Steering Prototypes with Prompt-tuning for Rehearsal-free Continual Learning

Zhuowei Li
Rutgers University

Long Zhao
Google Research

Zizhao Zhang
Google Cloud AI

Han Zhang
Google Research

Di Liu
Rutgers University

Ting Liu
Google Research

Dimitris N. Metaxas
Rutgers University

Abstract

In the context of continual learning, prototypes—as representative class embeddings—offer advantages in memory conservation and the mitigation of catastrophic forgetting. However, challenges related to semantic drift and prototype interference persist. In this study, we introduce the ***C****ontrastive* ***P****rototypical* ***P****rompt (CPP) approach. Through task-specific prompt-tuning, underpinned by a contrastive learning objective, we effectively address both aforementioned challenges. Our evaluations on four challenging class-incremental benchmarks reveal that CPP achieves a significant 4% to 6% improvement over state-of-the-art methods. Importantly, CPP operates without a rehearsal buffer and narrows the performance divergence between continual and offline joint-learning, suggesting an innovative scheme for Transformer-based continual learning systems*[1].

1. Introduction

Continual learning [45], defined as the model's ability to sequentially assimilate information from a continuous stream of potentially correlated data, is critical for modern intelligent systems, due to the inherently dynamic nature of the real world [17]. Yet, existing deep neural networks are known to be prone to *catastrophic forgetting* [32], *i.e.* models progressively degrade in performance on previously mastered tasks upon the acquisition of new information.

Among the plethora of strategies proposed to harness continual learning challenges [48], prototypes (*e.g.*, class mean embeddings [44]) exhibit a promising functionality as they can retain previous knowledge in a memory-efficient manner [62] and mitigate bias towards the latest task [40, 58] when coupled with a nearest class mean (NCM) [33] classifier. However, prototypes themselves are also subject to abrupt efficacy drops due to the following challenges:

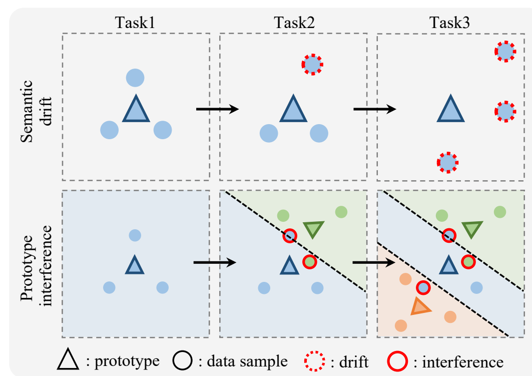

Figure 1. A schematic illustration of *semantic drift* and *prototype interference*. Different colors represent separate classes.

- **Semantic drift**. Sequential task learning using a unified model essentially results in a series of model snapshots, with only the most recent version being retained. This framework introduces a discrepancy between the data sample's embedding at inference time and its prototype—leading to pronounced semantic drift in the embedding space (see Fig. 1 top).

- **Prototype interference**. The emergence of new data samples, bearing high semantic resemblance to previous samples, can distort their embeddings in proximity to pre-existing prototypes, thereby causing interference (see Fig. 1 bottom).

During course of continual learning, both phenomena occur simultaneously and cause *forgetting*.

Although existing methodologies attempt to alleviate the impact of semantic drift—either through the preservation and replay of prior samples [11, 40] or by compensating for drifts post-facto [16, 58]—they often necessitate significant memory overhead or become susceptible to cumulative

[1] Code is available at here.

errors as the sequence of tasks extends. Furthermore, the interference issue is less attended to, and is often left implicitly to additional regularization terms [1, 28, 29]. To bridge these gaps, our approach seeks to synchronously align the embedding functions employed for prototype generation and inference, thereby avoiding semantic drift. Concurrently, we also advocate for a strategic regulation of distances between prototypes in the embedding space to minimize interference.

Drawing inspiration from prompt-tuning [20, 25], a new transfer learning paradigm that enables a tiny portion of additional learnable tokens to adapt a frozen Transformer [49] to down-stream tasks, we propose the use of task-specific prompts to bypass semantic drift. Concretely, we allow a data sample at inference to retrieve its corresponding task-specific prompts and assemble the exact embedding function that is used for generating its corresponding prototype. Under our schema, a frozen, pre-trained Transformer is treated as shared, stabilized global knowledge across tasks, ensuring system stability. Task-specific prompts, on the other hand, learn task-level specializations and keep the system plastic.

To reduce prototype interference in the embedding space, we propose a novel contrastive prototypical loss optimized using task-specific prompts. The crafted learning objective encourages intra-class clustering and increases inter-class distances upon on a mixture of data embeddings and up-to-now prototypes. Our architecture ensures that historical knowledge is assimilated and utilized solely as prototypes and anchor points. Task-specific prompts can, therefore, effectively navigate current prototypes to minimize interference, obviating the need for the storage and replay of explicit samples. We further enhance our framework with the multi-centroid prototype strategy, which deploys a set of fictive embeddings instead of a singular mean embedding, capturing the distributional essence of a class more comprehensively.

We term our method **C**ontrastive **P**rototypical **P**rompt (CPP), a simple and novel continual learning framework that explores embedding space holistically. Emperically, CPP excels in four challenging class-incremental benchmarks including split CIFAR-100, 5-datasets, split ImageNet-subset, and split ImageNet-R, bringing around 4% to 6% absolute improvements over state-of-the-art methods. Notably, CPP is rehearsal-free and consumes at least $5\times$ times fewer additional memories than rivals. Our primary achievements can be summarized as:

- The inception of CPP, a simple and novel framework for rehearsal-free continual learning. It leverages contrastively learned task-specific prompts to effectively address both semantic drift and prototype interference obstacles.
- The introduction of the multi-centroid prototype strategy, an innovative mechanism that enriches the representational prowess of prototypes and is seamlessly integrated into the CPP framework.
- Rigorous empirical validations confirming the supremacy of CPP under a light memory budget. Each crafted component is thoroughly studied and demonstrates clear and additive benefits.

2. Related Work

Continual learning. Prevalent algorithms can typically be categorized into three primary branches [17, 37, 48]. Regularization-based methods strike a balance under *stability–plasticity dilemma*. A subset of this, parameter-regularization methods, either restrict the magnitude [29, 59] or the orientation [14, 42] of parameter space alterations. Conversely, functional-regularization methods confine functional changes concerning certain 'anchor points' in the function space [2, 36, 46]. Notably, their performance tends to wane with extended task sequences [17]. Modular-based methods mitigate knowledge interference by either reallocating existing resources [31, 43] or provisioning additional learning capacities [26, 41, 57]. A caveat is their frequent dependence on test-time task identifiers, making them challenging to scale. In practice, by saving and replaying earlier data instances, rehearsal-based methods have consistently demonstrated versatility and resilience [4, 6]. However, their efficacy can vary based on buffer size [17, 38] and might not be practical for scenarios with tight memory constraints or heightened privacy concerns. Our proposed CPP, integrates the advantages of these diverse methods while strategically sidestepping their inherent limitations.

Prototypes for continual learning. Embeddings have been recognized for their resilience against information loss [10, 58]. Moreover, the parameter-driven linear classifiers often culminate in rapid forgetting, chiefly due to biases in favor of recent tasks [60]. Consequently, a majority of prototype-centric methodologies have harnessed prototypes in combination with the NCM classifier [40, 58, 62]. An alternative strategy, PASS [62], employs prototypes as latent space anchors, curtailing semantic overlaps. Similarly, CPP also deploys prototypes as anchors in the latent space. To empower the prototypes, existing approaches to counteract semantic drift have either preserved explicit exemplars for prototype updates [11, 40] or retroactively compensated for drifts by inferring them from contemporary data [58]. In stark contrast, CPP preemptively obstructs semantic drifts and proactively tackles prototype interference. Further, our methodology favors multi-centroid prototypes over mean embeddings, offering a nuanced representation of contextual distributions.

Prompt tuning. The practice of initializing deep neural networks with pre-trained weights is widely adopt. However, conventional fine-tuning might not consistently enhance performance on downstream tasks [23]. In response, prompt-tuning, initially gaining traction in the NLP domain [25, 27] followed by the adaption to the visual domain [20], has been explored. Its recent induction into continual learning has been evidenced by methodologies such as L2P [54] and DualPromt [53], which capitalize on shared prompt pools or universal prompts for incremental knowledge acquisition. In a similar vein, S-prompts [51] harnesses domain-specific prompts for domain-incremental learning challenges. Our approach uses task-specific prompts to address semantic drift and prototype interference. Importantly, we innovatively integrate prompt-tuning with prototypes, optimizing their combined benefits for continual learning.

3. Methodology

We begin by describing the problem setup and, along the way, introduce the notations in Sec. 3.1. We then proceed to a basic prototype-based framework in Sec.3.2, serving as our baseline model. Building on this baseline, Sec.3.3 describes our advanced CPP approach. The multi-centroid prototype strategy is further discussed in Sec.3.4. An illustrative framework overview can be seen in Fig. 2.

3.1. Problem Setup and Notion

Supervised continual learning can be defined as learning a model over a sequence of T tasks $\mathcal{T}_{1:T} = \{\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_T\}$. Each task $\mathcal{T}_t$ is associated with a dataset $\mathcal{D}^t = \{(\boldsymbol{x}_i^t, y_i^t)_{i=1}^{n_t}\}$ containing n_t data pairs, where $\boldsymbol{x}$ is the input and y is its corresponding label. Each data pair $(\boldsymbol{x}_i^t, y_i^t) \in (\mathcal{X}^t \times \mathcal{Y}^t)$ belongs to an unknown distribution $(\mathcal{X}^t \times \mathcal{Y}^t)$ and $\mathcal{Y}^t \cap \mathcal{Y}^{t'} = \emptyset$ while $t \neq t'$. In general, a neural network at session t can be split into an embedding function $f_{\theta^t}(\cdot) : \mathbb{R}^{W \times H \times C} \rightarrow \mathbb{R}^D$ and a classifier $g_{\phi^t}(\cdot) : \mathbb{R}^D \rightarrow \mathbb{R}^K$ that parameterized by θ^t and ϕ^t, respectively. The overall learning objective is to develop a pair of $f_{\theta^t}(\cdot)$ and $g_{\phi^t}(\cdot)$ that excels in all prior tasks $\mathcal{T}_{1:t}$. Note that in this paper, we focus on the challenging rehearsal-free class-incremental setting, where the task identity t is not available during inference and we do not augment the dataset $\mathcal{D}^t$ of task $\mathcal{T}_t$ by saving and replaying preceding exemplars during the training stage.

3.2. A Training-free Baseline

Let $\mathcal{D}_k^t$ denote the dataset belonging to class k at task t, and the prototype of class k is produced as the class mean embedding following [40]:

$$\boldsymbol{\mu}_k = \frac{1}{|\mathcal{D}_k^t|} \sum_{\boldsymbol{x} \in \mathcal{D}_k^t} f_\theta(\boldsymbol{x}). \tag{1}$$

Here, θ is initialized by a pre-trained ViT [13] and is kept frozen throughout the entire learning process. We then do classification using the nearest-class-mean (NCM) [33] classifier:

$$y^* = \underset{y \in \{1,\ldots,K\}}{\arg\min} \{d(\boldsymbol{u}_y, f_\theta(\boldsymbol{x}))\}, \tag{2}$$

where $d : \mathbb{R}^D \times \mathbb{R}^D \rightarrow \mathbb{R}$ is a distance measurement. In our design, we always define d as the cosine distance (similarity). This straightfoward training-free baseline model produces decent results under an apt embedding function (see Table 3), highlighting the value of the embedding and the potential of prototypes in continual learning.

3.3. Contrastive Prototypical Prompt

3.3.1 Steering prototypes with task-specific prompts

Ideally, a static embedding function can position data samples close to their corresponding prototypes in the embedding space, thus avoiding forgetting with the architecture outlined in Sec. 3.2. Yet, achieving a universally adept embedding function remains elusive. Instead, we permit the embedding function to adapt dynamically to new inputs with minimal forgetting. This is achieved by incorporating a tiny set of learnable tokens (*i.e.*, task-specific prompts), to inform a frozen embedding function.

Specifically, we prepend a prompt $\boldsymbol{p}_i \in \mathbb{R}^{L_p \times D}$ to the existing tokens in the i-th layer of a Transformer, where L_p is the length of the prompt and D denotes the embedding dimension. The computation of the i-th layer is then defined as:

$$[\boldsymbol{c}_i, \boldsymbol{e}_i] = T_i([\boldsymbol{c}_{i-1}, \boldsymbol{p}_{i-1}, \boldsymbol{e}_{i-1}]), \tag{3}$$

where T_i represents a multi-head self-attention block followed by a feed-forward network. Here, $\boldsymbol{c} \in \mathbb{R}^{1 \times D}$ denotes the class token, and $\boldsymbol{e} \in \mathbb{R}^{L_e \times D}$ is the existing data tokens. The operator $[\cdot]$ concatenates along the sequence length dimension. We adopt deep prompt [20] by adding prompts to all S layers. Thus the task-specific prompt for task t is given as $P^t = \{\boldsymbol{p}_1^t, \boldsymbol{p}_2^t, \ldots, \boldsymbol{p}_S^t\}$, and the embedding function for task t can be rewritten as:

$$f_{\theta^t}(\cdot) \rightarrow f_{\{\theta, P^t\}}(\cdot). \tag{4}$$

We maintain a collection of task-specific prompts in the memory space, each paired with distinct key and value prototypes (elaborated in Sec. 3.3.3). Thanks to the design of task-specific prompts, each task has a unique parameter space and the semantic drifts can be avoided by assembling the frozen embedding function with the correct task-specific prompt at inference.

3.3.2 Contrastive prototypical loss (CPL)

To address prototype interference, we introduce a novel loss function tailored to optimize task-specific prompts. Recall that CPP maintains information from different contexts as prototypes and uses the NCM classifier for discrimination. We therefore expect contemporary task-specific prompt can

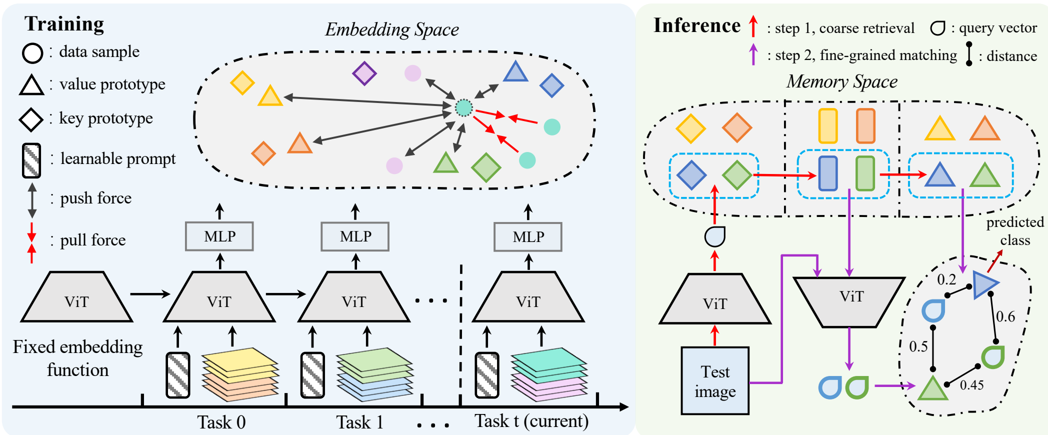

Figure 2. An overview of CPP. Different colors denote different classes. **Left:** As learning progresses, knowledge from prior tasks is anchored as prototypes in the embedding space, with task-specific prompts minimizing prototype interference. **Right:** During inference, a group of candidate prompts (shown as class-specific prompts for illustration purpose) are first retrieved and then followed by a fine-grained matching process.

encouraging intra-class clustering while avoiding overlap with prior prototypes in the embedding space.

Based on this rationale, we present CPL which fortifies intra-class coherence while distancing inter-class entities within a joint space of data embeddings and prototypes. For task t, let $I = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)\}$ denote a batch of N image pairs and $Z = \{\boldsymbol{z}_1, \ldots, \boldsymbol{z}_N\}$ be their corresponding embeddings. Here, omitting subscripts, $\boldsymbol{z}$ is computated as $\boldsymbol{z} = m_{\sigma^t}(f_{\{\theta;P^t\}}(\boldsymbol{x}))$, where $m_{\sigma^t}(\cdot)$ is a multi-layer perceptron (MLP) neck parameterized by σ^t. It is worth noting that $m_{\sigma^t}(\cdot)$ is re-initialized for each new task and disposed during inference. The learning objective is defined as:

$$\mathcal{L} = \frac{1}{N} \sum_{i \in \{1,\ldots,N\}} \mathcal{L}_i, \tag{5}$$

$$\mathcal{L}_i = \frac{-1}{|P(i)|} \sum_{\boldsymbol{z}_p \in P(i)} \log \frac{\exp(\boldsymbol{z}_i \cdot \boldsymbol{z}_p / \tau)}{\sum\limits_{\boldsymbol{z}_n \in N(i) \cup \hat{U}} \exp(\boldsymbol{z}_i \cdot \boldsymbol{z}_n / \tau)}, \tag{6}$$

where $P(i) = \{\boldsymbol{z}_p \in Z : y_p = y_i\}$ denotes a set of positive samples w.r.t. embedding $\boldsymbol{z}_i$, and $N(i) = \{\boldsymbol{z}_n \in Z : y_n \neq y_i\}$ represents a set of negative samples that do not share the same label as $\boldsymbol{z}_i$. We define $\hat{U} = \{\hat{\boldsymbol{u}}_1, \ldots, \hat{\boldsymbol{u}}_k\}$ as a collection of negative anchors, represented by value prototypes from preceding classes. Fig. 2 (left) illustrates the idea of our learning objective. To better restrain the discrimination boundaries, we apply prototype augmentation following [62]. At each iteration, prototypes in U are randomly sampled and perturbed by a scaled Gaussian noise $\hat{\boldsymbol{\mu}} = \boldsymbol{\mu} + m * \boldsymbol{e}$, where $\boldsymbol{e} \sim \mathcal{N}(0, 1)$ and m is a scale factor computed as the average variance of the corresponding class embeddings.

CPL differs from the canonical contrastive loss [9, 21] in two primary ways. First, we employ previous class prototypes as negative anchors, preserving spaces for earlier data and negating prototype interference. Second, we concentrate on aligning positive embeddings, omitting the emphasis on intra-class uniformity, a crucial aspect in conventional contrastive learning [50]. Given the NCM classifier's inclination to select the nearest prototype, enhancing intra-class uniformity might undesirably widen the distance between a sample and its prototype. This rationale can also be understood from an energy-based standpoint; further details are expounded in the supplementary material.

3.3.3 Inference by reemerging model snapshots

In this section, we delineate the inference procedure using CPP. The crux of the procedure is to associate a target data sample with its task-specific prompt, thereby reconstituting the complete embedding function, i.e., a model snapshot. This process is analog to predicting task identities and can be simplified to directly assigning correct task-specific prompt to a given class when the task identities are available.

Considering the prototype of class k in task t, we decouple it into a *key* prototype $\boldsymbol{u}_k = \frac{1}{|D_k^t|} \sum_{\boldsymbol{x} \in \mathcal{D}_k^t} f_\theta(\boldsymbol{x})$ and a *value* prototype $\hat{\boldsymbol{u}} = \frac{1}{|D_k^t|} \sum_{\boldsymbol{x} \in \mathcal{D}_k^t} f_{\{\theta,P^t\}}(\boldsymbol{x})$ corresponding to the class mean embedding before and after learning task t, respectively. We cache both key prototypes $U = \{\boldsymbol{u}_1, \ldots, \boldsymbol{u}_K\}$ and value prototypes $\hat{U} = \{\hat{\boldsymbol{u}}_1, \ldots, \hat{\boldsymbol{u}}_K\}$ of K learned classes. For inference, we start by constructing a coarse query vector $\boldsymbol{q} : \mathbb{R}^{1 \times D}$ of target sample $\boldsymbol{x}$. We subsequently employ a query function $q(\boldsymbol{q}, U, r)$ to locate r nearest key prototypes and retrieve their corresponding prompt sets $\{P^1, \ldots, P^J : J \leq r\}$. Here, $J \leq r$ since distinct classes might possess identical task-specific prompts. $\boldsymbol{q}$ is simply the class token from the last layer: $\boldsymbol{q} = f_\theta(\boldsymbol{x})$ (exact indexing operation is omitted to avoid notion clutter) and

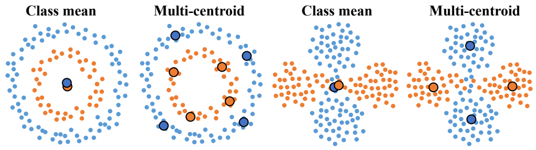

Figure 3. Different colors denote different classes. Small and large circles represent data samples and prototypes, respectively.

the query function measures the pairwise cosine similarity between $\boldsymbol{q}$ and U.

Deeming these retrieved prompts as candidates, we generate a set of fine-grained query vectors $\hat{Q} = \{\hat{\boldsymbol{q}}^j : j \in [1, J]\}$, where $\hat{\boldsymbol{q}}^j = f_{\{\theta, P^j\}}(\boldsymbol{x})$. The final class prediction is made as following:

$$y^* = \arg\min_y \{d(\hat{\boldsymbol{u}}_y, \hat{\boldsymbol{q}}^j) : y \in [1, K], j \in [1, J]\}. \quad (7)$$

Fig. 2 (right) depicts the information flow of the inference process. Detailed algorithms are provided in supplementary.

There are two key steps that ensure the success of the above inference process. First, we need to retrieve a group of candidate prompts that contain the target prompt. This is secured by an appropriate embedding function through which a data sample will locate in the vicinity of its corresponding distribution mass center (empirically validated in supplementary). Second, we expect the retrieved prompts, *i.e.*, both target prompt and mismatched prompts, can attribute. Thanks to our designed contrastive prototypical loss, target prompt will reduce the distance between the sample and its corresponding value prototype, while mismatched prompts act in the opposite way. Hence, the behavior of both target and mismatched prompts is beneficial for the NCM.

3.4. Multi-centroid Prototypes

Prevailing alrothims in continual learning adopt class mean embeddings as prototypes [58, 61, 62]. This practice harbors the belief that latent space distributions are both convex and isotropic. Furthermore, it assumes that the distance function belongs to Bregman divergence [44]. Yet, above premise don't always stand their ground in practice as no strict constraints are imposed on embedding distributions, and the cosine distance is not Bregman divergence (see Fig. 3 for an example).

To this end, we propose to use multi-centroid prototype. The essence of this strategy is the generation of a set of synthetic embeddings to encapsulate the quintessence of the target class distribution. Specifically, we calculate the similarity matrix $\boldsymbol{S}_k : \mathbb{R}^{|\mathcal{D}_k| \times |\mathcal{D}_k|}$ by measuring the pair-wise cosine similarities between embeddings in class k. $\boldsymbol{S}_k$ is then used as the affinity matrix for spectral clustering [35] to generate C centroids $\{\boldsymbol{u}_{k,c}\}_{c=1}^{C}$. In both training and inference stages, we seamlessly substitute class mean embeddings with their corresponding multi-centroid prototypes.

It is worth noting that the multi-centroid strategy is also complementary to CPP's inference framework serving as a hard-case miner. Consider a datum that either strays far from its distribution's mass center or finds itself mired amidst overlapping class distributions; the multi-centroid approach facilitates the retrieval of a broader spectrum of candidate prompts, thereby safeguarding accuracy. Conversely, for data inhabit near their distribution mass centers in a well-separated class territories, this strategy prunes the list of retrieved prompts, boosting efficiency. The upper and lower bounds of the retrieved prompt number are given as r and $\max(1, r/C)$, respectively.

4. Experiments

Experimental section is start with the delineation of datasets and metrics employed. Then, we benchmark CPP in relation to state-of-the-art methodologies. Afterwards, we thoroughly studied each introduced component, affirming their effectiveness. The section is concluded by an analysis of CPP's efficiency from both memory and computational aspects.

4.1. Datasets

Split CIFAR-100 is a standard continual learning benchmark wherein CIFAR-100 is partitioned into 10 discrete tasks. Extended results for alternative splits (e.g., 5 and 20 tasks) are available in the supplementary material.

5-datasets is a collection of datasets consisting of CIFAR-10 [22], MNIST [24], Fashion-MNIST [56], SVHN [34], and notMNIST [3]. Each is designated as a distinct task, mirroring real-world situations with significant differences between tasks.

Split ImageNet-subset is a frequently referenced benchmark that segments a 100-class subset of ImageNet [12] into 10 tasks, each comprising 10 classes. It resembles challenging real-world image with high resolution.

Split ImageNet-R is first adapted for continual learning through DualPrompt [53]. It seeks to replicate real-world conditions characterized by varied image styles and notable intra-class diversity. The original ImageNet-R [19] is divided into 24,000 training and 6,000 test images, with the 200 classes spread across 10 individual tasks.

4.2. Configuration and Evaluation Metric

Configuration. We use the following dataset-agnostic configuration unless stated otherwise. All experiments are conducted on four NVIDIA A100 GPUs. We train CPP for 50 epochs with a batch size of 256 using the AdamW optimizer [30]. The initial learning rate is set to 1×10^{-3} and anneals to 1×10^{-6} according to a cosine scheduler. The prompt length L_p is set to 1 and deep prompt is used by default. The multi-centroid number C and nearest neighbors r are set to 5 and 3, respectively. A 3-layer MLP with

Method	Buffer	Split CIFAR-100 Avg. Acc (↑)	Split CIFAR-100 Forget (↓)	Buffer	5-datasets Avg. Acc (↑)	5-datasets Forget (↓)	Buffer	Split ImageNet-R Avg. Acc (↑)	Split ImageNet-R Forget (↓)
ER [8]		82.53±0.17	16.46±0.25		84.26±0.84	12.85±0.62		65.18±0.40	23.31±0.89
BiC [55]		81.42±0.85	17.31±1.02		85.53±2.06	10.27±1.32		64.63±1.27	22.25±1.73
GDumb [38]	5000	81.67±0.02	-	500	-	-	5000	65.90±0.28	-
DER++ [4]		83.94±0.34	14.55±0.73		84.88±0.57	10.46±1.02		66.73±0.87	20.67±1.24
Co^{2}L [6]		82.49±0.89	17.48±1.80		86.05±1.03	12.28±1.44		65.90±0.14	23.36±0.71
EWC [29]		47.01±0.29	33.27±1.17		50.93±0.09	34.94±0.07		35.00±0.43	56.16±0.88
LwF [28]		60.69±0.63	27.77±2.17		47.91±0.33	38.01±0.28		38.54±1.23	52.37±0.64
L2P [54]		83.86±0.28	7.35±0.38		81.14±0.93	4.64±0.52		61.57±0.66	9.73±0.47
ESN [52]		86.34±0.52	4.76±0.14		85.71±1.47	2.58±0.61		-	-
DualPrompt [53]		86.51±0.33	5.16±0.09		88.08±0.36	2.21±0.69		68.13±0.49	4.68±0.20
CPP (ours)		**91.12**±0.12	**3.33**±0.18		**92.92**±0.17	**0.19**±0.07		**74.88**±0.07	**3.65**±0.03
Upper-bound	-	93.15±0.09	-	-	97.81±0.02	-	-	83.87±0.30	-

Table 1. Comparison to state-of-the-art methods on split CIFAR-100, 5-datasets, and split ImageNet-R. Results of ESN are reported from the original paper [52]. Other results except for the upper-bounds are reported from DualPrompt [53].

2048 hidden units and 768 output dimensions is randomly initialized at each new task. Other detailed configurations are provided in supplementary.

Evaluation metric. We adopt the widely used average accuracy and forgetting from the end session [7, 29, 53] as our evaluation metrics. We report average and standard deviation according to five runs with different random seeds. Detailed illustration of each metric and more results under alternative protocols are relegated to the supplementary.

4.3. Benchmarking Against Leading Approaches

In this section, we first compare CPP to state-of-the-art methods that are compatible with the Transformer architecture on split-CIFAR100, 5-datasets, and split ImageNet-R datasets following DualPrompt [53]. Then we reproduce state-of-the-art prototype-based methods as well as Transformer-based methods and compare CPP to them on split ImageNet-subset and split-CIFAR-100. Note that *all methods reported in this section are implemented using the same pre-trained ViT-B/16.*

Main results on split CIFAR-100, 5-datasets, and split ImageNet-R. We compare CPP to regularization-based methods (*EWC* [29] and *LwF* [28]), advanced rehearsal-based methods (*ER* [8], *GDumb* [38], *BiC* [55], *DER++* [4], and *Co^{2}L* [6]), and Transformer-based methods (*L2P* [54], *ESN* [52], and *DualPrompt* [53]). As shown in Table 1, despite the rehearsal-free property of regularization-based methods, their results lack vigor. Rehearsal-based methods, on the other hand, produce decent results under a large memory budget. Yet, they are still outperformed by prompt-based methods, which do not require rehearsal. Among the family of new emerging Transformer-based methods, CPP surpasses alternatives by a significant margin, showcasing the superiority of our framework which leverages task-specific prompts that are optimized by the contrastive prototypical loss to steer prototypes.

Comparison to prototype-based methods. We further compare CPP against state-of-the-art prototype-based methods including iCaRL [40] and PASS [62] on split ImageNet-subset and split CIFAR-100. For calibration purpose, we also reproduce DualPrompt [53] on split ImageNet-subset. *To avoid information leakage, we adopt pretrained model using self-supervised MAE method for split ImageNet-subset.* Details on reproduction are consigned to the supplementary. As shown in Table 2, a pre-trained ViT backbone can significantly boost the performance of existing prototype-based methods, which aligns with the observation in [39]. However, CPP still demonstrates an unparalleled performance with the same backbone, suggesting a clear advantage of CPP over alternatives.

4.4. Ablation Study

Effectiveness of the proposed modules. Recognizing that the choice of embedding function is instrumental to our approach, it is critical to analyze CPP under different embedding functions. In pursuit of this, CPP is instantiated employing state-of-the-art pre-training mechanisms encompassing *ViT* [13], *Deit* [47], *Dino* [5], and *MAE* [18]—a spectrum that spans supervised to self/un-supervised methodologies and discriminative to generative paradigms. As Table 3 elucidates, both task-specific prompts and multi-centroid prototypes remain robust across all pre-training methods, bringing around 10% to 20% absolute improvements over the baseline model. In addition, these modules, when isolated, retain efficacy and exhibit synergistic performance when combined together. An intriguing observation pertains to the variations induced by distinct pre-training paradigms, highlighting a pronounced correlation between baseline models and their CPP counterparts (*e.g.*, $\rho = 1.0$ for split CIFAR-100).

Contrastive prototypical loss outperforms alternatives. To discern the merits of CPL, we first compare it with two widely-used loss functions: *CE* (cross-entropy) and *SupCon* (supervised contrastive loss) [21]. As shown in Table 4, CPL outperforms both of them by a clear margin. Among

Method	Buffer size	Split ImageNet-subset			Split CIFAR-100		
		Backbone	Pretrain	Avg. Acc (↑)	Backbone	Pretrain	Avg. Acc (↑)
Upper-bound	-	ViT/B-16	ImageNet	94.22$\pm$0.18	ViT/B-16	MAE	93.15$\pm$0.09
iCaRL [40]	2000	ResNet-18	✗	23.77$\pm$0.35	ResNet-18	✗	51.12$\pm$0.36
PASS [62]	0	ResNet-18	✗	27.16$\pm$0.24	ResNet-18	✗	36.32$\pm$0.33
iCaRL [40]	2000	ViT-B/16	MAE	87.96$\pm$0.26	ViT-B/16	ImageNet	75.10$\pm$0.26
PASS [62]	0	ViT-B/16	MAE	72.72$\pm$0.31	ViT-B/16	ImageNet	64.10$\pm$0.20
DualPrompt [53]	0	ViT-B/16	MAE	92.50$\pm$0.24	ViT-B/16	ImageNet	86.51$\pm$0.33
CPP (ours)	0	ViT-B/16	MAE	**93.82**$\pm$**0.06**	ViT-B/16	ImageNet	**91.12**$\pm$**0.12**

Table 2. Comparison to prototype-based methods and DualPrompt on split ImageNet-subset and split CIFAR-100. Results of DualPrompt on split CIFAR-100 are reported from the original paper. All other results are reproduced using the same pre-trained ViT-B/16.

Pretrain	CPP	Multi-centroids	Split CIFAR-100		5-datasets		Split ImageNet-R	
			Avg. Acc (↑)	Forgetting (↓)	Avg. Acc (↑)	Forgetting (↓)	Avg. Acc (↑)	Forgetting (↓)
Deit [47]			71.9	9.97	70.54	0.28	51.29	5.84
	✓		77.64$\pm$0.18	8.06$\pm$0.14	81.67$\pm$0.13	1.71$\pm$0.21	57.85$\pm$0.07	6.74$\pm$0.10
		✓	74.23$\pm$0.08	9.11$\pm$0.14	72.36$\pm$0.02	**0.11**$\pm$**0.01**	49.50$\pm$0.16	6.16$\pm$0.19
	✓	✓	**82.24**$\pm$**0.20**	**6.05**$\pm$**0.19**	**90.94**$\pm$**0.23**	0.22$\pm$0.04	**67.45**$\pm$**0.25**	**4.94**$\pm$**0.12**
Dino [5]			76.69	8.91	72.18	0.68	45.59	7.77
	✓		80.11$\pm$0.22	6.88$\pm$0.20	81.56$\pm$0.02	0.74$\pm$0.06	51.88$\pm$0.10	8.67$\pm$0.15
		✓	79.71$\pm$0.08	7.68$\pm$0.07	74.31$\pm$0.04	**0.18**$\pm$**0.01**	48.63$\pm$0.18	5.72$\pm$0.15
	✓	✓	**83.59**$\pm$**0.12**	**5.43**$\pm$**0.18**	**89.10**$\pm$**0.10**	0.21$\pm$0.08	**61.22**$\pm$**0.59**	**5.04**$\pm$**0.08**
MAE [18]			74.65	8.60	72.77	0.30	55.25	6.21
	✓		78.90$\pm$0.32	8.42$\pm$0.24	82.48$\pm$0.06	0.41$\pm$0.05	62.78$\pm$0.09	5.88$\pm$0.19
		✓	76.68$\pm$0.15	8.16$\pm$0.09	73.98$\pm$0.06	0.11$\pm$0.04	54.68$\pm$0.09	5.22$\pm$0.20
	✓	✓	**83.36**$\pm$**0.18**	**6.50**$\pm$**0.38**	**91.76**$\pm$**0.02**	**0.11**$\pm$**0.01**	**71.04**$\pm$**0.30**	**4.23**$\pm$**0.07**
ViT [13]			82.82	5.94	69.71	0.30	60.20	5.26
	✓		87.18$\pm$0.22	4.70$\pm$0.33	81.54$\pm$0.05	0.42$\pm$0.05	65.88$\pm$0.14	6.76$\pm$0.02
		✓	84.09$\pm$0.05	5.47$\pm$0.12	72.1$\pm$0.17	**0.15**$\pm$**0.01**	61.11$\pm$0.23	4.84$\pm$0.04
	✓	✓	**91.12**$\pm$**0.12**	**3.33**$\pm$**0.18**	**92.92**$\pm$**0.17**	0.19$\pm$0.07	**74.88**$\pm$**0.07**	**3.65**$\pm$**0.03**

Table 3. Ablation studies of the proposed modules under four different pre-training methods. When neither CPP nor multi-centroid prototype are applied, the model is reduced to the training-free baseline model as described in Sec. 3.2.

comparisons, *SupCon* is most compatible with ours, further confirming the advantage of coupling the contrastive loss design with a NCM classifier. Then, we independently add uniformity (w/ uniform) or remove prototypes (w/o proto), under varied temperatures to validate the efficacy of each proposed component. As shown in Fig. 4 (right), encouraging uniformity decreases the performance, and removing prototypes aggravates the prototype interference, which are coherent with our analysis in Sec. 3.3.2.

Stability and plasticity dilemma is mediated by temperature coefficient. Fig. 4 (right) reveals the interplay between temperatures and model performance. Restrictive temperatures, which add greater penalties on negative pairs, reduce forgetting at the cost of damaging accuracy. Higher temperatures, in opposite, improve the overall performance but escalating forgetting. We empirically found $\tau = 0.6$ makes a good trade-off between stability and plasticity and use it by default.

MLP is non-negligible. We show in Table 4 that non-linearity introduced by the MLP is crucial to the success training of prompts, irrespective of the underlying loss mechanism. Replacing the MLP neck by a single linear layer consistently leads to suboptimal outcomes.

Method	Split CIFAR-100	
	Avg. Acc (↑)	Forgetting (↓)
CE (w/ linear neck)	85.48$\pm$0.49	8.42$\pm$0.64
SupCon (w/ linear neck)	87.94$\pm$0.10	3.99$\pm$0.10
CPL (w/ linear neck)	87.20$\pm$0.12	4.08$\pm$0.08
CE [15]	90.42$\pm$0.04	4.45$\pm$0.17
SupCon [21]	90.63$\pm$0.27	3.35$\pm$0.29
CPL (ours)	**91.12**$\pm$**0.12**	**3.33**$\pm$**0.18**

Table 4. CPL vs. alternative loss functions.

Dissection of prompt architecture. The architecture, comprising prompt length and depth, plays a pivotal role in dictating the efficacy of the task-specific prompt. As shown in Fig. 4 (middle right), deep prompts consistently outperforms their shallow counterparts, manifesting the importance of steering features at different levels of abstraction. Intriguingly, a configuration with $L_p = 1$ when paired with deep prompts emerges as adequate for achieving commendable outcomes on split CIFAR-100, and excessive longer prompts can cause over-fitting.

Centroid number vs. query radius. The interplay between centroid number C and query radius r can impact the overall performance. To identify an appropriate configuration, we

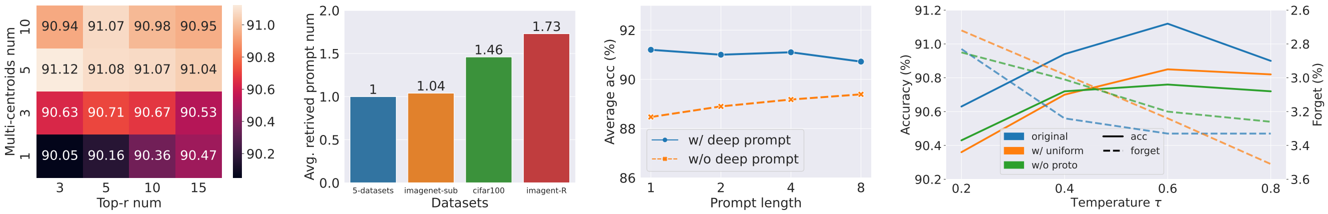

Figure 4. **Left:** Centroid number vs. query neighbors. **Middle left:** Our average number of retrieved prompts on different datasets. **Middle right:** Ablation studies on prompt length and deep prompt. **Right:** Results of CPL and its variants under different temperature coefficients.

Method	Split CIFAR-100	
	Extra Mem. (MB / task) (↓)	Avg. Acc (↑)
DER++ [4]	71.780	83.94±0.34
L2P [54]	0.194	83.86±0.28
DualPrompt [53]	0.190	86.51±0.33
CPP (ours)	**0.035**	**91.12±0.12**

Table 5. Comparison on additional memory usage per task.

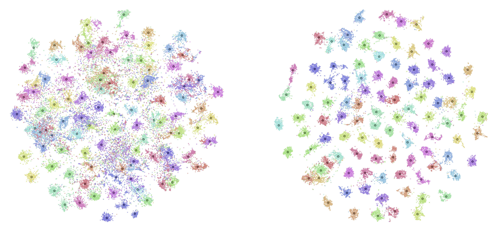

Figure 5. t-SNE plots for data samples without (left) and with (right) task-specific prompts on CIFAR-100.

conduct a simple grid search on split CIFAR-100 and found the setting $C = 5$ with $r = 3$ works fairly well across benchmarks. As Fig. 4 (left) illustrates, the proposed multi-centroid strategy can better characterize the distribution of a given class distribution and thus effectively reduce the query radius and improve the performance.

Visual effect of the task-specific prompts. We visualize data instances alongside their corresponding prototypes from CIFAR-100, both in the presence and absence of task-specific prompts, in Fig. 5. Prior to the infusion of task-specific prompts, intra-class instances tend to cluster in the latent space, Nonetheless, inter-class samples exhibit intertwined behavior. By introducing task-specific prompts, intra-class instances coalesce tightly, while disparate class clusters diverge markedly. We refer to supplementary for more visualization results and analysis.

4.5. Analysis of Efficiency

Here, we delve into the efficiency of CPP, examining memory consumption and computational demands. Comprehensive discussions are provided in the supplementary.

Memory footprint. For an in-depth understanding of memory efficiency, we contrast CPP against contemporaneous methods in the context of *incremental memory consumption per task*. This metric underscores the growth in memory requirement with each additional task. Referencing Table 5, CPP clearly distinguishes itself, registering a significantly minimized memory increment. This trait augments the scalability of CPP to handle long task sequences.

Computational efficiency. Thanks to the design of task-specific prompt and prototypes, the computational burden during training is modest: only tiny portion of parameters are updated through back-propagation and no explicit exemplars need to be forwarded. At inference, CPP may induce multiple forward propagations. This warrants an exploration of its computational feasibility. As observed in Fig. 4 (middle left), large inter-task divergences typically lead to fewer retrieved prompts (*e.g.*, only one for 5-datasets), and thus have the similar inference cost as a linear classifier. As the intra-class divergence increases or inter-class divergence reduces, the retrieved prompt count amplifies. Yet, even on the challenging ImageNet-R dataset, CPP needs, on average, merely an additional 0.73 forward passes, which is affordable under most scenarios.

5. Conclusion

We introduced an innovative, rehearsal-free framework for continual learning, capitalizing on task-specific prompts optimized through a tailored contrastive prototypical loss. This approach adeptly circumvents semantic drift and curtails prototype interference. The multi-centroid prototype strategy augments the expressiveness of prototypes. In evaluations, CPP outperforms established methodologies by a large margin, with an in-depth assessment underscoring the salience of each component. We posit that CPP provides valuable insights into constructing scalable continual learning systems, informed by the recent advances in architectural design and representation learning. Future avenues include enhancing inference efficiency and broadening CPP to encompass complex scenarios such as blurred task boundaries, few-shot, and open-vocabulary configurations. Another promising direction involves devising automatic embedding function selection strategy so as to accommodate diverse downstream contexts.

Acknowledgements. This research is partially funded by grants to Metaxas from NSF: 2310966, 2235405, 2212301, 2003874, 1951890 and NIH 2R01HL127661.

References

[1] Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. Memory aware synapses: Learning what (not) to forget. In Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss, editors, *Computer Vision – ECCV 2018*, pages 144–161, Cham, 2018. Springer International Publishing. 2

[2] Ari S. Benjamin, David Rolnick, and Konrad P. Körding. Measuring and regularizing networks in function space. *CoRR*, abs/1805.08289, 2018. 2

[3] Yaroslav Bulatov. Notmnist dataset. *Google (Books/OCR), Tech. Rep.[Online]. Available: http://yaroslavvb. blogspot. it/2011/09/notmnist-dataset. html*, 2, 2011. 5

[4] Pietro Buzzega, Matteo Boschini, Angelo Porrello, Davide Abati, and SIMONE CALDERARA. Dark experience for general continual learning: a strong, simple baseline. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 15920–15930. Curran Associates, Inc., 2020. 2, 6, 8

[5] Mathilde Caron, Hugo Touvron, Ishan Misra, Herv'e J'egou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9630–9640, 2021. 6, 7

[6] H. Cha, J. Lee, and J. Shin. Co2l: Contrastive continual learning. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9496–9505, 2021. 2, 6

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

[8] Arslan Chaudhry, Marcus Rohrbach, Mohamed Elhoseiny, Thalaiyasingam Ajanthan, Puneet K Dokania, Philip HS Torr, and Marc'Aurelio Ranzato. On tiny episodic memories in continual learning. *arXiv preprint arXiv:1902.10486*, 2019. 6

[9] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20. JMLR.org, 2020. 4

[10] MohammadReza Davari, Nader Asadi, Sudhir Mudur, Rahaf Aljundi, and Eugene Belilovsky. Probing representation forgetting in supervised and unsupervised continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16712–16721, June 2022. 2

[11] Matthias De Lange and Tinne Tuytelaars. Continual prototype evolution: Learning online from non-stationary data streams. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 8250–8259, October 2021. 1, 2

[12] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 5

[13] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021. 3, 6, 7

[14] Mehrdad Farajtabar, Navid Azizan, Alex Mott, and Ang Li. Orthogonal gradient descent for continual learning. In Silvia Chiappa and Roberto Calandra, editors, *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pages 3762–3773. PMLR, 26–28 Aug 2020. 2

[15] I. J. Good. Rational decisions. *Journal of the Royal Statistical Society. Series B (Methodological)*, 14(1):107–114, 1952. 7

[16] Yanan Gu, Cheng Deng, and Kun Wei. Class-incremental instance segmentation via multi-teacher networks. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35:1478–1486, May 2021. 1

[17] Raia Hadsell, Dushyant Rao, Andrei Rusu, and Razvan Pascanu. Embracing change: Continual learning in deep neural networks. *Trends in Cognitive Sciences*, 24:1028–1040, 12 2020. 1, 2

[18] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16000–16009, June 2022. 6, 7

[19] D. Hendrycks, S. Basart, N. Mu, S. Kadavath, F. Wang, E. Dorundo, R. Desai, T. Zhu, S. Parajuli, M. Guo, D. Song, J. Steinhardt, and J. Gilmer. The many faces of robustness: A critical analysis of out-of-distribution generalization. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021. 5

[20] Menglin Jia, Luming Tang, Bor-Chun Chen, Claire Cardie, Serge Belongie, Bharath Hariharan, and Ser-Nam Lim. Visual prompt tuning. In *European Conference on Computer Vision (ECCV)*, 2022. 2, 3

[21] Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. Supervised contrastive learning. In *Advances in Neural Information Processing Systems*, volume 33, pages 18661–18673. Curran Associates, Inc., 2020. 4, 6, 7

[22] Alex Krizhevsky. Learning multiple layers of features from tiny images, 2009. 5

[23] Ananya Kumar, Aditi Raghunathan, Robbie Matthew Jones, Tengyu Ma, and Percy Liang. Fine-tuning can distort pretrained features and underperform out-of-distribution. In *International Conference on Learning Representations*, 2022. 3

[24] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 1998. 5

[25] Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, Nov. 2021. 2, 3

[26] Xilai Li, Yingbo Zhou, Tianfu Wu, Richard Socher, and Caiming Xiong. Learn to grow: A continual structure learning framework for overcoming catastrophic forgetting. In *ICML*, 2019. 2

[27] Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4582–4597. Association for Computational Linguistics, aug 2021. 3

[28] Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(12):2935–2947, 2018. 2, 6

[29] David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. 2, 6

[30] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. 5

[31] Arun Mallya and Svetlana Lazebnik. Packnet: Adding multiple tasks to a single network by iterative pruning. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7765–7773, 2018. 2

[32] Michael Mccloskey and Neil J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. *The Psychology of Learning and Motivation*, 24:104–169, 1989. 1

[33] Thomas Mensink, Jakob J. Verbeek, Florent Perronnin, and Gabriela Csurka. Distance-based image classification: Generalizing to new classes at near-zero cost. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35:2624–2637, 2013. 1, 3

[34] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*, 2011. 5

[35] Andrew Y. Ng, Michael I. Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. In *ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS*, pages 849–856. MIT Press, 2001. 5

[36] Pingbo Pan, Siddharth Swaroop, Alexander Immer, Runa Eschenhagen, Richard Turner, and Mohammad Emtiyaz E Khan. Continual deep learning by functional regularisation of memorable past. In *Advances in Neural Information Processing Systems*, volume 33, 2020. 2

[37] German I. Parisi, Ronald Kemker, Jose L. Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *Neural Networks*, 113:54–71, 2019. 2

[38] Ameya Prabhu, Philip Torr, and Puneet Dokania. Gdumb: A simple approach that questions our progress in continual learning. In *The European Conference on Computer Vision (ECCV)*, August 2020. 2, 6

[39] Vinay Venkatesh Ramasesh, Aitor Lewkowycz, and Ethan Dyer. Effect of scale on catastrophic forgetting in neural networks. In *International Conference on Learning Representations*, 2022. 6

[40] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, G. Sperl, and Christoph H. Lampert. icarl: Incremental classifier and representation learning. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5533–5542, 2017. 1, 2, 3, 6, 7

[41] Andrei A. Rusu, Neil C. Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *CoRR*, abs/1606.04671, 2016. 2

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

[43] Joan Serra, Didac Suris, Marius Miron, and Alexandros Karatzoglou. Overcoming catastrophic forgetting with hard attention to the task. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 4548–4557. PMLR, 10–15 Jul 2018. 2

[44] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. 1, 5

[45] S. Thrun. A lifelong learning perspective for mobile robot control. In V. Graefe, editor, *Intelligent Robots and Systems*. Elsevier, 1995. 1

[46] Michalis K Titsias, Jonathan Schwarz, AG de G Matthews, Razvan Pascanu, and Yee Whye Teh. Functional regularisation for continual learning using gaussian processes. *ICLR 2020*, 2020. 2

[47] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Herve Jegou. Training data-efficient image transformers & distillation through attention. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 10347–10357. PMLR, 18–24 Jul 2021. 6, 7

[48] Gido van de Ven, Tinne Tuytelaars, and Andreas Tolias. Three types of incremental learning. *Nature Machine Intelligence*, 4:1–13, 12 2022. 1, 2

[49] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. 2

[50] Tongzhou Wang and Phillip Isola. Understanding contrastive representation learning through alignment and uniformity on the hypersphere. In *Proceedings of the 37th International Conference on Machine Learning*, pages 9929–9939, 2020. 4

[51] Yabin Wang, Zhiwu Huang, and Xiaopeng Hong. S-prompts learning with pre-trained transformers: An occam's razor for domain incremental learning, 2022. 3

[52] Yabin Wang, Zhiheng Ma, Zhiwu Huang, Yaowei Wang, Zhou Su, and Xiaopeng Hong. Isolation and impartial aggregation: A paradigm of incremental learning without interference. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2023. 6

[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. *European Conference on Computer Vision*, 2022. 3, 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. 3, 6, 8

[55] Yue Wu, Yinpeng Chen, Lijuan Wang, Yuancheng Ye, Zicheng Liu, Yandong Guo, and Yun Fu. Large scale incremental learning. In *CVPR*, pages 374–382, 2019. 6

[56] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017. 5

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

[58] Lu Yu, Bartlomiej Twardowski, Xialei Liu, Luis Herranz, Kai Wang, Yongmei Cheng, Shangling Jui, and Joost van de Weijer. Semantic drift compensation for class-incremental learning. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6980–6989, 2020. 1, 2, 5

[59] Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 3987–3995. PMLR, 06–11 Aug 2017. 2

[60] Chi Zhang, Nan Song, Guosheng Lin, Yun Zheng, Pan Pan, and Yinghui Xu. Few-shot incremental learning with continually evolved classifiers. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2021. 2

[61] Da-Wei Zhou, Fu-Yun Wang, Han-Jia Ye, Liang Ma, Shiliang Pu, and De-Chuan Zhan. Forward compatible few-shot class-incremental learning, 2022. 5

[62] Fei Zhu, Xu-Yao Zhang, Chuang Wang, Fei Yin, and Cheng-Lin Liu. Prototype augmentation and self-supervision for incremental learning. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 1, 2, 4, 5, 6, 7