussian processes. In *International Conference on Learning Representations*, 2019. 2

[59] Gido M van de Ven, Hava T Siegelmann, and Andreas S Tolias. Brain-inspired replay for continual learning with artificial neural networks. *Nature communications*, 11(1):1–14, 2020. 2

[60] Gido M van de Ven and Andreas S Tolias. Generative replay with feedback connections as a general strategy for continual learning. *arXiv preprint arXiv:1809.10635*, 2018. 2

[61] Gido M Van de Ven and Andreas S Tolias. Three scenarios for continual learning. *arXiv preprint arXiv:1904.07734*, 2019. 2

[62] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. 3

[63] Johannes von Oswald, Christian Henning, João Sacramento, and Benjamin F Grewe. Continual learning with hypernetworks. *arXiv preprint arXiv:1906.00695*, 2019. 2, 5

[64] 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. 3

[65] Zifeng Wang, Zizhao Zhang, Sayna Ebrahimi, Ruoxi Sun, Han Zhang, Chen-Yu Lee, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, et al. Dualprompt: Complementary prompting for rehearsal-free continual learning. *arXiv preprint arXiv:2204.04799*, 2022. 1, 2, 3, 4, 6

[66] Zifeng Wang, Zizhao Zhang, Chen-Yu Lee, Han Zhang, Ruoxi Sun, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, and Tomas Pfister. Learning to prompt for continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 139–149, 2022. 2, 3, 4, 6

[67] Yue Wu, Yinpeng Chen, Lijuan Wang, Yuancheng Ye, Zicheng Liu, Yandong Guo, and Yun Fu. Large scale incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 374–382, 2019. 2, 3

[68] Hongxu Yin, Pavlo Molchanov, Jose M Alvarez, Zhizhong Li, Arun Mallya, Derek Hoiem, Niraj K Jha, and Jan Kautz. Dreaming to distill: Data-free knowledge transfer via deepinversion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8715–8724, 2020. 2

[69] Wenpeng Yin, Jamaal Hay, and Dan Roth. Benchmarking zero-shot text classification: Datasets, evaluation and entailment approach. *arXiv preprint arXiv:1909.00161*, 2019. 3

[70] Pei Yu, Yinpeng Chen, Ying Jin, and Zicheng Liu. Improving vision transformers for incremental learning. *arXiv preprint arXiv:2112.06103*, 2021. 3

[71] Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *International Conference on Machine Learning*, 2017. 2