CIP-Net: Continual Interpretable Prototype-based Network

Federico Di Valerio[1], **Michela Proietti**[1], **Alessio Ragno**[2,3], **Roberto Capobianco**[4]
[1]Department of Computer, Control and Management Engineering (DIAG), Sapienza University, IT-00185, Rome, Italy
[2]INSA Lyon, CNRS, LIRIS UMR 5205, FR-94276, Villeurbanne France
[3]EPITA Research Laboratory (LRE), FR-69621, Le Kremlin-Bicêtre, France
[4]Sony AI, CH-8952, Zurich, Switzerland
{divalerio, mproietti}@diag.uniroma1.it, alessio.ragno@{insa-lyon.fr, epita.fr}, roberto.capobianco@sony.com

Abstract

Continual learning constrains models to learn new tasks over time without forgetting what they have already learned. A key challenge in this setting is catastrophic forgetting, where learning new information causes the model to lose its performance on previous tasks. Recently, explainable AI has been proposed as a promising way to better understand and reduce forgetting. In particular, self-explainable models are useful because they generate explanations during prediction, which can help preserve knowledge. However, most existing explainable approaches use post-hoc explanations or require additional memory for each new task, resulting in limited scalability. In this work, we introduce CIP-Net, an exemplar-free self-explainable prototype-based model designed for continual learning. CIP-Net avoids storing past examples and maintains a simple architecture, while still providing useful explanations and strong performance. We demonstrate that CIP-Net achieves state-of-the-art performances compared to previous exemplar-free and self-explainable methods in both task- and class-incremental settings, while bearing significantly lower memory-related overhead. This makes it a practical and interpretable solution for continual learning.

Code & Supplementary Material — https://github.com/KRLGroup/CIP-Net

Introduction

Continual learning (CL) research aims at developing models that can learn tasks sequentially, without revisiting past data. In this setting, the major obstacle is *catastrophic forgetting* (French 1999; Goodfellow et al. 2013; Kirkpatrick et al. 2017): learning new tasks overwrites or distorts earlier knowledge, sharply degrading performance on past tasks. Explainable artificial intelligence (XAI) helps with understanding this phenomenon by analyzing why and how certain knowledge is retained or lost across tasks. In this line of work, Cossu et al. (2023) leverage XAI to study explanation drift, i.e., a shift in the explanations of past tasks as new knowledge is acquired. Given that these representational modifications are usually a direct cause of catastrophic forgetting, several works directly exploit explanations to develop XAI-guided approaches that prevent explanation drift and, in turn, mitigate forgetting. Most methods use post-hoc explainability (Ede et al. 2022; Ebrahimi et al. 2021; Saha and Roy 2023; Bai et al. 2023) (i.e., explanations are produced after each task's training, looking at the model's output and parameters), whereas recent work introduces self-interpretable CL models (Rymarczyk et al. 2023; Proietti et al. 2025). Among them, ICICLE (Rymarczyk et al. 2023) builds on a popular prototype-based architecture, namely ProtoPNet (Chen et al. 2019), to tackle catastrophic forgetting in continual image classification. Prototype-based networks directly build interpretability into the model, generating explanations during prediction (Rudin 2019). Specifically, these models link each prediction to a handful of visually meaningful image patches, thus enabling intuitive *this-looks-like-that* reasoning that is self-generated rather than post-produced (Chen et al. 2019). In CL, this explicit evidence makes drift easier to detect and track. However, ICICLE comes with one major drawback: it adds a new prototype head for each task, where a prototype head is a specific module that enables prototype-based reasoning in the model. As a consequence, computation and memory grow with the number of tasks, limiting ICICLE's scalability, and separate heads limit cross-task sharing.

To overcome these issues, we propose a novel Continual Interpretable Prototype-based Network (CIP-Net), built on PIP-Net's (Nauta et al. 2023) lighter prototype-based reasoning. A single fixed-size pool of shared patch-level prototypes yields robust accuracy and stable explanations, eliminating the need for exemplar storage or multiple task-specific prototype layers. To prevent interference in the shared prototype layer, we adopt alignment–uniformity self-supervision and targeted regularization mechanisms. On CUB-200-2011 (CUB, Wah et al. (2011)) and Stanford Cars (CARS, Krause et al. (2013)) benchmarks, CIP-Net outperforms ICICLE and other exemplar-free standard CL approaches in both *task-incremental learning* (TIL, task identity known at test time) and *class-incremental learning* (CIL, task identity not known at test time). Our contributions are:

- CIP-Net: an exemplar-free, self-explainable, prototype-based CL model with a shared prototype layer, that enables knowledge sharing and avoids the memory-related overhead associated with task-specific prototype heads;
- Introduction of targeted prototype regularization, discouraging changes in important prototypes for past tasks, and loss terms to encourage the use of sparse prototypes

and promote orthogonality between classification heads;

- CIP-Net mitigates catastrophic forgetting, outperforming ICICLE in both TIL and CIL scenarios on CUB and CARS datasets, with gains up to $+35\%$ (TIL) and $+11.9\%$ (CIL) in final average accuracy on CUB and up to $+34.2\%$ (TIL) and $+38.2\%$ (CIL) on CARS;
- Explanation drift analysis, which reveals that explanations remain stable during training;
- An open-source implementation of CIP-Net

Related Work

CL methods can be broadly framed within a five-branch taxonomy (Wang et al. 2024). Regularization-based approaches introduce additional penalties to discourage changes in weights or activations. For instance, EWC (Kirkpatrick et al. 2017) estimates parameter importance via the Fisher information matrix, penalizing changes to high-importance weights. In contrast, LwF (Li and Hoiem 2017) distills the previous model's outputs on new-task data to preserve old decision boundaries. Differently, replay-based strategies approximate past data by storing a small exemplar buffer (Rebuffi et al. 2017; Hou et al. 2019) or generating pseudo-samples (Shin et al. 2017), which are then replayed alongside the current task's data. Representation-based methods are typically exemplar-free alternatives, producing robust representations through self-supervised learning, large-scale pre-training, or representation learning. Examples belonging to this category are FeTrIL (Petit et al. 2023), which learns a feature-translation module to align new representations with a frozen old classifier, and PASS (Zhu et al. 2021), which combines prototype augmentation with self-supervision to maintain class discrimination without storing images. Optimization-based approaches, on the other hand, directly act on the optimization process, by projecting gradients in non-interfering directions (Farajtabar et al. 2020), or using meta-learning (Rajasegaran et al. 2020). Finally, architecture-based solutions either allocate task-specific parameters through masking (Mallya, Davis, and Lazebnik 2018), pruning-and-packing (Mallya and Lazebnik 2018), or dynamic expansion (Yan, Xie, and He 2021), to reduce task interference. Within these families, some methods effectively leverage XAI. For example, LwM (Dhar et al. 2019) extends the idea behind LwF by distilling attention/activation maps instead of (or in addition to) logits, enabling exemplar-free knowledge transfer. Others employ explanations to select the examples to store in the replay buffer (Saha and Roy 2023; Shim et al. 2021; Bai et al. 2023). Ede et al. (2022), instead, propose to create task-specific network components. In these contexts, XAI allows for the identification of a phenomenon known as explanation drift, which consists of a change in the explanations of old tasks after training on subsequent ones (Cossu et al. 2024). By attenuating explanation drift, XAI-guided approaches prove to be effective in mitigating catastrophic forgetting (Proietti, Ragno, and Capobianco 2025; Ebrahimi et al. 2021).

While most of the existing XAI-guided CL approaches are post-hoc, meaning they provide explanations for pre-trained models, ICICLE (Rymarczyk et al. 2023) builds on ProtoPNet (Chen et al. 2019), a prototype-based self-interpretable architecture. This type of architectures learn a set of prototypes that represent parts of the training input images, and perform the classification based on the similarity of the patches of the current input to the learned prototypes. While this scheme allows providing explanations at prediction time, ICICLE shows that it can also be exploited for suppressing catastrophic forgetting. To adapt this architecture to the CL scenarios, ICICLE features task-specific prototype layers and classification heads. However, this means that the network grows considerably with the number of tasks.

In this work, we take inspiration from this idea and specifically address these issues. We propose CIP-Net, which relies on a different prototype-based network, namely PIP-Net (Nauta et al. 2023). While replicating ICICLE's exemplar-free protocol, we replace multiple prototype heads with a *single and shared* prototype layer. In this way, we manage to keep the memory and computational overhead introduced by prototype-based reasoning constant as the number of tasks grows, while maintaining interpretability and improving accuracy.

Background

In this section, we introduce the mathematical basis necessary to understand our proposed method. We begin by introducing prototype-based self-explainable models, which offer an interpretable alternative to standard deep learning approaches by providing interpretable predictions through comparisons of input features to learned prototypical parts. In particular, we focus on ProtoPNet (Chen et al. 2019) and then PIP-Net (Nauta et al. 2023), which are at the basis of ICICLE and our proposed approach, CIP-Net.

Prototype-based Neural Networks

A prototype-based neural network is an architecture that consists of a feature extractor followed by a prototype layer g_p, which learns a set of P prototypical representations used for classification. Formally, a feature extractor f maps an input $\mathbf{x} \in \mathbb{R}^{H \times W \times C}$ to a latent representation $\mathbf{z} = f(x) \in \mathbb{R}^{H' \times W' \times D}$, and the prototype layer g_p computes class evidence by comparing this latent representation to a set of prototypes using a model-specific similarity function.

In ProtoPNet (Chen et al. 2019), each prototype $\mathbf{p}_j \in \mathbb{R}^{H_p \times W_p \times D}$ is a learnable parameter tensor representing a prototypical latent patch. Similarity is measured using the negative squared Euclidean distance:

$$\text{sim}(\mathbf{z}_i, \mathbf{p}_j) = -\|\mathbf{z}_i - \mathbf{p}_j\|_2^2, \tag{1}$$

where $\mathbf{z}_i$ is a patch from the latent representation. For each prototype, the maximum similarity across all patches is used. The final class logit for class k is computed as:

$$\mathbf{s}_k = \sum_{j=1}^{P} w_{j,k} \cdot \max_i \text{sim}(\mathbf{z}_i, \mathbf{p}_j), \tag{2}$$

where $w_{j,k}$ is the learned importance weight of prototype $\mathbf{p}_j$ for class k. Additionally, during training, each prototype is

projected onto the closest latent patch in the training set to ground it in a real image. The training loss combines cross-entropy and interpretability terms:

$$\mathcal{L}_{\text{ProtoPNet}} = \mathcal{L}_{\text{CE}} + \lambda_{\text{clst}} \cdot \mathcal{L}_{\text{clst}} + \lambda_{\text{sep}} \cdot \mathcal{L}_{\text{sep}}, \quad (3)$$

where $\mathcal{L}_{\text{clst}}$ pulls latent patches closer to their class prototypes, and $\mathcal{L}_{\text{sep}}$ pushes them away from other-class prototypes.

In PIP-Net (Nauta et al. 2023), prototypes are not learned directly but they are simply one channel of the final feature map. The presence of a prototype in the latent representation is determined through a softmax operation across the channel dimension. Given a latent feature map $\mathbf{z} \in \mathbb{R}^{H' \times W' \times D}$, a softmax is applied at each spatial location:

$$z'_{h,w,d} = \frac{\exp(z_{h,w,d})}{\sum_{d'=1}^{D} \exp(z_{h,w,d'})}, \quad (4)$$

and the presence score of prototype d is obtained via spatial max-pooling:

$$p_d = \max_{h,w} z'_{h,w,d}. \quad (5)$$

Classification scores are computed using a sparse, non-negative linear layer:

$$s_k = \sum_{d=1}^{D} w_{d,k} \cdot p_d, \quad \text{with } w_{d,k} \geq 0. \quad (6)$$

Due to the lack of specific prototype representations and the impossibility of performing the prototype projection, in order to ensure semantic consistency, PIP-Net employs a contrastive learning objective during pretraining. The full loss is:

$$\mathcal{L}_{\text{PIP-Net}} = \lambda_C \mathcal{L}_{\text{CE}} + \lambda_A \mathcal{L}_{\text{A}} + \lambda_T \mathcal{L}_{\text{T}}, \quad (7)$$

where $\mathcal{L}_{\text{A}}$ aligns prototype activations between augmented views of the same image:

$$\mathcal{L}_{\text{A}} = -\frac{1}{HW} \sum_{h,w} \log \left(\mathbf{z}'_{h,w,:} \cdot \mathbf{z}''_{h,w,:} \right), \quad (8)$$

and $\mathcal{L}_{\text{T}}$ encourages uniform prototype usage:

$$\mathcal{L}_{\text{T}} = -\frac{1}{D} \sum_{d=1}^{D} \log \left(\tanh \left(\sum_{b=1}^{B} \mathbf{p}^{(b)} \right) + \epsilon \right). \quad (9)$$

Here, $\mathbf{z}'$ and $\mathbf{z}''$ are feature maps from two augmentations of the same image, and $\mathbf{p}^{(b)}$ is the prototype presence score vector in minibatch b. This design allows PIP-Net to produce interpretable predictions using shared, semantically meaningful prototypes.

ICICLE

ICICLE (Rymarczyk et al. 2023) extends prototype–based reasoning to exemplar-free CL. Assume a stream of T tasks $\left(C^1, X^1, Y^1, \right), \ldots, \left(C^T, X^T, Y^T\right)$, where C^t is the set of classes introduced at task t, X^t and Y^t the associated samples and respective labels. During task t, only X^t, Y^t are accessible, and no data from earlier tasks is stored. Clearly, $C^i \cap C^j = \emptyset \quad \text{for all } i \neq j$.

While ICICLE is built on top of ProtoPNet, the authors propose to use a dedicated prototype layer g^t and classification head h^t for every task. Each prototype layer g^t contains $M^t = K \cdot |C^t|$ prototypes, with K being the number of prototypical parts per class.

During training, $\mathcal{L}_{CE}$ is calculated on the concatenation of all task logits. On the other hand, $\mathcal{L}_{clst}$ and $\mathcal{L}_{sep}$ are left unaltered and computed within the g^t head. Inspired by Keswani et al. (2022), a regularization term prevents explanation shift by minimizing the difference between prototype similarities at tasks t and $t-1$:

$$\mathcal{L}_{\text{IR}} = \sum_{i=1}^{H} \sum_{j=1}^{W} \left| \text{sim}\left(p^{t-1}, z^t_{i,j}\right) - \text{sim}\left(p^t, z^t_{i,j}\right) \right| S_{i,j}, \quad (10)$$

where p^{t-1} is a prototype frozen after task $t-1$, p^t its current counterpart, and S is a binary mask selecting the γ-quantile of the pixels with the highest similarity.

Methods

We propose CIP-Net, a CL model that combines interpretability, efficiency, and robustness to catastrophic forgetting. CIP-Net, illustrated in Figure 1, extends PIP-Net by adapting the shared prototype layer and introducing targeted task-aware regularization to incremental learning scenarios. In this section, we describe the architectural modifications, prototype pretraining, and full training objective.

Architecture CIP-Net builds upon the prototype extraction pipeline of PIP-Net: each input image is processed by a convolutional backbone that produces a latent feature map; prototype presence scores are computed via a softmax over the channel dimension, followed by spatial max-pooling, yielding a vector $\mathbf{p} = [p_1, ..., p_D] \in \mathbb{R}^D$. A known issue in CL arises when logits from newer tasks exhibit higher magnitudes than those from earlier ones. This scale mismatch introduces a bias toward recent tasks, thereby accelerating catastrophic forgetting. To address this, we modify the classification head to normalize both the prototype activation vector $\mathbf{p}$ and the classifier weight vectors $\mathbf{w}_c^{(t)} \in \mathbb{R}^D$ for each class c in task t. The class scores are then computed as:

$$s_k = \tau \cdot \frac{\mathbf{p}^\top \mathbf{w}_c^{(t)}}{\|\mathbf{p}\|_2 \cdot \|\mathbf{w}_c^{(t)}\|_2} \quad (11)$$

where τ is a learnable temperature parameter that compensates for the reduced variance introduced by normalization, ensuring that logits remain expressive and comparable across tasks.

Training Following the structure of PIP-Net, for each task, our training is composed of two distinct phases: a pretraining phase aimed at producing semantically rich and diverse prototypes, and a continual training phase in which task-specific classifiers are learned.

In the first phase, the convolutional backbone and prototype layer are trained jointly using an unsupervised objective

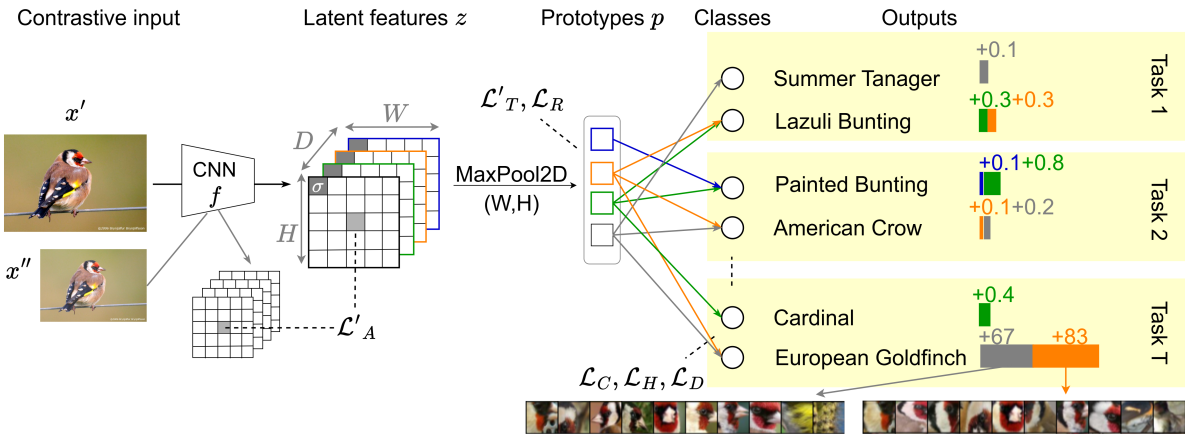


Figure 1: CIP-Net overview. Two augmented views are processed by a CNN to produce a prototype feature map $\mathbf{z}$; channel-wise softmax and maxpooling operation yield prototype-presence scores $\mathbf{p}$. Contrastive alignment loss $\mathcal{L}'_A$ aligns prototypes across views, $\mathcal{L}'_T$ promotes rarely used prototypes, and $\mathcal{L}_R$ limits drift of important ones over tasks. Task-specific sparse, non-negative heads map prototypes to classes, trained with negative log-likelihood loss $\mathcal{L}_C$ plus sparsity $\mathcal{L}_H$ and head-decorrelation $\mathcal{L}_D$.

that does not rely on class labels. The goal here is to shape the prototypes into meaningful and reusable concepts before introducing the task-specific classification. From PIP-Net, we modify slightly $\mathcal{L}_A$ to encourage faster convergence between the two augmented views to their midpoint representation:

$$\mathcal{L}'_{\mathrm{A}} = -\frac{1}{2HW}\sum_{h,w}\log\left((\mathbf{z}'_{h,w,:}\cdot\mathbf{z}^{m}_{h,w,:})(\mathbf{z}''_{h,w,:}\cdot\mathbf{z}^{m}_{h,w,:})\right), \tag{12}$$

where $\mathbf{z}^{m}_{h,w,:} = \frac{\mathbf{z}'_{h,w,:}+\mathbf{z}''_{h,w,:}}{2}$. Given that a trivial solution to $\mathcal{L}'_A$ would be to collapse onto a single active prototype, we adopt the same solution as PIP-Net with a diversity-promoting term $\mathcal{L}'_T$. We modify the diversity loss with a selection mechanism that filters the prototypes on which we want to induce uniform activation. The resulting loss is then:

$$\mathcal{L}'_T = -\frac{1}{|\tilde{\mathcal{I}}|}\sum_{d\in\tilde{\mathcal{I}}}\log\Big(\tanh\big(\sum_{b}^{B}\mathbf{p}_b\big)+\varepsilon\Big), \quad \tilde{\mathcal{I}} = \bigcup_{t=1}^{\hat{t}}\tilde{\mathcal{I}}^t. \tag{13}$$

Here $\hat{t}$ is the current task and $\tilde{\mathcal{I}}$ is the set of rarely activated prototypes. To obtain $\tilde{\mathcal{I}}$, for each head we select the least frequently activated prototypes ($<$ 75th percentile of prototypes with a presence score ≥ 0.5) and we consider them as *rarely-used*. Essentially, $\mathcal{L}'_T$ forces rarely-used prototypes to activate at least once per batch. Additionally, we introduce another regularization term, which is responsible for controlling the explanation drift, that is, the change of prototype activations when learning successive tasks. The resulting prototype-stability regularizer acts separately on every previously learned classification head. In this case, for each task t, only *highly-used* prototypes belonging to $\mathcal{I}^t = P\setminus\tilde{\mathcal{I}}^t$ enter the loss:

$$\mathcal{L}_R = \sum_{t=1}^{t<\hat{t}}\mathcal{L}^t_R = \sum_{t=1}^{t<\hat{t}}\frac{1}{|\mathcal{I}^t|}\sum_{d\in\mathcal{I}^t}\left(\max_c |w^t_{d,c}|\right)\|f^{\hat{t}}_d - f^t_d\|_2, \tag{14}$$

where $w^t_{c,d}$ are the class-prototype weights for the previous task t. In $\mathcal{L}^t_R$, the Euclidean distance between the current prototype feature vector f_d and the prototype features obtained from the frozen model trained on the previous task f^t_d is penalized and weighted by the prototype's maximum outgoing class weight. This term stops the semantics of prototypes that are critical for earlier tasks from drifting too much while leaving rarely activated prototypes free to adapt. As a result, $\mathcal{L}_{\mathcal{R}}$ safeguards past performance and explanations without hindering the network's capacity to learn and update prototypes for the current task. Note that this loss term is only active in tasks following the first one. The total pretraining objective is given by:

$$\mathcal{L}_{\text{pre}} = \lambda_A\mathcal{L}'_A + \lambda_T\mathcal{L}'_T + \lambda_R\mathcal{L}_R, \tag{15}$$

where each term contributes to building a diverse and interpretable prototype set.

Once pretraining is complete, the model enters the incremental learning phase. For each new task $\hat{t}$, only the classification head $\mathbf{w}^{\hat{t}}$ is newly initialized and trained, while the backbone and prototype layer are fine-tuned and earlier heads are frozen. The core of the objective is the standard negative log-likelihood classification loss $\mathcal{L}_C$, which ensures that the model correctly predicts the labels of the current task. However, to maintain interpretability and prevent interference across tasks, we introduce two additional regularizers. The first is a Hoyer sparsity loss (Hoyer 2004):

$$\mathcal{L}_H = \left(1 - \frac{1}{|C^{\hat{t}}|}\sum_{c\in C^{\hat{t}}}\frac{\sqrt{D} - \frac{\|\mathbf{w}^{\hat{t}}_c\|_1}{\|\mathbf{w}^{\hat{t}}_c\|_2+\epsilon}}{\sqrt{D}-1}\right). \tag{16}$$

$\mathcal{L}_H$ encourages each class in the current task to rely on a small number of prototypes. This leads to concise and disentangled explanations while limiting overlapping prototype

usage. The second term is a head decorrelation loss:

$$\mathcal{L}_D = \sum_{t=0}^{\hat{t}} \Big(\|\mathbf{S}^t\| - \|\mathrm{diag}(\mathbf{S}^t)\| \Big), \tag{17}$$

where $\mathbf{S}^t = \left(\mathbf{W}^{(t)}\,\mathbf{W}^{(\hat{t})\top}\right)^2$ is the squared pairwise dot product between the weight vectors of the t-th head and the current head $\hat{t}$. $\mathcal{L}_D$ promotes orthogonality between the current classification head and those from previous tasks. By minimizing the off-diagonal elements of the dot-product matrices between heads, this term prevents new tasks from reusing prototype combinations already assigned to earlier classes, reducing representational interference and forgetting. The full training objective for the current task is therefore:

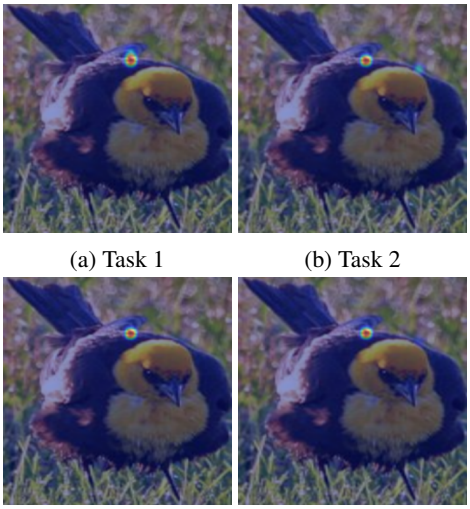
$$\mathcal{L} = \mathcal{L}_{\text{pre}} + \lambda_C \mathcal{L}_C + \lambda_H \mathcal{L}_H + \lambda_D \mathcal{L}_D, \tag{18}$$

where each component plays a distinct role: preserving diversity and stability from the pretraining phase, ensuring accurate task-specific predictions, and enforcing structured, sparse, and disjoint prototype usage over time.

Results

We evaluate CIP-Net on the standard CL benchmarks in the prototypes literature of CUB (Wah et al. 2011) (200 bird species) and CARS (Krause et al. 2013) (196 car models). We split the classes in each dataset into {4, 10, 20} and {4, 7, 14} tasks, respectively. We run each experiment on 3 different seeds, except for the ablation study (1 seed). Further details on compute resources and used hyperparameter values are described in the *Supplementary Material*.

CIP-Net Outperforms Baselines in All Tested Scenarios We compare CIP-Net with ICICLE (Rymarczyk et al. 2023) and the baselines it was compared against, as it is the current state-of-the-art exemplar-free interpretable CL method and, to our knowledge, the closest prototype-based CL approach to our method in the literature. In Table 1, we report the final average accuracy, i.e., the average accuracy computed on all the tasks after training the model on the last one, obtained by the tested methods in the TIL and CIL scenarios on the CUB dataset. Final average accuracies for every task configuration on CARS and the per-task average accuracies for the 4-task setting on CUB (also with ResNet34 backbone) can be found in the *Supplementary Material*. We use these metrics as they are standard in CL literature (Rymarczyk et al. 2023; Wang et al. 2024). CIP-Net consistently outperforms the baselines across all settings, with improvements over ICICLE in final average accuracy from $+11.5\%$ (4 tasks) to $+35.0\%$ (20 tasks) in TIL and from $+11.9\%$ (4 tasks) to $+8.1\%$ (20 tasks) in CIL on CUB. On CARS, we get even higher improvements ranging from $+20.8\%$ (4 tasks) to $+34.2\%$ (14 tasks) in TIL and from $+25.4\%$ (4 tasks) to $+38.2\%$ (7 tasks) and $+27.3\%$ (14 tasks) in CIL (see the *Supplementary Material*). Interestingly, in the TIL scenario, the performance remains high as the number of tasks increases, suggesting that CIP-Net better mitigates cumulative forgetting when the task identity is known. We hypothesize this could be the result of CIP-Net learning more generalizable prototypes, selectively re-adapting important ones, or progressively recognizing a broader set of patterns as new tasks are introduced, while still treating specific prototypes as discriminative for earlier tasks, whose classification heads remain frozen. Some intuition of this can be noticed in Figure 3, where the same prototypes are visualized for each task after training on the last one. In contrast, a more typical trend appears in the CIL scenario. As the number of tasks increases, forgetting also increases, and inferring the correct class from the input alone becomes more challenging for the model. As it commonly happens with CL models in the CIL setting, CIP-Net also exhibits a bias toward the most recent task (shown for the 4-task settings on CUB in the *Supplementary Material*), leading to reduced average accuracy as the task space expands. However, CIP-Net still outperforms the considered baselines in all task settings, highlighting a better retention of past knowledge.



(a) Task 1 (b) Task 2

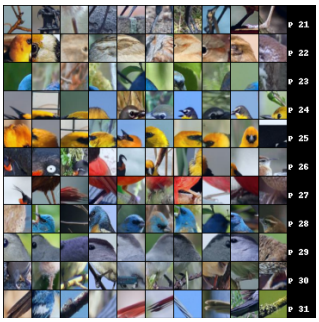
(c) Task 3 (d) Task 3

Figure 2: Example of suppressed prototype activation drift for one of the regularized prototypes from task 1 to task 4 (4-tasks setting) on CUB. More details in Supplementary Material.

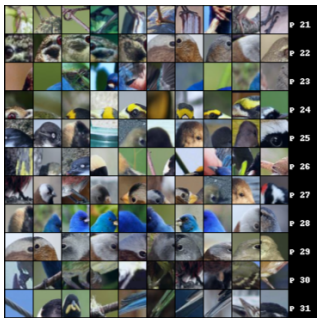
CIP-Net Mitigates Explanation Drift To assess whether our XAI-guided learning truly mitigates explanation drift, we monitor how the activations of prototypes learned in the earliest tasks on specific input images evolve as subsequent tasks are introduced. Figures 2 shows how the activation of one of the regularized prototypes does not change from the first task to the last one (4-task setting). This example offers a qualitative confirmation that CIP-Net's regularization term suppresses modifications to the most frequently used prototypes, thereby minimizing inter-task interference and keeping their activations essentially stable across the entire training process. Examples of other images together with a rare example of a drifted regularized prototype are shown in the *Supplementary Material*. A quantitative evaluation of the suppression of the drift is presented in Figure 4: panel (a) plots the average change in prototype activations computed

Method	Final Avg. Task-Incremental Accuracy			Final Avg. Class-Incremental Accuracy		
	4 Tasks	10 Tasks	20 Tasks	4 Tasks	10 Tasks	20 Tasks
EWC	0.445 ± 0.012	0.288 ± 0.034	0.188 ± 0.031	0.213 ± 0.007	0.095 ± 0.007	0.046 ± 0.011
LWM	0.452 ± 0.023	0.294 ± 0.032	0.226 ± 0.025	0.180 ± 0.011	0.090 ± 0.011	0.044 ± 0.008
LWF	0.301 ± 0.048	0.175 ± 0.028	0.129 ± 0.023	0.219 ± 0.011	0.078 ± 0.008	0.072 ± 0.008
ICICLE	0.654 ± 0.011	0.602 ± 0.035	0.497 ± 0.099	0.350 ± 0.053	0.185 ± 0.005	0.099 ± 0.003
CIP-Net	$\mathbf{0.772 \pm 0.015}$	$\mathbf{0.846 \pm 0.024}$	$\mathbf{0.847 \pm 0.016}$	$\mathbf{0.469 \pm 0.012}$	$\mathbf{0.358 \pm 0.030}$	$\mathbf{0.180 \pm 0.012}$
FeTrIL	0.750 ± 0.008	0.607 ± 0.018	0.407 ± 0.051	0.375 ± 0.006	0.199 ± 0.003	0.127 ± 0.011
PASS	0.775 ± 0.006	0.647 ± 0.006	0.518 ± 0.012	0.395 ± 0.001	0.233 ± 0.009	0.139 ± 0.017
PIP-Net C	0.843 ± 0.002					

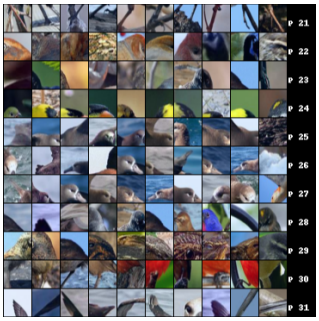
Table 1: Final average (3 seeds) accuracy comparison for different numbers of tasks on CUB for both task- and class-incremental scenarios. This table describes the behavior of CIP-Net with an increasing number of tasks to be learned and compares it to other baselines. CIP-Net outperforms the baseline methods across all task numbers. Additionally, we show the gap between interpretable and black-box models by comparing CIP-Net to FeTrIL (Petit et al. 2023) and PASS (Zhu et al. 2021). At last, the PIP-Net (with ConvNeXt (Liu et al. 2022) backbone) performance upper bound is presented. Results of methods different from CIP-Net are taken from Rymarczyk et al. (2023).



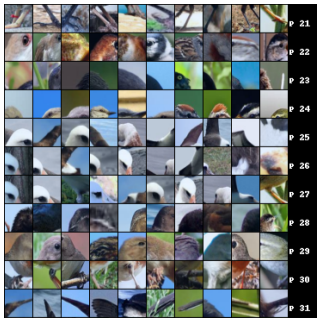
(a) Task 0



(b) Task 1



(c) Task 2

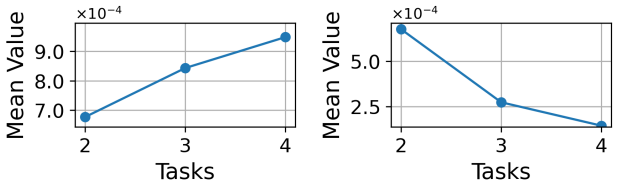


(d) Task 3

Figure 3: Visualization of a portion of prototypes for each task in CUB's 4-task setting), obtained after training on the whole task sequence.

over classes between the current task and the first task, while panel (b) depicts the average change between the current task and the immediately preceding task. The plots clearly show that throughout the tasks, the activations stay mostly stable.

CIP-Net Reduces Overhead As described in the Methods section, CIP-Net integrates a single prototype layer as the final layer of the convolutional backbone. The number of prototypes, a predefined hyperparameter, serves as an upper bound since the model may use only a subset. This is a key



(a) (b)

Figure 4: (a) Average difference of activation values over classes between the current task and the first task. (b) Average difference of activation values over classes between the current task and the immediate previous task.

Model	# Prototypes	Total Parameters (4-tasks)
ICICLE	$\sum_t^T M^t = \sum_t^T K \cdot \lvert C_t \rvert$	$\sum_t^T M^t \cdot \#(\mathbf{W}_{g_t}) \approx 1\text{MLN}$
CIP-Net	$\leq D$	0

Table 2: Comparison of parameters' overhead between CIP-Net and ICICLE used for the prototype layers. Note that $\#(\mathbf{W}_{g_t})$ represents the number of parameters of layer g_t in ICICLE, where they use prototypes of dimension 512. Unlike ICICLE, CIP-Net does not use additional parameters for its prototype-based reasoning. The classification heads are not considered in the prototype layers.

difference from ICICLE, which introduces a separate prototypical part layer g_t for each task, causing parameters to grow linearly with the number of tasks. Table 2 compares the memory overhead from prototype-based reasoning in ICICLE and CIP-Net. Instead of adding $M^t = K \cdot |C_t|$ new prototypes at each incremental step, CIP-Net reuses and incrementally updates a fixed, shared prototype pool, regularized to retain prior knowledge while adapting to new classes. This approach reduces prototype-based reasoning related storage and computation while preserving interpretability without ICICLE's architectural growth. For each task, only a tensor of D values is stored to record frequently used prototypes, enabling the regularization described in the Methods section.

CIP-Net	Task-Incremental Accuracy					Class-Incremental Accuracy				
	Task 1	Task 2	Task 3	Task 4	Final Avg.	Task 1	Task 2	Task 3	Task 4	Final Avg.
w/o $\mathcal{L}_D$	0.72 ± 0.04	0.71 ± 0.13	0.81 ± 0.03	0.85 ± 0.05	0.77 ± 0.01	0.09 ± 0.01	0.29 ± 0.06	0.50 ± 0.01	0.82 ± 0.06	0.43 ± 0.00
w/o $\mathcal{L}_H$	0.85 ± 0.01	0.72 ± 0.14	0.81 ± 0.03	0.84 ± 0.06	0.81 ± 0.03	0.19 ± 0.04	0.45 ± 0.05	0.61 ± 0.04	0.76 ± 0.07	0.50 ± 0.03
w/o $\mathcal{L}_D$&$\mathcal{L}_H$	0.81 ± 0.01	0.74 ± 0.13	0.82 ± 0.02	0.84 ± 0.05	0.80 ± 0.03	0.11 ± 0.03	0.24 ± 0.10	0.49 ± 0.04	0.82 ± 0.05	0.42 ± 0.03
w/o $\mathcal{L}_R$	0.14 ± 0.02	0.35 ± 0.04	0.56 ± 0.04	0.86 ± 0.06	0.48 ± 0.03	0.00 ± 0.00	0.08 ± 0.05	0.17 ± 0.03	0.83 ± 0.07	0.27 ± 0.02
w/o τ	0.83 ± 0.04	0.37 ± 0.04	0.32 ± 0.04	0.31 ± 0.08	0.46 ± 0.05	0.83 ± 0.04	0.02 ± 0.01	0.00 ± 0.00	0.00 ± 0.00	0.21 ± 0.01
CIP-Net	0.83 ± 0.00	0.63 ± 0.15	0.81 ± 0.04	0.82 ± 0.06	0.77 ± 0.01	0.25 ± 0.04	0.42 ± 0.02	0.51 ± 0.04	0.70 ± 0.06	0.47 ± 0.01

Table 3: Model's ablations (3 seeds) comparing accuracies in TIL and CIL 4-tasks scenarios.

Loss Components Functional Importance to CIP-Net's Performance To assess the importance of each loss term in maintaining CIP-Net's performance and stability across tasks, we conduct an ablation study, with results reported in Table 3.

Removing the decorrelation loss $\mathcal{L}_D$ caused a slight TIL drop and a larger CIL decline, disproportionately affecting earlier tasks. Interestingly, omitting the Hoyer term $\mathcal{L}_H$ yielded performance comparable to the full model. Further analysis showed that setting to 0 all near-zero weights below the 50th percentile maintained accuracy, suggesting that classification heads naturally favor sparse weight matrices, enabling more compact explanations. This points to future work on promoting sparsity more efficiently or allowing it to emerge naturally. When both $\mathcal{L}_D$ and $\mathcal{L}_H$ were removed, CIL accuracy declined, again affecting early tasks more. The absence of these regularizers loosens the constraints on the prototypes, which can reduce their intuitiveness and discriminative power and marginally lower overall performance. The prototype regularization term $\mathcal{L}_R$ proved essential: its removal caused one of the two worst performances in both CIL and TIL, with strong bias toward recent tasks and aggravated forgetting. Finally, excluding the temperature parameter τ produced a similar overall accuracy drop to removing $\mathcal{L}_R$, but with an opposite trend of better performance on the earliest task and severely reduced results on the most recent ones.

Overall, these results confirm that each component distinctly contributes to balancing stability, plasticity, and discriminative power in CIP-Net.

Global and Local Explanations Like PIP-Net, CIP-Net also offers two complementary forms of interpretability. Global explanations are provided by the decision layers: the weights link each class to its relevant prototypes, clearly counting only those prototypes whose weights are non-zero, with smaller values indicating less importance.

Local explanations zoom in on a single prediction, pinpointing the prototypes that fire at specific spatial locations. This is valid for both TIL and CIL scenarios. Naturally, the particular subset of prototypes, and thus the visual rationale, may differ between TIL and CIL evaluations. An intuitive illustration of this patch-level rationale is shown in the *Supplementary Material*.

To get a visual representation of each prototype d, we retrieve the top-k training patches that maximize its importance score, computed as the element-wise product of the normalized prototype activations for sample i and classification weights considered, based on the scenario being TIL or CIL: $\mathbf{m}_d = \max_c \mathbf{p}_i^\top \mathbf{w}_d$. We empirically found the image patches with a score $\mathbf{m}_d > 0.01$ to be more intuitive, so only patches whose score exceeds this value are retained. In the TIL case, the search is restricted to each task's training images, so that for each task we get a full visual representation of the prototypes. Whereas in the CIL case, the full training set is considered. An example for the 4-tasks setting is shown in Figure 3.

Conclusions

We have introduced CIP-Net, an exemplar-free, prototype-based CL framework that delivers state-of-the-art performance compared to previous exemplar-free and self-explainable methods in both TIL and CIL settings. By grounding every decision in a combination of shared prototypes, CIP-Net offers explanations that remain useful and stable across tasks, effectively mitigates catastrophic forgetting, and avoids the increasing memory-related overhead commonly associated with prototype-based reasoning.

Limitations CIP-Net retains the interpretability benefits of prototype-based models but also their drawbacks, including the latent–human semantic gap, vulnerability to small adversarial perturbations, and other known issues (Gautam et al. 2023; Hoffmann et al. 2021; Kim et al. 2022; Nauta et al. 2021; Rymarczyk et al. 2022). Although the PIP-Net backbone generalizes beyond fine-grained datasets like CUB and CARS, CIP-Net has not yet been tested on them. Our frequency-based regularization constrains only some prototypes, so an inappropriate proportion can blur explanations or reduce accuracy, particularly over long task sequences. Frequent prototypes are not always meaningful, as they may represent common backgrounds (e.g., sky or sea), though their usefulness can depend on context: for instance, in CUB, a "sea" prototype aids seabird classification.

Future Work Promising research directions include devising better sparsity-promoting strategies for more compact explanations and exploring alternative prototype regularization schemes to address the previously noted limitations. Additionally, while PIP-Net is designed to recognize out-of-distribution samples, this has not been explored in CIP-Net yet; leveraging this capability for autonomous new-task detection could be an interesting direction.

Acknowledgments

This work was supported by the French National Research Agency (ANR) through the France 2030 program, under the PEPR "WAIT4" (ANR-22-PEAE-0008). This work has been carried out while Michela Proietti and Federico Di Valerio were enrolled in the Italian National Doctorate on Artificial Intelligence and the PhD in Engineering in Computer Science, respectively, run by Sapienza University of Rome.

References

Bai, G.; Ling, C.; Gao, Y.; and Zhao, L. 2023. *Saliency-Augmented Memory Completion for Continual Learning*, 244–252.

Chen, C.; Li, O.; Tao, D.; Barnett, A.; Rudin, C.; and Su, J. K. 2019. This looks like that: deep learning for interpretable image recognition. *Advances in neural information processing systems*, 32.

Cossu, A.; Spinnato, F.; Guidotti, R.; and Bacciu, D. 2023. A protocol for continual explanation of shap. *arXiv preprint arXiv:2306.07218*.

Cossu, A.; Spinnato, F.; Guidotti, R.; and Bacciu, D. 2024. Drifting explanations in continual learning. *Neurocomputing*, 127960.

Dhar, P.; Singh, R. V.; Peng, K.-C.; Wu, Z.; and Chellappa, R. 2019. Learning without memorizing. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 5138–5146.

Ebrahimi, S.; Petryk, S.; Gokul, A.; Gan, W.; Gonzalez, J. E.; Rohrbach, M.; and trevor darrell. 2021. Remembering for the Right Reasons: Explanations Reduce Catastrophic Forgetting.

Ede, S.; Baghdadlian, S.; Weber, L.; Nguyen, A.; Zanca, D.; Samek, W.; and Lapuschkin, S. 2022. Explain to not forget: Defending against catastrophic forgetting with xai.

Farajtabar, M.; Azizan, N.; Mott, A.; and Li, A. 2020. Orthogonal Gradient Descent for Continual Learning. In Chiappa, S.; and Calandra, R., eds., *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, 3762–3773. PMLR.

French, R. M. 1999. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3(4): 128–135.

Gautam, S.; Höhne, M. M.-C.; Hansen, S.; Jenssen, R.; and Kampffmeyer, M. 2023. This looks more like that: Enhancing self-explaining models by prototypical relevance propagation. *Pattern Recognition*, 136: 109172.

Goodfellow, I. J.; Mirza, M.; Xiao, D.; Courville, A.; and Bengio, Y. 2013. An empirical investigation of catastrophic forgetting in gradient-based neural networks. *arXiv preprint arXiv:1312.6211*.

Hoffmann, A.; Fanconi, C.; Rade, R.; and Kohler, J. 2021. This looks like that... does it? shortcomings of latent space prototype interpretability in deep networks. *arXiv preprint arXiv:2105.02968*.

Hou, S.; Pan, X.; Loy, C. C.; Wang, Z.; and Lin, D. 2019. Learning a unified classifier incrementally via rebalancing. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 831–839.

Hoyer, P. O. 2004. Non-negative matrix factorization with sparseness constraints. *Journal of machine learning research*, 5(Nov): 1457–1469.

Keswani, M.; Ramakrishnan, S.; Reddy, N.; and Balasubramanian, V. N. 2022. Proto2Proto: Can you recognize the car, the way I do? In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 10233–10243.

Kim, S. S.; Meister, N.; Ramaswamy, V. V.; Fong, R.; and Russakovsky, O. 2022. HIVE: Evaluating the human interpretability of visual explanations. In *European Conference on Computer Vision*, 280–298. Springer.

Kirkpatrick, J.; Pascanu, R.; Rabinowitz, N.; Veness, J.; Desjardins, G.; Rusu, A. A.; Milan, K.; Quan, J.; Ramalho, T.; Grabska-Barwinska, A.; et al. 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13): 3521–3526.

Krause, J.; Stark, M.; Deng, J.; and Fei-Fei, L. 2013. 3d object representations for fine-grained categorization. In *Proceedings of the IEEE international conference on computer vision workshops*, 554–561.

Li, Z.; and Hoiem, D. 2017. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12): 2935–2947.

Liu, Z.; Mao, H.; Wu, C.-Y.; Feichtenhofer, C.; Darrell, T.; and Xie, S. 2022. A convnet for the 2020s. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 11976–11986.

Mallya, A.; Davis, D.; and Lazebnik, S. 2018. Piggyback: Adapting a single network to multiple tasks by learning to mask weights. In *Proceedings of the European conference on computer vision (ECCV)*, 67–82.

Mallya, A.; and Lazebnik, S. 2018. Packnet: Adding multiple tasks to a single network by iterative pruning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 7765–7773.

Nauta, M.; Jutte, A.; Provoost, J.; and Seifert, C. 2021. This looks like that, because... explaining prototypes for interpretable image recognition. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 441–456. Springer.

Nauta, M.; Schlötterer, J.; Van Keulen, M.; and Seifert, C. 2023. Pip-net: Patch-based intuitive prototypes for interpretable image classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2744–2753.

Petit, G.; Popescu, A.; Schindler, H.; Picard, D.; and Delezoide, B. 2023. Fetril: Feature translation for exemplar-free class-incremental learning. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, 3911–3920.

Proietti, M.; Ragno, A.; and Capobianco, R. 2025. XAI-Guided Continual Learning: Rationale, Methods, and Future

Directions. *WIREs Data Mining and Knowledge Discovery*, 15(4): e70046. E70046 DMKD-00697.R2.

Proietti, M.; Wurman, P. R.; Stone, P.; and Capobianco, R. 2025. ProtoCRL: Prototype-based Network for Continual Reinforcement Learning. In *Reinforcement Learning Conference*.

Rajasegaran, J.; Khan, S. H.; Hayat, M.; Khan, F. S.; and Shah, M. 2020. iTAML: An Incremental Task-Agnostic Meta-learning Approach. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 13585–13594.

Rebuffi, S.-A.; Kolesnikov, A.; Sperl, G.; and Lampert, C. H. 2017. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2001–2010.

Rudin, C. 2019. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature machine intelligence*, 1(5): 206–215.

Rymarczyk, D.; Struski, Ł.; Górszczak, M.; Lewandowska, K.; Tabor, J.; and Zieliński, B. 2022. Interpretable image classification with differentiable prototypes assignment. In *European Conference on Computer Vision*, 351–368. Springer.

Rymarczyk, D.; Van De Weijer, J.; Zieliński, B.; and Twardowski, B. 2023. Icicle: Interpretable class incremental continual learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 1887–1898.

Saha, G.; and Roy, K. 2023. Saliency Guided Experience Packing for Replay in Continual Learning.

Shim, D.; Mai, Z.; Jeong, J.; Sanner, S.; Kim, H.; and Jang, J. 2021. Online Class-Incremental Continual Learning with Adversarial Shapley Value. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(11): 9630–9638.

Shin, H.; Lee, J. K.; Kim, J.; and Kim, J. 2017. Continual learning with deep generative replay. *Advances in neural information processing systems*, 30.

Wah, C.; Branson, S.; Welinder, P.; Perona, P.; and Belongie, S. 2011. The caltech-ucsd birds-200-2011 dataset.

Wang, L.; Zhang, X.; Su, H.; and Zhu, J. 2024. A Comprehensive Survey of Continual Learning: Theory, Method and Application. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1–20.

Yan, S.; Xie, J.; and He, X. 2021. Der: Dynamically expandable representation for class incremental learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 3014–3023.

Zhu, F.; Zhang, X.-Y.; Wang, C.; Yin, F.; and Liu, C.-L. 2021. Prototype augmentation and self-supervision for incremental learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 5871–5880.