Integrating Present and Past in Unsupervised Continual Learning

Yipeng Zhang[1,2†], Laurent Charlin[1,3], Richard Zemel[4], Mengye Ren[5]

[1]Mila, [2]Université de Montréal, [3]HEC Montréal, [4]Columbia University, [5]NYU
{yipeng.zhang, lcharlin}@mila.quebec
zemel@cs.columbia.edu, mengye@cs.nyu.edu

Abstract

We formulate a unifying framework for *unsupervised continual learning* (UCL), which disentangles learning objectives that are specific to the present and the past data, encompassing *stability*, *plasticity*, and *cross-task consolidation*. The framework reveals that many existing UCL approaches overlook cross-task consolidation and try to balance plasticity and stability in a shared embedding space. This results in worse performance due to a lack of within-task data diversity and reduced effectiveness in learning the current task. Our method, `Osiris`, which explicitly optimizes all three objectives on separate embedding spaces, achieves state-of-the-art performance on all benchmarks, including two novel benchmarks proposed in this paper featuring semantically structured task sequences. Compared to standard benchmarks, these two structured benchmarks more closely resemble visual signals received by humans and animals when navigating real-world environments. Finally, we show some preliminary evidence that continual models can benefit from such more realistic learning scenarios. [1]

1 Introduction

Humans and animals learn visual knowledge through continuous streams of experiences. In machine learning, *continual learning* is central to the development of learning from data that changes over time (Parisi et al., 2019). In continual learning, the learner typically encounters a non-stationary data stream through a series of learning episodes, similar to how humans learn (Kurby and Zacks, 2008), where each episode assumes a stationary data distribution. When this learning process is unlabelled, it is referred to as *unsupervised continual learning* (UCL) (Madaan et al., 2022; Fini et al., 2022). A popular current approach to UCL uses *self-supervised learning* (SSL) (Chen et al., 2020; Zbontar et al., 2021), which aims at learning invariant representations across pairs of visually similar images. Representations learned with SSL are believed to exhibit less *forgetting* (McCloskey and Cohen, 1989; French, 1999) than when learned with supervised objectives such as cross-entropy (Madaan et al., 2022; Davari et al., 2022). This confers SSL an important advantage since minimizing forgetting is a central objective of continual learning (Parisi et al., 2019).

However, models trained in the UCL setting still do not perform as well as models trained from iid data (offline). Learning from iid data is ideal but unrealistic when the underlying data distribution changes over time. In contrast, in UCL, models only have access to data from the present distribution and limited access to the past. Despite recent effort in advancing UCL (Madaan et al., 2022; Fini et al., 2022; Gomez-Villa et al., 2024), limited progress has been made in closing this performance gap.

In this study, we take a step back to examine what features current UCL methods learn and why such challenges persist. Our investigation yields a unifying framework that disentangles the learning objectives specific to the present and past data. In particular, our framework jointly optimizes: **1) plasticity** for learning within the present episode, **2) consolidation** for integrating the present and the past representations, and **3) stability** for maintaining the past representations.

[†]Work done partially at Columbia University.
[1]Published as a conference paper at CoLLAs 2024. Code is available at https://github.com/SkrighYZ/Osiris.

Our framework reveals that existing UCL methods either are not very effective at optimizing for **plasticity** or lack an explicit formulation of **cross-task**[2] **consolidation**. To improve plasticity, we find that it is crucial to project features to an embedding space exclusively for optimizing the current-task objective since optimizing other objectives on the same space can impair the model's ability to adapt to the new data distribution. Meanwhile, the lack of an explicit cross-task consolidation objective reduces the data diversity within a batch and causes the present-task representations to overlap with the past ones.

We address these two limitations by explicitly optimizing all three objectives in our framework with two parallel projector branches. Our innovations are inspired by recent progress in the SSL literature on contrastive loss decomposition (Wang and Isola, 2020) and leveraging multiple embedding spaces (Xiao et al., 2021). We name our method `Osiris` (*Optimizing stability, plasticity, and cross-task consolidation via isolated spaces*). `Osiris` achieves state-of-the-art performance on a suite of UCL benchmarks, including the standard Split-CIFAR-100 (Rebuffi et al., 2017) where tasks consist of randomly-drawn object classes. Additionally, we find that BatchNorm (Ioffe and Szegedy, 2015) is not suitable for UCL since it presupposes a stationary distribution, and advise future studies to use GroupNorm (Wu and He, 2018) instead.

Besides existing benchmarks, we consider the impact of structure in the sequence of episodes typically encountered in our everyday experiences. We build temporally structured task sequences of CIFAR-100 and Tiny-ImageNet images (Le and Yang, 2015; Deng et al., 2009) that resemble the structure of visual signals that humans and animals receive when navigating real-world environments. Interestingly, on the Structured Tiny-ImageNet benchmark, our method outperforms the offline iid model, showing some preliminary evidence that UCL algorithms can benefit from real-world task structures.

In summary, our main contributions are:

- We propose a unifying framework for UCL consisting of three objectives that integrate the present and the past tasks, and show that existing methods optimize a subset of these objectives but not all of them.
- We propose `Osiris`, a UCL method that directly optimizes all three objectives in our framework.
- For emulating more realistic learning environments, we propose two UCL benchmarks, Structured CIFAR-100 and Structured Tiny-ImageNet, that feature semantic structure on classes within or across tasks. We also propose two new metrics to measure plasticity and consolidation in UCL.
- We show that `Osiris` achieves state-of-the-art performance on all benchmarks, matching offline iid learning on the standard Split-CIFAR-100 with five tasks, and even outperforms it on the Structured Tiny-ImageNet benchmark.

2 Preliminaries

2.1 Self-Supervised Learning

Self-supervised learning (SSL) objectives are remarkably effective in learning good representations from unlabeled image data (Chen et al., 2020; Zbontar et al., 2021; He et al., 2020; Caron et al., 2020, 2021). Their idea is to enforce the model to be invariant to low-level cropping and distortions of the image, which encourages it to encode semantically meaningful features. We focus our analysis on the representative contrastive learning method SimCLR (Chen et al., 2020) because it has well-studied geometric properties on the feature space, which can help analyses (Wang and Isola, 2020), and it exhibits strong performance in our experiments.

Formally, let $\mathcal{A}$ be a stochastic function that applies augmentations (random cropping, color jittering, etc.) to $\boldsymbol{x}_i \sim \mathcal{D}$. For brevity, we fix the anchor $\boldsymbol{x}_i$ when describing the loss and denote the two augmented views of the anchor with $\boldsymbol{x}_i, \boldsymbol{x}_i' = \mathcal{A}(\boldsymbol{x}_i)$, and augmented views of other images with $\boldsymbol{x}_j\,(j \neq i)$. Let $f_\Theta : \mathcal{X} \to \mathbb{R}^{d_\mathrm{E}}$ denote our hypothesis family parameterized by Θ, where d_E is the output feature dimension of our model and $\mathcal{X} \supseteq \mathcal{D}$. Let $g_\Phi : \mathbb{R}^{d_\mathrm{E}} \to \mathbb{R}^{d_\mathrm{P}}$ be a non-linear function that projects $f_\Theta(\cdot)$ to some subspace of $\mathbb{R}^{d_\mathrm{P}}$. Then, the contrastive loss is defined as

$$\mathcal{L}^{\mathrm{SSL}}(\mathcal{D}; f_\Theta, g_\Phi) = \mathbb{E}_{\boldsymbol{x}_i \sim \mathcal{D},\, \{\boldsymbol{x}_j\}_j \overset{\mathrm{iid}}{\sim} \mathcal{D}} \left[-\log \frac{\exp(\boldsymbol{z}_i^\top \boldsymbol{z}_i')/\tau}{\exp(\boldsymbol{z}_i^\top \boldsymbol{z}_i')/\tau + \sum_{j \neq i} \exp(\boldsymbol{z}_i^\top \boldsymbol{z}_j)/\tau} \right], \tag{1}$$

[2]We use *task* and *episode* interchangeably.

where $\boldsymbol{z}_i$ is the normalized feature vector, i.e, $\boldsymbol{z}_i = g_\Phi(f_\Theta(\boldsymbol{x}_i))/\|g_\Phi(f_\Theta(\boldsymbol{x}_i))\|_2$ and similarly for $\boldsymbol{z}'_i$ and $\boldsymbol{z}_j$; $\boldsymbol{z}_i, \boldsymbol{z}'_i, \boldsymbol{z}_j \in \mathcal{S}^{d_\text{P}-1}$ where $\mathcal{S}^{d_\text{P}-1}$ denotes the d_P-dimensional unit sphere. τ is a temperature hyperparameter which we omit for brevity in our analysis. Intuitively, the gradient of this loss with respect to $\boldsymbol{z}_i$ is a weighted (with weights in $[0, 1]$) sum of $-\boldsymbol{z}'_i$ and every $\boldsymbol{z}_j$ $(j \neq i)$. The optimal model minimizes the distance between representations of positive pairs and maximizes the pairwise distance of different inputs. In the remainder of this paper, we use $\mathcal{L}^\text{SSL}(\mathcal{D}; f_\Theta, g_\Phi)$ to denote the contrastive loss in Eq. 1 within set $\mathcal{D}$ on the normalized output space of $g_\Phi \circ f_\Theta$ for brevity.

Generalized contrastive loss. We can extend Eq. 1 to a more general form, $\mathcal{L}^\text{SSL}(\mathbb{S}_+, \mathbb{S}_-; f_1, f_2)$, to denote the asymmetric contrastive loss where we use views of the same example in set $\mathbb{S}_+$ as positive pairs and views of examples in set $\mathbb{S}_-$ as negatives. One augmented view of the anchor, $\boldsymbol{x}_i$, is encoded by f_1 and the comparands, $\boldsymbol{x}'_i$ and $\boldsymbol{x}_j$'s, are encoded by f_2. Formally,

$$\mathcal{L}^\text{SSL}(\mathbb{S}_+, \mathbb{S}_-; f_1, f_2) = \mathbb{E}_{\boldsymbol{x}_i \sim \mathbb{S}_+, \{\boldsymbol{x}_j\}_j \sim \mathbb{S}_-} \left[-\log \frac{\exp\left(s(f_1(\boldsymbol{x}_i), f_2(\boldsymbol{x}'_i)\right)}{\exp\left(s(f_1(\boldsymbol{x}_i), f_2(\boldsymbol{x}'_i))\right) + \sum_j \exp\left(s(f_1(\boldsymbol{x}_i), f_2(\boldsymbol{x}_j))\right)} \right], \quad (2)$$

where $s(\cdot, \cdot)$ denotes the cosine similarity function.

2.2 Unsupervised Continual Learning

UCL studies the problem of representation learning on a set of unlabeled data distributions, $\{\mathcal{D}_1, \ldots, \mathcal{D}_T\}$, which the learner sequentially observes. Within each task $\mathcal{D}_t$, the learner is presented a batch of randomly selected examples $X = \{\boldsymbol{x}_i\}_{i=1}^B$ at each step, where $\boldsymbol{x}_i \overset{\text{iid}}{\sim} \mathcal{D}_t$ and B is the batch size. Suppose $\boldsymbol{x}$ is an image that belongs to some semantic concept class; then, the learner does not know either the class label or the task label t and only observes the image itself. The goal is to learn a good Θ such that $f_\Theta(\boldsymbol{x})$ encodes useful information about $\boldsymbol{x} \sim \mathcal{X}$ that can be directly used in subsequent tasks, where $\bigcup_{t=1}^T \mathcal{D}_t \subseteq \mathcal{X}$. A similar learning setup, with no knowledge of task labels but with class labels, is typically referred to as *class-incremental learning* in the supervised continual learning (SCL) literature (van de Ven et al., 2022).

To use SSL objectives as our learning signal in UCL, the expectation in Eq. 1 can be estimated by averaging the loss over all examples in a batch X. For each $\boldsymbol{x}_i$, we use $\boldsymbol{x}_i$ as the anchor and $X \setminus \{\boldsymbol{x}_i\}$ as negatives. Two common baselines are considered in UCL:

- **Sequential Finetuning (FT):** At task t, we only sample the batch X from $\mathcal{D}_t$.
- **Offline Training (Offline):** X is sampled iid from $\mathcal{D} = \bigcup_{t=1}^T \mathcal{D}_t$ throughout training.

In SCL, `FT` serves as the performance lower bound of models trained sequentially on $\mathcal{D}_{1\ldots T}$, whereas `Offline` is expected to be a *soft*[3] upper bound because it has access to the full dataset at any training step.

3 Dissecting the Learning Objective of UCL

We organize this section as follows. In Sec. 3.1, we describe three desirable properties for representation learning under the UCL setting. We then present `Osiris` in Sec. 3.2, which explicitly optimizes these properties. Finally, in Sec. 3.3, we show that `Osiris` is an instance of a more general framework, and that existing UCL methods implicitly address a subset of its components.

3.1 Three Desirable Properties

Features that facilitate *plasticity* or *stability* are commonly studied in UCL. In this section, we highlight another category of features, which we call *cross-task consolidation* features. We argue that UCL models need to consider plasticity, stability, and consolidation, in order to achieve good performance.

[3]See Sec. 4.4 for cases when UCL methods outperform `Offline`.

Plasticity and stability. Plasticity refers to the model's ability to optimize the learning objective on the *present task* $\mathcal{D}_t$. Intuitively, `FT` usually learns the present task well because it does not consider data in other tasks. On the other hand, stability refers to the model's ability to maintain performance on *past tasks*. This is commonly achieved either by regularizing the model—with some previous checkpoints—in their parameter space (Kirkpatrick et al., 2017; Schwarz et al., 2018; Chaudhry et al., 2018) or their output space (Buzzega et al., 2020; Fini et al., 2022), or by jointly optimizing the learning objective on some data sampled from $\mathcal{D}_{1...t-1}$ so that the model still performs well in expectation on previous tasks (Lin, 1992; Robins, 1995; Madaan et al., 2022). In practice, the distribution $\mathcal{D}_{1...t-1}$ is usually estimated online with a memory buffer.

The *stability-plasticity dilemma* (Ditzler et al., 2015) refers to the conundrum where parameters of continual learning models need to be stable in order not to forget learned knowledge but also need to be plastic to improve the representations continually. Tackling this challenge has been the main focus of prior work in both SCL and UCL.

Cross-task consolidation. Consolidation refers to the ability to distinguish data from *different tasks*. For example, if one task contains images of cats and dogs and another contains images of tigers and wolves, then learning to contrast dogs and wolves may yield fine-grained features that help reduce cross-task errors. Consolidation has been explored in SCL by leveraging class labels (Hou et al., 2019; Abati et al., 2020; Masana et al., 2021; Kim et al., 2022) or frozen representations (Aljundi et al., 2017; Wang et al., 2023), but it has been overlooked in UCL. Since we want to continually improve a unified representation for all seen data without labels, existing methods are not applicable.

3.2 Osiris: Integrating Objectives of Present and Past

Now we present `Osiris`, a method that explicitly optimizes plasticity, stability, and cross-task consolidation. All `Osiris`'s losses share the same encoder f_Θ but may use different nonlinear MLP projectors denoted by $g_\Phi, h_\Psi : \mathbb{R}^{d_\text{E}} \to \mathbb{R}^{d_\text{P}}$. We illustrate the method in Fig. 1.

To estimate $\mathcal{D}_{1...t-1}$ online, `Osiris` uses a memory buffer $\mathcal{M}$ to store data examples observed by the model so far. In this study, we assume the sampling strategy for data storing and retrieval are both uniform, with the former being achieved through online *reservoir sampling* (Vitter, 1985). Various works have studied non-uniform storing and retrieval (Aljundi et al., 2019a,b,c; Yoon et al., 2021; Gu et al., 2022), but they are orthogonal to this study. Throughout our analysis, we use X to denote a batch of data sampled iid from $\mathcal{D}_t$ and Y to denote a batch sampled iid from $\mathcal{M}$.

3.2.1 Plasticity Loss

The loss of the current task in the form of Eq. 1 is

$$\mathcal{L}_\text{current} = \mathcal{L}^\text{SSL}\left(X; f_\Theta, g_\Phi\right) . \tag{3}$$

It has been shown by Wang and Isola (2020) that, asymptotically, the perfect minimizer of $\mathcal{L}_\text{current}$ projects all $\boldsymbol{x} \in \mathcal{D}_t$ uniformly to the representation space, a unit hypersphere. We hypothesize that additional losses may prevent the model from learning this solution on $\mathcal{D}_t$ effectively, because the uniform distribution for $\mathcal{D}_t$ is unlikely the global optima for other losses.

Fortunately, prior SSL work offer insights on tackling the stability-plasticity dilemma: *the backbone encoder f_Θ encodes the necessary information that helps minimize SSL losses on multiple nonlinearly projected output spaces* (Xiao et al., 2021; Chen et al., 2020). To learn the new task effectively, we do not apply other losses on the output space of $g_\Phi \circ f_\Theta$; additional losses are calculated on representations projected from the outputs of f_Θ with some other projector h_Ψ. This allows the model to freely distribute $g_\Phi(f_\Theta(\boldsymbol{x}))$ in order to optimize $\mathcal{L}_\text{current}$, while potentially maintaining some other distributions of $f_\Theta(\boldsymbol{x})$ or $h_\Psi(f_\Theta(\boldsymbol{x}))$. The benefit of this approach is that f_Θ still encodes the features that help optimize $\mathcal{L}_\text{current}$ on the output space of $g_\Phi \circ f_\Theta$.

3.2.2 Stability Loss

Like most prior studies in continual learning, we introduce a loss to promote stability and reduce forgetting. We study two approaches, which we discuss next. They both use the projector h_Ψ.

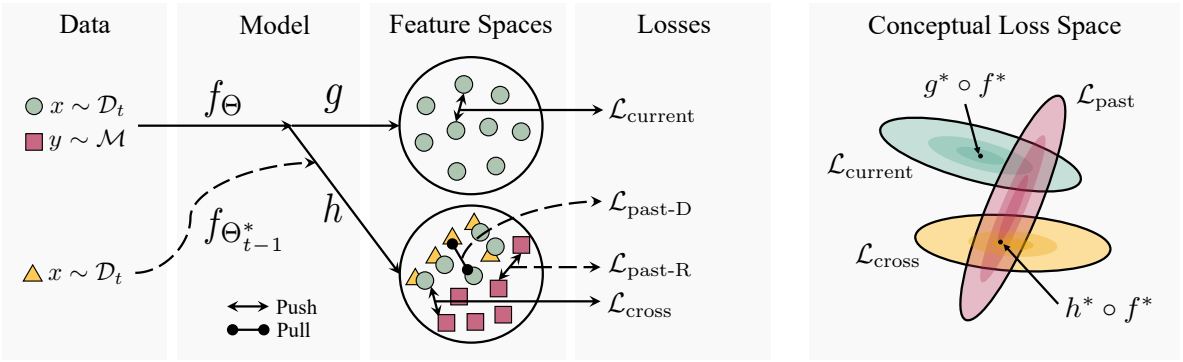


Figure 1: Left: illustration of our method. Dashed arrows denote optional computations because the stability loss $\mathcal{L}_{\text{past}}$ can be achieved through distillation or replay. Right: conceptual loss space. A separate projector helps with optimization.

Osiris-D(istillation). The first approach uses distillation and requires storing a frozen checkpoint of the encoder, $f_{\Theta^*_{t-1}}$, at the end of task $t-1$. It asks the current model to predict a data example from a batch of examples encoded by the checkpoint, therefore encouraging it to retain previously learned features. The loss can be written with the notation of Eq. 2 as

$$\mathcal{L}_{\text{past-D}} = \frac{1}{2}\left(\mathcal{L}^{\text{SSL}}\left(X, X; h_\Psi \circ f_{\Theta^*_{t-1}}, h_\Psi \circ f_\Theta\right) + \mathcal{L}^{\text{SSL}}\left(X, X; h_\Psi \circ f_\Theta, h_\Psi \circ f_{\Theta^*_{t-1}}\right)\right) . \tag{4}$$

The idea of $\mathcal{L}_{\text{past-D}}$ is similar to `CaSSLe` (Fini et al., 2022), but we distill our model with $f_{\Theta^*_{t-1}}(\boldsymbol{x})$ and not $g_{\Phi^*_{t-1}}(f_{\Theta^*_{t-1}}(\boldsymbol{x}))$, where $g_{\Phi^*_{t-1}}$ is the projector checkpoint. Our approach has several advantages: (a) it has been shown that the encoder output produces better representations than the projector output (Chen et al., 2020); (b) the gradient of our stability loss does not pass through g_Φ, which allows more freedom in exploring the current-task features; and (c) we do not need to store $g_{\Phi^*_{t-1}}$. In practice, we find the symmetric loss works better than having only the first term. Note that this induces negligible computational overheads because we can reuse the representations of X to compute the second term.

Osiris-R(eplay). The second approach does not require storing parameters. It applies the contrastive loss on Y alone. The loss can be written as

$$\mathcal{L}_{\text{past-R}} = \mathcal{L}^{\text{SSL}}\left(Y; f_\Theta, h_\Psi\right) . \tag{5}$$

This loss prevents forgetting by optimizing the learning objective on $\mathcal{D}_{1\ldots t-1}$, in expectation. It is similar to `ER`, to be discussed in Sec. 3.3, but uses a different projector.

Remark. Although using both $\mathcal{L}_{\text{past-R}}$ and $\mathcal{L}_{\text{past-D}}$ may yield better performance, we emphasize that this is not our goal. Because both losses aim to reduce forgetting, we use them to demonstrate the flexibility of our framework and investigate the pros and cons of replay versus distillation in our experiments. We expect that it is possible to use any replay or output regularization method explored in SCL (e.g., `DER` in Buzzega et al. 2020) as our stability loss, as long as they operate on the output space of h_Ψ.

3.2.3 Cross-Task Consolidation Loss

Recall from Sec. 3.2.1 that the perfect minimizer of $\mathcal{L}_{\text{current}}$ projects all data from $\mathcal{D}_t$ uniformly to the representation space. Similarly, representations of $\mathcal{D}_{1\ldots t-1}$ encoded by the perfect minimizer of $\mathcal{L}_{\text{past}}$ are also distributed uniformly. This means that the model may still suffer from representation overlaps between $\mathcal{D}_{1\ldots t-1}$ and $\mathcal{D}_t$ (or between a pair of tasks from $\mathcal{D}_{1\ldots t-1}$) even if they successfully optimize $\mathcal{L}_{\text{current}}$ and

$\mathcal{L}_{\text{past}}$. Although using separate projectors for $\mathcal{L}_{\text{current}}$ and $\mathcal{L}_{\text{past}}$ may help, we propose to introduce a loss that explicitly reduces the overlap.

Consider features that are useful in discriminating instances of $\mathcal{D}_t$ from those of $\mathcal{D}_{1\ldots t-1}$; they may not be readily encoded in $f_{\Theta^*_{t-1}}$, as the model has not seen any data from $\mathcal{D}_t$ at the end of task $t-1$. Thus, distillation does not help much in this case. Instead, we propose to leverage the memory $\mathcal{M}$. We find using an additional projector for this loss is unnecessary, so we reuse the output space of h_Ψ. Our consolidation loss is

$$\mathcal{L}_{\text{cross}} = \frac{1}{2}\left(\mathcal{L}^{\text{SSL}}\left(X, Y; h_\Psi \circ f_\Theta, h_\Psi \circ f_\Theta\right) + \mathcal{L}^{\text{SSL}}\left(Y, X; h_\Psi \circ f_\Theta, h_\Psi \circ f_\Theta\right)\right). \tag{6}$$

This loss contrasts the current task and the memory, which promotes learning features that help discriminate the instances from the current task and past tasks. Similar to Eq. 4, we use a symmetric loss for $\mathcal{L}_{\text{cross}}$.

Remark. One might expect that $\mathcal{L}_{\text{cross}}$ encourages representations of $\mathcal{M}$ to collapse to a single point to minimize their similarity with $\mathcal{D}_t$. We believe that this is unlikely because we find empirically that the stability loss helps the model to learn well-behaved representations of $\mathcal{M}$. For example, optimizing $\mathcal{L}_{\text{past-D}}$, which is calculated on $\mathcal{D}_t$, yields features that are transferable to $\mathcal{M}$. Similarly, $\mathcal{L}_{\text{past-R}}$ directly prevents collapses with the contrastive loss on $\mathcal{M}$. In addition, since data storing is performed online, $\mathcal{M}$ may contain examples from $\mathcal{D}_t$ while learning it, causing $\mathcal{L}_{\text{cross}}$ to contrast examples within $\mathcal{D}_t$. Nevertheless, because most SSL objectives encourage discrimination on the instance level rather than the class level, the model would still learn useful features with $\mathcal{L}_{\text{cross}}$ in this scenario.

3.2.4 Overall Loss

The overall loss of our model is

$$\mathcal{L} = \mathcal{L}_{\text{current}} + \frac{1}{2}\left(\mathcal{L}_{\text{cross}} + \mathcal{L}_{\text{past}}\right). \tag{7}$$

Similarly to Fini et al. (2022), we do not perform hyperparameter tuning on the loss weights (although it might yield better results) and fix the additional weights to sum to one to demonstrate the potential of this framework. In Fig. 1, we illustrate the conceptual optimization landscape. As discussed above, no solution minimizes all three losses individually in the same space. With isolated features spaces, our method reduces the extent to which $\mathcal{L}_{\text{cross}}$ or $\mathcal{L}_{\text{past}}$ directly constrain the model from learning the current task, which promotes plasticity. Note that f_Θ still needs to maintain a unified representation that preserves information useful for minimizing all three losses.

3.3 Expressing UCL Methods with A Unifying Framework

Now, we extend `Osiris` to a more general framework in the form of a unified optimization objective, based on the encoder-projector architecture of SSL models. At task t, it can be expressed as the following:

$$\begin{aligned}\mathcal{L}^*_{X \overset{\text{iid}}{\sim} \mathcal{D}_t, Y \overset{\text{iid}}{\sim} \mathcal{M}} &\left(X, Y; f_\Theta, f_{\Theta^*_{t-1}}, g_\Phi, g_{\Phi^*_{t-1}}, h_\Psi, m_\Omega\right) \\ &= \mathcal{L}_{\text{current}}(X; f_\Theta, g_\Phi) + \lambda_1 \mathcal{L}_{\text{cross}}(X, Y; f_\Theta, h_\Psi) + \lambda_2 \mathcal{L}_{\text{past}}(X, Y; f_\Theta, f_{\Theta^*_{t-1}}, g_\Phi, g_{\Phi^*_{t-1}}, m_\Omega).\end{aligned} \tag{8}$$

Here, we reuse the notations from Sec. 3.2 and introduce some new ones: m_Ω is an additional nonlinear MLP projector parameterized by Ω, and λ_1,λ_2 denote the loss weights ($\lambda_1, \lambda_2 \geq 0$). The three terms in Eq. 8 learn features that promote plasticity, cross-task consolidation, and stability, respectively. Note that it is not necessary for the objective to use all the arguments included in the parentheses or different parameterizations for g_Φ, h_Ψ, m_Ω; we only use Eq. 8 as the general form. *Nevertheless, an ideal model explores good features from the first two terms and uses the third term to ensure that it does not forget them over time.*

Interestingly, most existing UCL methods implicitly optimize the terms in Eq. 8, which we formally show next. Our analysis is similar to Wang et al. (2024), but is distinct in that we decompose the objective from a representation learning perspective rather than a methodology perspective. We fix the first term (shared across methods) and let f_Θ and g_Φ be the current-task encoder and projector as described in Sec. 2.1. One exception is dynamic model architectures where new parameters are added during learning (Yoon et al., 2018; Rusu et al., 2016); we do not discuss architecture-based methods as it would be equivalent to progressively adding arguments to the first term. We illustrate the focus of different UCL methods in Table 1.

Table 1: Comparison of UCL methods based on the feature components optimized. ‡Analysis for `LUMP` is based on applying *mixup* in the latent space.

	Method	Isolated Space for $\mathcal{L}_{\text{current}}$	$\mathcal{L}_{\text{cross}}$	$\mathcal{L}_{\text{past}}$
	FT	✓		
Replay	ER (Lin, 1992; Robins, 1995)			✓
Replay	ER+		✓	✓
Replay	ER++		✓	✓
Replay	DER (Buzzega et al., 2020)			✓
Replay	LUMP‡ (Madaan et al., 2022)		✓	✓
Distill.	EWC (Schwarz et al., 2018)			✓
Distill.	CaSSLe (Fini et al., 2022)	✓		✓
Distill.	POCON (Gomez-Villa et al., 2024)	✓		✓
Ours	Osiris-R (Replay)	✓	✓	✓
Ours	Osiris-D (Replay+Distillation)	✓	✓	✓

Elastic Weight Consolidation (EWC) is a classic baseline in continual learning (Kirkpatrick et al., 2017; Schwarz et al., 2018; Chaudhry et al., 2018). It uses $(\Theta - \Theta^*_{t-1})^\top F_t (\Theta - \Theta^*_{t-1})$ as $\mathcal{L}_{\text{past}}$, where F_t is the diagonal Fisher information matrix at task t that can be estimated with $F_{t-1}, \mathcal{D}_{t-1}, f_{\Theta^*_{t-1}}$, and $g_{\Phi^*_{t-1}}$ at the end of task $t-1$. `EWC` does not use a memory buffer, but needs to store F_t and Θ^*_{t-1}. It sets $\lambda_1 = 0$ and does not consider cross-task consolidation.

CaSSLe (Fini et al., 2022) is the previous state of the art in UCL. It uses the SSL objective as $\mathcal{L}_{\text{past}}$ to regularize the model on a separate output space projected from the main model output. The `CaSSLe` loss can be expressed with Eq. 2 as $\mathcal{L}^{\text{SSL}}(X, X; m_\Omega \circ f_\Theta, g_{\Phi^*_{t-1}} \circ f_{\Theta^*_{t-1}})$. `CaSSLe` uses another projector h on the output of g to form m, i.e., $m_\Omega = h_\Psi \circ g_\Phi$. This objective encourages the main model $g_\Phi \circ f_\Theta$ to encode information that can be used to predict representations of a previous checkpoint, $g_{\Phi^*_{t-1}} \circ f_{\Theta^*_{t-1}}$, thereby restricting the main model from losing features learned. The current task loss still acts on $g_\Phi \circ f_\Theta$. Since the gradient from the regularization is back-propagated to g, `CaSSLe` may still limit the model's ability to learn the new task effectively. It also does not consider cross-task consolidation ($\lambda_1 = 0$) and does not use a memory buffer.

Experience Replay (ER) (Lin, 1992; Robins, 1995) is a classic replay-based baseline. It uses $\mathcal{L}^{\text{SSL}}(Y; f_\Theta, g_\Phi)$ as $\mathcal{L}_{\text{past}}$ and sets $\lambda_1 = 0$. Unlike regularization-based methods, this loss may implicitly allow the model to discover new features that improve cross-task consolidation since Y includes examples from $\mathcal{D}_{1\ldots t-1}$.

ER+ and ER++ are our attempts to improve `ER` which, in addition to `ER`, exploit the abundance of the current task data for $\mathcal{L}_{\text{cross}}$. One such way is to add the memory examples into negatives augmenting the current task negatives, i.e., $\mathcal{L}^{\text{ER+}} = \mathcal{L}^{\text{SSL}}(X, X \cup Y; g_\Phi \circ f_\Theta, g_\Phi \circ f_\Theta)$; it is similar to the asymmetric loss used by Cha et al. (2021) in SCL and we refer to it as `ER+`. This loss indeed pushes representations of the current task and the memory apart, but it does not yield gradient that enforces *alignment* (Wang and Isola, 2020) of different views generated from memory examples. An alternative way is to use a full SSL loss on the union of the current batch and memory, i.e., $\mathcal{L}^{\text{ER++}} = \mathcal{L}^{\text{SSL}}(X \cup Y; f_\Theta, g_\Phi)$. We refer to this method as `ER++`.

Dark Experience Replay (DER) (Buzzega et al., 2020; Madaan et al., 2022) is an improved version of `ER`, where λ_1 remains 0 and $\mathcal{L}_{\text{past}}$ becomes a regularizer on the output space which is empirically estimated with $\frac{1}{|Y|} \sum_{\boldsymbol{y} \in Y} \|g_\Phi(f_\Theta(\boldsymbol{y})) - \boldsymbol{z}\|_2^2$, where $\boldsymbol{z}$ is the representation $g(f(\boldsymbol{y}))$ encoded by the model when $\boldsymbol{y}$ is

stored into $\mathcal{M}$. It does not consider cross-task consolidation because this objective does not encourage learning new features.

LUMP (Madaan et al., 2022) is the state-of-the-art replay-based UCL method. It applies *mixup* (Zhang et al., 2018), a linear interpolation between $\boldsymbol{x} \in X$ and $\boldsymbol{y} \in Y$, to generate inputs: $\tilde{\boldsymbol{x}}_i = \nu \boldsymbol{x}_i + (1-\nu)\boldsymbol{y}_i$ where $\nu \sim \text{Beta}(\alpha, \alpha)$ with α being a hyperparameter. The batch $\tilde{X} = \{\tilde{\boldsymbol{x}}_i\}_{i=1}^{B}$ is passed to the only loss term of the model, $\mathcal{L}^{\text{SSL}}(\tilde{X}; f_\Theta, g_\Phi)$. This loss form does not fall directly into our framework because it is hard to disentangle the effect of the loss on points in between data examples. We give the proposition below and prove it in Appendix A.

Proposition 1. *Let $\nu \sim Beta(\alpha, \alpha)$ and let $\mathcal{L}^{LUMP}(X, Y; \nu, f_\Theta, g_\Phi) \coloneqq \mathcal{L}^{SSL}(\tilde{X}; f_\Theta, g_\Phi)$ be as described above. Define $\boldsymbol{z}_i \coloneqq g(f(\boldsymbol{x}_i))/\|g(f(\boldsymbol{x}_i))\|_2$, $\boldsymbol{u}_i \coloneqq g(f(\boldsymbol{y}_i))/\|g(f(\boldsymbol{y}_i))\|_2$, and $\tilde{\boldsymbol{z}}_i \coloneqq g(f(\tilde{\boldsymbol{x}}_i))/\|g(f(\tilde{\boldsymbol{x}}_i))\|_2$ for all $i \in \{1, \dots |X|\}$. Suppose that the representations are linear in between $\boldsymbol{x}_i$ and $\boldsymbol{y}_i$, i.e., $\tilde{\boldsymbol{z}}_i = \nu \boldsymbol{z}_i + (1-\nu)\boldsymbol{u}_i$. Then*

$$\mathbb{E}_\nu \left[\frac{\partial \mathcal{L}^{LUMP}}{\partial \boldsymbol{z}_i} \right] = \underbrace{-a_i \boldsymbol{z}_i' + \sum_{j \neq i} a_j \boldsymbol{z}_j}_{\mathcal{L}_{current}} + \underbrace{\sum_{j \neq i} b_j \boldsymbol{u}_j}_{\mathcal{L}_{cross}} - b_i \boldsymbol{u}_i', \tag{9}$$

$$\textit{and} \quad \mathbb{E}_\nu \left[\frac{\partial \mathcal{L}^{LUMP}}{\partial \boldsymbol{u}_i} \right] = \underbrace{-c_i \boldsymbol{u}_i' + \sum_{j \neq i} c_j \boldsymbol{u}_j}_{\mathcal{L}_{past}} + \underbrace{\sum_{j \neq i} d_j \boldsymbol{z}_j}_{\mathcal{L}_{cross}} - d_i \boldsymbol{z}_i', \tag{10}$$

where $a_{(\cdot)}, b_{(\cdot)}, c_{(\cdot)}, d_{(\cdot)} \geq 0$ are scalar functions of α and the softmax probabilities of predictions.

The first equation represents the gradient of the loss w.r.t. representations of examples in the current-task batch X. In contrast, the second equation represents the gradient w.r.t. examples in the batch Y sampled from the memory. Recall from Sec. 2.1 that $(\cdot)'$ denotes the representation of another view of the same input. The linearity assumption we make above may not seem to be true in general, but note that *mixup* aims to help the model behave linearly in between examples to generalize better (Zhang et al., 2018). Nevertheless, we aim to estimate `LUMP`'s effect with this decomposition. Besides the last term in each equation, which indicates the gradient that pulls $\boldsymbol{z}_i$ and $\boldsymbol{u}_i$ (the pair of examples being mixed) closer, Prop. 1 says that the gradient of `LUMP` with contrastive learning can be decomposed and affects all components of Eq. 8: (a) current task learning, (b) cross-task discrimination, and (c) past task learning. However, since all the losses act on the same output space and the coefficients are correlated, this does not allow flexible control of the learning emphasis.

4 Experiments

We organize this section as follows. In Sec 4.1, we describe our baselines and benchmarks, and discuss the incompatibility between BatchNorm and UCL. We then detail the evaluation metrics used in this study in Sec. 4.2. We show our results on the standard benchmarks in Sec. 4.3 and on the structured benchmarks in Sec. 4.4. Finally, in Sec. 4.5, we analyze the models' behavior with additional experiments.

4.1 Experimental Protocol

Baselines. We compare our method with `FT` and `Offline` defined in Sec. 2.2. We also compare with classic CL methods: online `EWC` (Schwarz et al., 2018), `ER` (Lin, 1992; Robins, 1995), `DER` (Buzzega et al., 2020) which we described in Sec. 3.3. We find that we can improve `DER`'s performance by normalizing the features before performing L2 regularization, so we report the performance of improved `DER` only. We also report the performance of `ER+` and `ER++` described in Sec. 3.3. In addition to these standard continual learning baselines, we compare with methods designed specifically for UCL, `CaSSLe` (Fini et al., 2022) and `LUMP` (Madaan et al., 2022). Concurrent work, `POCON` (Gomez-Villa et al., 2024), also aims to maximize plasticity for UCL. We use the online version of `POCON` for a fair comparison so that all methods observe

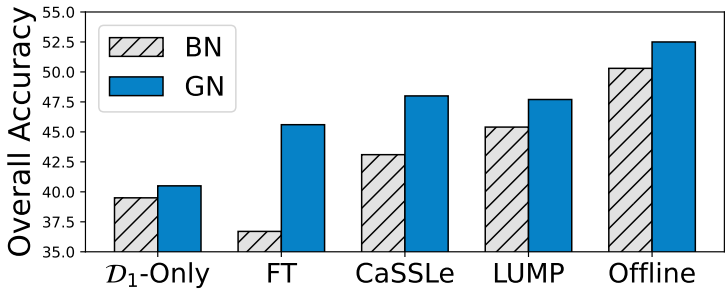


Figure 2: KNN accuracy of methods trained on the 20-task Split-CIFAR-100 with BatchNorm (BN) or GroupNorm (GN). $\mathcal{D}_1$-`Only` denotes an offline model trained for the same number of steps but only on the first task. `CaSSLe` (Fini et al., 2022) and `LUMP` (Madaan et al., 2022) are state-of-the-art UCL methods. The incompatibility between BN and UCL can be mitigated by using GN instead.

the data the same number of times. Finally, we do not compare with `C`^{2}`ASR` (Cheng et al., 2023) because it requires sorting the entire task stream and is an additional plug-in for UCL methods.

Benchmarks.

- **Standard Split-CIFAR-100.** Following Madaan et al. (2022); Fini et al. (2022), we evaluate the models on the 5-task and 20-task sequences of CIFAR-100 (Krizhevsky, 2009). It contains 50,000 32×32 images from 100 classes that are randomly grouped into a disjoint set of tasks.
- **Structured CIFAR-100.** In real-world environments, consecutive visual scenes are often similar and correlated in time. For example, seeing a great white shark immediately followed by office chairs is rather unlikely. We construct a temporally structured CIFAR-100 sequence by grouping the classes with the same superclass label (provided by the dataset) into a task and randomly shuffle the task order, which results in ten tasks. Examples of superclasses include vehicles, flowers, and aquatic mammals.
- **Structured Tiny-ImageNet.** Real-world environments also boast an abundance of hierarchies. We may visit different city blocks in an urban area and then tour multiple spots in a wild park. To create a task sequence that captures hierarchical environment structure, we use Tiny-ImageNet-200 (Le and Yang, 2015; Deng et al., 2009), which includes 100,000 images of size 64×64 categorized into 200 classes whose location spans different environments. We first use a pre-trained scene classifier trained on Places365 (Zhou et al., 2017) to classify all images into `indoor`, `city`, and `wild` environments. We then use a majority vote to decide the environment label for each class. Finally, we arrange the classes in the order of `indoor` → `city` → `wild` and group them into ten tasks in order. This leads to four tasks indoors, three tasks in the city, and three tasks in the wild. Compared with Structured CIFAR-100, which only enforces correlation within each task, this benchmark additionally imposes correlation between consecutive tasks. It provides a realistic structure, but also aligns nicely with the classic task-incremental learning setup.

Implementation details. We use a single-head ResNet-18 (He et al., 2016) as our backbone encoder. We use a two-layer MLP with a hidden dimension of 2048 and an output dimension of 128 as the projector. We use the ReLU function as activation after the hidden layer but not the output layer following Chen et al. (2020). For `POCON`, we follow the authors' implementation and use four-layer projectors for distillation. We set the memory size $|\mathcal{M}| = 500$ for all experiments. We train the models with a batch size of 256 for 200 epochs for UCL training, following Madaan et al. (2022). All methods use the same loss as `FT` during the first task. We provide additional training, data augmentation hyperparameters, and other details in Appendix B.

Incompatibility between BatchNorm and UCL. It has been shown that BatchNorm (Ioffe and Szegedy, 2015) is not suitable for SCL since its running estimates of the feature moments (over the batch

dimension) are biased towards the most recent task (Pham et al., 2021). An alternative normalization layer is GroupNorm (Wu and He, 2018), where the batch statistics are not needed because the normalization is applied along the feature dimension. The performance of BatchNorm and GroupNorm has not been investigated in UCL, although UCL usually requires much more training iterations than SCL. Thus, we hypothesize that using BatchNorm is harmful in UCL and the improvement of UCL methods over `FT` may not be as large as previously believed. In Fig. 2, we show that after we switch to GroupNorm, `FT` becomes a very strong baseline, and existing UCL methods do not help as much, although all methods show improved performance. Moreover, a model trained on only one task outperforms `FT` when equipped with BatchNorm but not GroupNorm, further showing the detrimental effect of BatchNorm in UCL. Therefore, we use GroupNorm in our experiments to highlight the core factors contributing to UCL performance.

4.2 Metrics for Evaluating UCL Methods

We have discussed the elements contributing to learning a good representation in UCL. We now introduce accompanying fine-grained metrics to evaluate them. After unsupervised training, we keep the encoders and discard the projectors following standard practice because projectors learn the invariance that minimizes the SSL loss and may discard too much information for downstream tasks (Chen et al., 2020). Following Madaan et al. (2022), let $A_{t,i}$ be the weighted KNN (Wu et al., 2018) test accuracy of the encoder $f_{\Theta_t^*}$ on task i after it finishes training on task t. Note that the KNN classifier does not know the task labels (see Appendix C for results when task labels are given); therefore, it is important for $f_{\Theta_t^*}$ to obtain a good representation geometry over the entire dataset. We report the mean and standard deviation of results obtained from three random seeds in all of our tables and plots. We use five metrics in this study; accuracy, forgetting, and forward transfer are commonly used and we propose knowledge gain and cross-task consolidation score to measure the model's ability to optimize $\mathcal{L}_{\text{current}}$ and $\mathcal{L}_{\text{cross}}$ in Eq. 8:

- **Overall Accuracy (A)** is the accuracy of the final model on all classes in the dataset: $A = \frac{1}{T}\sum_{i=1}^{T} A_{T,i}$.
- **Forgetting (F)** measures the difference between the model's best accuracy on task i at any point during training and its accuracy on task i after training: $F = \frac{1}{T-1}\sum_{i=1}^{T-1} \max_{t\in\{1,\dots T\}}(A_{t,i} - A_{T,i})$. Forgetting measures the model's ability to optimize $\mathcal{L}_{\text{past}}$ in Eq. 8.
- **Knowldege Gain (K).** It has been shown that representations learned in UCL are significantly less prone to forgetting and are more stable than their SCL counterparts (Davari et al., 2022; Madaan et al., 2022). On the other hand, models learned continually lose plasticity (Abbas et al., 2023). Thus, we use knowledge gain to quantify the accuracy increase on task i before and after the model is trained on it. It is defined as $K = \frac{1}{T-1}\sum_{i=2}^{T}(A_{i,i} - A_{i-1,i})$. Knowledge gain is similar to the SCL metrics proposed by Chaudhry et al. (2018); Koh et al. (2023), but it is simple to calculate and is more suitable for UCL because UCL models generally have a reasonable performance on a task before even learning on it thanks to the generalization capability of SSL. Knowledge gain quantifies the model's ability to optimize $\mathcal{L}_{\text{current}}$.
- **Cross-Task Consolidation (C)** is defined as the test accuracy of a task-level KNN classifier on the frozen representations of the final model. As discussed in the previous sections, new knowledge acquisition is quantified by both knowledge gain and the ability to learn features that discriminate data across tasks, i.e., to optimize $\mathcal{L}_{\text{cross}}$.
- **Forward Transfer (T)** quantifies the generalization ability of UCL models by measuring how much of the learned representation can be helpful to an unseen task. It is defined as $T = \frac{1}{T-1}\sum_{i=2}^{T}(A_{i-1,i} - R_i)$ where R_i is the accuracy of a randomly initialized model on task i. It is used by Madaan et al. (2022); Fini et al. (2022).

4.3 Results on the Standard Benchmarks

In Table 2, we report the results of all models on Split-CIFAR-100. Our first observation is that `FT` is already a very strong baseline with a final accuracy only 2% below the `Offline` model's accuracy on the 5-task sequence. Moreover, `Osiris-D` closes this gap with `Offline` on both overall accuracy ($p = 0.185$

Table 2: Results on standard Split-CIFAR-100 with five or 20 tasks. The best model in each column is made **bold** and the second-best model is underlined. †Improved `DER`. §Online version of `POCON`. We separate replay-based (top) and distillation-based methods (bottom) for easier comparisons.

	5-Task Split-CIFAR-100					20-Task Split-CIFAR-100				
	A (↑)	**F** (↓)	**K** (↑)	**C** (↑)	**T** (↑)	**A** (↑)	**F** (↓)	**K** (↑)	**C** (↑)	**T** (↑)
FT	50.7 (± 0.4)	2.9 (± 0.0)	9.0 (± 0.1)	59.1 (± 0.2)	30.7 (± 0.3)	45.6 (± 0.3)	2.9 (± 0.4)	2.5 (± 0.1)	47.2 (± 0.1)	28.9 (± 0.2)
ER	51.5 (± 0.4)	2.7 (± 0.4)	8.4 (± 0.3)	59.7 (± 0.4)	31.8 (± 0.2)	47.1 (± 0.7)	3.4 (± 0.6)	3.5 (± 0.4)	48.2 (± 0.5)	30.1 (± 0.2)
DER†	51.0 (± 0.6)	3.0 (± 0.7)	9.6 (± 0.3)	59.0 (± 0.4)	30.5 (± 0.1)	45.7 (± 0.1)	2.6 (± 0.2)	2.6 (± 0.4)	47.2 (± 0.2)	28.7 (± 0.2)
LUMP	50.2 (± 0.6)	1.4 (± 1.1)	7.3 (± 0.3)	58.4 (± 0.4)	30.2 (± 0.1)	47.7 (± 1.1)	2.6 (± 0.9)	3.1 (± 0.3)	49.1 (± 1.0)	29.7 (± 0.0)
ER+	51.8 (± 0.6)	3.4 (± 0.5)	**10.2** (± 0.3)	60.1 (± 0.3)	31.1 (± 0.4)	46.7 (± 0.3)	3.1 (± 0.3)	4.4 (± 0.0)	48.0 (± 0.1)	28.8 (± 0.1)
ER++	51.8 (± 0.3)	2.9 (± 0.6)	9.1 (± 0.4)	59.8 (± 0.3)	31.6 (± 0.4)	47.7 (± 0.3)	3.7 (± 0.3)	**5.0** (± 0.4)	49.0 (± 0.5)	30.0 (± 0.1)
Osiris-R (Ours)	52.3 (± 0.5)	2.5 (± 0.7)	8.5 (± 0.1)	60.1 (± 0.1)	**32.1** (± 0.2)	49.3 (± 0.3)	3.1 (± 0.2)	4.7 (± 0.2)	50.5 (± 0.5)	**31.5** (± 0.3)
EWC	43.8 (± 0.6)	2.3 (± 0.8)	5.0 (± 0.4)	53.9 (± 0.7)	26.4 (± 0.3)	37.6 (± 0.2)	**1.7** (± 0.1)	2.0 (± 0.3)	39.4 (± 0.3)	21.4 (± 0.4)
CaSSLe	51.2 (± 0.3)	**0.7** (± 0.1)	7.2 (± 0.5)	59.5 (± 0.3)	30.4 (± 0.5)	48.0 (± 0.1)	1.9 (± 0.1)	-0.4 (± 0.2)	49.2 (± 0.2)	30.2 (± 0.2)
POCON§	50.6 (± 0.7)	3.2 (± 1.0)	9.3 (± 0.4)	59.3 (± 0.3)	30.6 (± 0.3)	45.2 (± 0.4)	3.0 (± 0.6)	2.7 (± 0.4)	46.8 (± 0.3)	28.8 (± 0.1)
Osiris-D (Ours)	**53.0** (± 0.2)	1.6 (± 0.5)	8.4 (± 0.3)	**60.5** (± 0.1)	31.7 (± 0.3)	**50.1** (± 0.2)	2.3 (± 0.2)	4.2 (± 0.3)	**51.3** (± 0.1)	**31.3** (± 0.4)
Offline	52.5 (± 0.4)	-	-	60.0 (± 0.4)	-	52.5 (± 0.4)	-	-	53.9 (± 0.2)	-

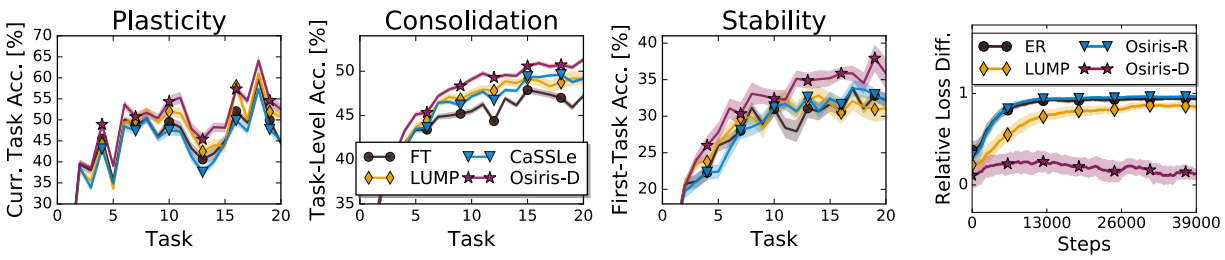
Table 3: Results on 10-task sequences on structured CIFAR-100 and Tiny-ImageNet. The two best models are marked. `Osiris-D` performs the best, surpassing `Offline` on Structured Tiny-ImageNet.

	Structured CIFAR-100					Structured Tiny-ImageNet				
	A (↑)	**F** (↓)	**K** (↑)	**C** (↑)	**T** (↑)	**A** (↑)	**F** (↓)	**K** (↑)	**C** (↑)	**T** (↑)
FT	45.0 (± 0.6)	5.5 (± 0.4)	7.8 (± 0.3)	59.2 (± 0.5)	28.7 (± 0.2)	34.2 (± 0.2)	4.7 (± 0.1)	5.7 (± 0.4)	43.5 (± 0.2)	26.0 (± 0.2)
LUMP	48.5 (± 0.7)	3.6 (± 0.6)	7.9 (± 0.5)	62.7 (± 0.7)	29.7 (± 0.6)	36.0 (± 1.0)	2.9 (± 1.1)	5.3 (± 0.2)	45.2 (± 0.8)	26.2 (± 0.4)
CaSSLe	47.4 (± 0.3)	**1.9** (± 0.3)	4.1 (± 0.2)	61.6 (± 0.1)	30.0 (± 0.3)	35.4 (± 0.3)	**2.1** (± 0.2)	3.2 (± 0.1)	43.9 (± 0.3)	26.7 (± 0.1)
POCON§	45.8 (± 0.5)	5.2 (± 0.3)	7.7 (± 0.3)	59.6 (± 0.8)	29.3 (± 0.4)	34.4 (± 0.6)	3.8 (± 0.8)	**5.5** (± 0.2)	43.6 (± 0.5)	25.7 (± 0.1)
Osiris-R (Ours)	49.0 (± 0.4)	5.0 (± 0.6)	**8.3** (± 0.6)	62.8 (± 0.2)	**31.7** (± 0.1)	36.3 (± 0.1)	3.6 (± 0.2)	**5.5** (± 0.1)	45.1 (± 0.1)	27.5 (± 0.3)
Osiris-D (Ours)	**49.8** (± 0.1)	4.4 (± 0.3)	**8.4** (± 0.2)	**64.2** (± 0.1)	**31.5** (± 0.2)	**37.5** (± 0.4)	2.9 (± 0.2)	5.0 (± 0.4)	**46.5** (± 0.2)	**28.1** (± 0.1)
Offline	52.5 (± 0.4)	-	-	67.9 (± 0.1)	-	36.8 (± 0.1)	-	-	46.4 (± 0.2)	-

by an unpaired t-test) and consolidation score. We hypothesize that this is in part attributed to that `Osiris-D` leverages the data ordering by contrasting the current task and the memory, although task labels are not explicitly given. On the other hand, `Offline` does not have any task label information. Among all methods, `Osiris` consistently achieves the highest overall accuracy, consolidation score, and forward transfer, regardless of the number of tasks. Comparing `Osiris-R` and `Osiris-D`, we find that there's still a trade-off between plasticity and stability. `Osiris-R` shows more knowledge gain at the expense of higher forgetting, and `Osiris-D` shows lower forgetting but sometimes lower knowledge gain.

`CaSSLe` shows low forgetting on both task sequences but lower knowledge gains than all the other methods except `EWC`, indicating that parallel projectors (in `Osiris`) might be a better choice than sequential ones (in `CaSSLe`) at improving plasticity. `POCON` is designed to maximize plasticity by distilling the model with a single-task expert, achieving the highest knowledge gain among the distillation-based methods (except `Osiris-D` on the 20 tasks). `LUMP` improves over `FT` on the 20-task but not on the 5-task sequence, and we hypothesize that it is because memory is more important when individual tasks are less diverse. Interestingly, the knowledge gain of `FT` is not the upper bound for UCL methods. This may be because UCL methods implicitly leverage learned representations or memory to help learn a new task (a form of forward transfer). For example, the methods with the highest knowledge gain on each benchmark are `ER+` and `ER++`, which both involve contrasting the current task and the memory, a component that is not present in `ER`.

General findings. Similar to Madaan et al. (2022), we find all UCL methods exhibit very low forgetting compared to previously reported numbers in SCL (Chaudhry et al., 2018). In general, the models with the



(a) (b)

Figure 3: (a) Interplay between plasticity (current-task accuracy), cross-task consolidation (task-level KNN accuracy), and stability (accuracy of the first task throughout training). `Osiris-D` balances the three aspects well and is usually the top performer. (b) Relative difference between the contrastive loss on past-task data and on memory for replay-based methods. All methods except `Osiris-D` show signs of overfitting.

lowest forgetting applies distillation. Indeed, a previously hypothesized criticism for replay-based methods is that they are prune to overfitting to memory (Fini et al., 2022), which we analyze empirically in Sec. 4.5. On the other hand, they show more plasticity and have higher knowledge gain on the new task.

4.4 Results on the Structured Benchmarks

We show the results of recent UCL methods on the structured benchmarks in Table 3. On Structured CIFAR-100, the methods show higher forgetting than on both random 5-task and 20-task sequences (a caveat for such a comparison is that here we have ten tasks). Nevertheless, `CaSSLe` and `LUMP` show relatively low forgetting but fail to address some of the other components of our framework (Eq. 8). All UCL methods improve over `FT`, and the two variants of `Osiris` outperform others in terms of knowledge gain, consolidation, forward transfer, and overall accuracy. This indicates that `Osiris` is robust to correlated task sequences.

On Structured Tiny-ImageNet, `FT` shows the highest knowledge gain, which means it benefits more from intra-task similarity. We hypothesize that contrastive learning benefits from high intra-task similarity because it provides hard negative pairs. `Osiris` again outperforms other UCL methods in terms of knowledge gain, consolidation, and forward transfer. Surprisingly, `Osiris-D` obtains better accuracy than `Offline` ($p = 0.008$ by an unpaired t-test). From a curriculum-learning perspective, this suggests that the realistic, hierarchical structure offers a better task ordering than randomly constructed scenarios. We explore how such a task ordering affects the representation structure in the next section.

4.5 Analysis

Balancing stability, plasticity, and consolidation. We now examine how UCL methods balance plasticity, cross-task consolidation, and stability. We use `Osiris-D` for our analysis in this section since `Osiris-R` shows similar behavior. Similar to the analysis provided by Gomez-Villa et al. (2024), in Fig. 3, we plot for different UCL methods the current task accuracy (plasticity), task-level KNN accuracy (cross-task consolidation), and accuracy of the first task (stability) throughout training on standard, 20-task Split-CIFAR-100. We plot the accuracy for all the other tasks in Appendix D. The first observation is that `Osiris` performs relatively well on all three aspects throughout training. `LUMP` and `CaSSLe` have similar overall accuracy in Table 2. They show the same level of cross-task consolidation in Fig. 3 because they do not directly enforce it. Among the two, `LUMP` shows higher plasticity but lower stability near the end of training. Both methods show better overall accuracy than `FT`, which may be attributed to their better consolidation scores.

Ablations. In Table 4, we show the results of our framework after removing each component. When using a shared projector for all three losses, the model's knowledge gain drops from 4.2% to 2.0%, which shows that using separate spaces helps plasticity, as we have hypothesized. When not using $\mathcal{L}_{\text{cross}}$, the model shows the lowest cross-task discrimination score among all the models being compared here. It also shows low

Table 4: Ablation of Osiris-D's components on 20-task Split-CIFAR-100. Isolating feature spaces is crucial for high knowledge gain; $\mathcal{L}_{\text{cross}}$ is important for knowledge gain and separation of task representations; $\mathcal{L}_{\text{past}}$ helps reduce forgetting.

	A (↑)	**F** (↓)	**K** (↑)	**C** (↑)	**T** (↑)
w/o isolated space	47.8 (± 0.3)	2.8 (± 0.3)	2.0 (± 0.4)	49.3 (± 0.2)	30.4 (± 0.3)
w/o $\mathcal{L}_{\text{cross}}$	45.9 (± 0.1)	2.5 (± 0.3)	1.8 (± 0.2)	47.2 (± 0.2)	29.2 (± 0.2)
w/o $\mathcal{L}_{\text{past}}$	49.2 (± 0.4)	2.6 (± 0.3)	**4.8** (± 0.5)	50.4 (± 0.3)	30.5 (± 0.2)
Full	**50.1** (± 0.2)	**2.3** (± 0.2)	4.2 (± 0.3)	**51.3** (± 0.1)	**31.3** (± 0.4)

Table 5: Within-environment accuracy on Structured Tiny-ImageNet. Both `FT` and `Osiris-D` outperforms `Offline` on Environment 3, but only `Osiris-D` achieves performance similar to `Offline` on Env. 1 and 2.

Method	Env. 1	Env. 2	Env. 3
Offline	44.2 (± 0.6)	60.9 (± 0.7)	50.3 (± 0.5)
FT	39.0 (± 0.2)	57.1 (± 0.4)	54.7 (± 0.5)
Osiris-D	43.7 (± 0.6)	60.2 (± 0.7)	55.1 (± 0.5)

forgetting because the only loss applied on output space of $h \circ f$ is now the distillation loss ($\mathcal{L}_{\text{past}}$). On the other hand, after removing $\mathcal{L}_{\text{past}}$, the model exhibits the highest knowledge gain, which means that learning the new task gains benefit from $\mathcal{L}_{\text{cross}}$. Finally, with all the components, `Osiris` balances all three aspects of our framework and achieves the best scores on all metrics except knowledge gain, without requiring manually adjusting the loss weights.

How does a structured task sequence affect representation? In Table 5, we show the within-environment accuracy for `Offline`, `FT`, and `Osiris-D`. Both `FT` and `Osiris` perform better than `Offline` on the last environment but not as well on the first two. Compared to `FT`, `Osiris` shows less forgetting and performs better in previously-observed environments. To examine their representation geometry, we plot the mean cosine similarity matrix between features of examples from pairs of classes, for these three methods, in Fig. 4. The matrix shows that classes within the last environment are naturally projected by `Offline` to nearby positions on the representation hypersphere. Differently, `FT` and `Osiris` can better distinguish between classes in the third environment. Among the two, `Osiris` distinguishes between different classes within the first two environments better. Together with within-environment accuracy, this provides evidence that UCL methods benefit from the ordered task sequence such that they distinguish examples in the last environment better. At the same time, `Offline` is less sensitive distinguishing these examples.

Does replay-based methods overfit to memory? It has been shown that replay-based methods can overfit to memory in SCL (Verwimp et al., 2021; Buzzega et al., 2021). The same phenomenon has been hypothesized by Fini et al. (2022) to also exist in UCL, but not yet empirically verified. As SSL learns features that generalize better (Madaan et al., 2022), we test this hypothesis empirically. In Fig. 3b, we plot for replay-based methods the relative difference between the contrastive loss of a batch sampled from all data observed ($\mathcal{L}_{\text{all}}$) and from the memory ($\mathcal{L}_{\text{mem}}$), both excluding any data from the current task. The relative difference is defined as $\frac{\mathcal{L}_{\text{all}} - \mathcal{L}_{\text{mem}}}{\mathcal{L}_{\text{all}}}$. A large difference could indicate the failure to generalize the representations learned from memory to past tasks, which defeats the purpose of replay. For `Osiris`, we plot the curves with $h \circ f$ here in Fig. 3b and for $g \circ f$ in Appendix D. In Fig. 3b, the curves for `ER`, `LUMP`, and `Osiris-R` increase at first and become relatively stable afterwards. Their final values are all much larger than zero, indicating the possibility of overfitting to memory. Since `Osiris-D` does not explicitly minimize the contrastive loss on the memory, it does not overfit to it and shows close-to-zero relative loss difference. This could explain why it always has lower forgetting than `Osiris-R`.

While Fig. 3b could indicate that `Osiris-R` overfits to the memory with $h \circ f$, the effect of $\mathcal{L}_{\text{cross}}$ on the encoder f appears to be less sensitive to the precision of the representations produced by $h \circ f$. In all of our

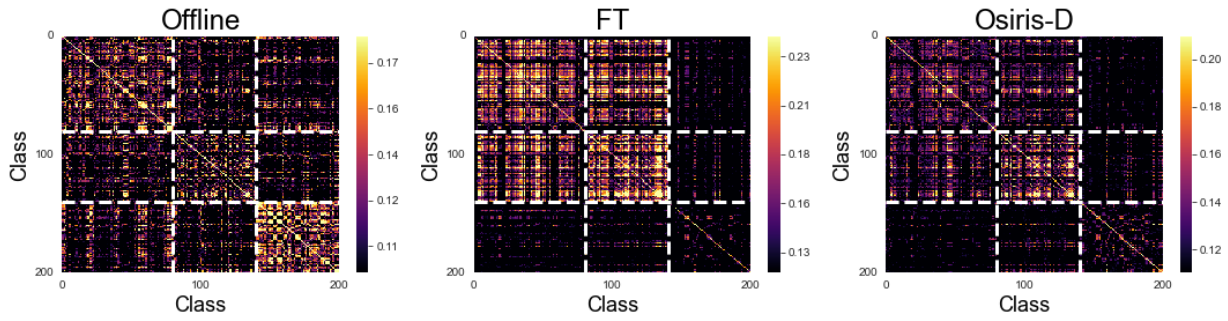


Figure 4: Mean cosine similarity between pairs of examples drawn from pairs of classes. Environment switches are marked with dashed white lines. Classes within the third environment are projected to nearby positions on the representation space by `Offline`, but not by `FT` and `Osiris-D`.

results (Tables 2, 3, 6, and 7), `Osiris-R` and `Osiris-D` are consistently the best performers in consolidation scores, which leads to good overall accuracy. Additionally, neither `Osiris-R` nor `Osiris-D` overfits with the representations produced by $g \circ f$ as shown in Fig. 5a. Overall, our findings empirically support the claim in prior work that using replay for UCL may cause the model to overfit to the memory (Fini et al., 2022), but also show that we still need the memory to improve consolidation, which is crucial for performance.

5 Related Work

Self-supervised learning. A large body of work in SSL belongs to the contrastive learning family (Chen et al., 2020; He et al., 2020; Misra and Maaten, 2020; Hadsell et al., 2006; Tian et al., 2020; Oord et al., 2018; Wu et al., 2018; Gutmann and Hyvärinen, 2010), which we focus on in this study. The main idea is to match the representations of two augmented views of the same image and repel the representations of different images to learn semantically meaningful representations. Clustering-based methods share high-level intuition but perform contrastive learning on the cluster level rather than the instance level (Caron et al., 2020; He et al., 2016). Other works have explored relaxing the need for negative pairs, usually by asymmetric architectures (Grill et al., 2020; Chen and He, 2021) or losses that enforce variance in representations (Zbontar et al., 2021; Bardes et al., 2022; Ermolov et al., 2021). SSL methods that work on transformers have also been proposed in recent years (He et al., 2022; Caron et al., 2021).

Continual learning. SCL methods are commonly partitioned into three categories. *Regularization*-based methods (Kirkpatrick et al., 2017; Schwarz et al., 2018; Chaudhry et al., 2018; Zenke et al., 2017; Aljundi et al., 2018; Castro et al., 2018; Douillard et al., 2020; Hou et al., 2019; Wu et al., 2019) regularize model parameters such that they do not drift too far from previous optima. *Replay*-based methods (Buzzega et al., 2020; Robins, 1995; Hayes et al., 2020; Rebuffi et al., 2017; Chaudhry et al., 2019; Lopez-Paz and Ranzato, 2017; Aljundi et al., 2019a; Ostapenko et al., 2019) use a memory buffer storing data from past task and use them for replay. Finally, *architecture*-based methods (Ostapenko et al., 2021; Rusu et al., 2016; Serra et al., 2018; Li et al., 2019) dynamically introduce new parameters for each task to reduce forgetting. Limited progress has been made in UCL. The area is first explored by Rao et al. (2019); Smith et al. (2021), but their work is limited to small datasets such as handwritten digits and is hard to scale. Recent work focuses on improving SSL-based UCL (Madaan et al., 2022; Fini et al., 2022; Gomez-Villa et al., 2024, 2022; Cheng et al., 2023). In contrast, we investigate what features are essential in UCL and offer new insights and practical guidance around normalization. SSL also helps SCL (Cha et al., 2021; Caccia et al., 2021) by improving the model's representations. It is worth mentioning that properties of representations learned with SSL in continual learning have been empirically studied (Davari et al., 2022; Galashov et al., 2023; Gallardo et al., 2021).

6 Conclusion

This work identifies three key components in UCL for integrating representation learning in the present and the past tasks: plasticity, stability and cross-task consolidation. Existing methods fall under our unifying framework by optimizing only a subset of objectives, whereas our proposed method `Osiris` explicitly optimizes and balances all three desiderata. `Osiris` achieves state-of-the-art performance on all UCL benchmarks and shows better accuracy on our realistic Structured Tiny-ImageNet benchmark than offline iid training. Our work sheds new light on the potential learning mechanisms of continual learning agents in the real world. Future work will extend our framework to non-contrastive SSL approaches and evaluate more realistic learning environments, such as lifelong video recordings.

Acknowledgment

We appreciate the constructive feedback from five anonymous reviewers. We also thank Lucas Caccia, Siddarth Venkatraman, and Michael Chong Wang for helpful discussions and Avery Ryoo for pointing out some typos in earlier drafts. LC and YZ acknowledge the generous support of the CIFAR AI Chair program. This work obtained support by the funds provided by the National Science Foundation and by DoD OUSD (R&E) under Cooperative Agreement PHY-2229929 (The NSF AI Institute for Artificial and Natural Intelligence). This research was also enabled in part by compute resources provided by Mila (mila.quebec).

References

Abati, D., Tomczak, J., Blankevoort, T., Calderara, S., Cucchiara, R., and Bejnordi, B. E. (2020). Conditional channel gated networks for task-aware continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3931–3940.

Abbas, Z., Zhao, R., Modayil, J., White, A., and Machado, M. C. (2023). Loss of plasticity in continual deep reinforcement learning. In *Conference on Lifelong Learning Agents (CoLLAs)*, volume 232 of *Proceedings of Machine Learning Research*, pages 620–636. PMLR.

Aljundi, R., Babiloni, F., Elhoseiny, M., Rohrbach, M., and Tuytelaars, T. (2018). Memory aware synapses: Learning what (not) to forget. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 139–154.

Aljundi, R., Belilovsky, E., Tuytelaars, T., Charlin, L., Caccia, M., Lin, M., and Page-Caccia, L. (2019a). Online continual learning with maximally interfered retrieval. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Aljundi, R., Chakravarty, P., and Tuytelaars, T. (2017). Expert gate: Lifelong learning with a network of experts. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3366–3375.

Aljundi, R., Kelchtermans, K., and Tuytelaars, T. (2019b). Task-free continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11254–11263.

Aljundi, R., Lin, M., Goujaud, B., and Bengio, Y. (2019c). Gradient based sample selection for online continual learning. *Advances in Neural Information Processing Systems (NeurIPS)*, 32.

Bardes, A., Ponce, J., and LeCun, Y. (2022). VICReg: Variance-invariance-covariance regularization for self-supervised learning. In *International Conference on Learning Representations (ICLR)*.

Buzzega, P., Boschini, M., Porrello, A., Abati, D., and Calderara, S. (2020). Dark experience for general continual learning: a strong, simple baseline. *Advances in Neural Information Processing Systems (NeurIPS)*, 33:15920–15930.

Buzzega, P., Boschini, M., Porrello, A., and Calderara, S. (2021). Rethinking experience replay: a bag of tricks for continual learning. In *International Conference on Pattern Recognition (ICPR)*, pages 2180–2187. IEEE.

Caccia, L., Aljundi, R., Asadi, N., Tuytelaars, T., Pineau, J., and Belilovsky, E. (2021). New insights on reducing abrupt representation change in online continual learning. In *International Conference on Learning Representations (ICLR)*.

Caron, M., Misra, I., Mairal, J., Goyal, P., Bojanowski, P., and Joulin, A. (2020). Unsupervised learning of visual features by contrasting cluster assignments. *Advances in Neural Information Processing Systems (NeurIPS)*, 33:9912–9924.

Caron, M., Touvron, H., Misra, I., Jégou, H., Mairal, J., Bojanowski, P., and Joulin, A. (2021). Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9650–9660.

Castro, F. M., Marín-Jiménez, M. J., Guil, N., Schmid, C., and Alahari, K. (2018). End-to-end incremental learning. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 233–248.

Cha, H., Lee, J., and Shin, J. (2021). Co2l: Contrastive continual learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9516–9525.

Chaudhry, A., Dokania, P. K., Ajanthan, T., and Torr, P. H. (2018). Riemannian walk for incremental learning: Understanding forgetting and intransigence. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 532–547.

Chaudhry, A., Ranzato, M., Rohrbach, M., and Elhoseiny, M. (2019). Efficient lifelong learning with a-GEM. In *International Conference on Learning Representations (ICLR)*.

Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. (2020). A simple framework for contrastive learning of visual representations. In *International Conference on Machine Learning (ICML)*, pages 1597–1607. PMLR.

Chen, X. and He, K. (2021). Exploring simple siamese representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15750–15758.

Cheng, H., Wen, H., Zhang, X., Qiu, H., Wang, L., and Li, H. (2023). Contrastive continuity on augmentation stability rehearsal for continual self-supervised learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5707–5717.

Davari, M., Asadi, N., Mudur, S., Aljundi, R., and Belilovsky, E. (2022). 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.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 248–255. IEEE.

Ditzler, G., Roveri, M., Alippi, C., and Polikar, R. (2015). Learning in nonstationary environments: A survey. *IEEE Computational Intelligence Magazine*, 10(4):12–25.

Douillard, A., Cord, M., Ollion, C., Robert, T., and Valle, E. (2020). Podnet: Pooled outputs distillation for small-tasks incremental learning. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 86–102. Springer.

Ermolov, A., Siarohin, A., Sangineto, E., and Sebe, N. (2021). Whitening for self-supervised representation learning. In *International Conference on Machine Learning (ICML)*, pages 3015–3024. PMLR.

Fini, E., Da Costa, V. G. T., Alameda-Pineda, X., Ricci, E., Alahari, K., and Mairal, J. (2022). Self-supervised models are continual learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9621–9630.

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

Galashov, A., Mitrovic, J., Tirumala, D., Teh, Y. W., Nguyen, T., Chaudhry, A., and Pascanu, R. (2023). Continually learning representations at scale. In *Conference on Lifelong Learning Agents (CoLLAs)*, pages 534–547. PMLR.

Gallardo, J., Hayes, T. L., and Kanan, C. (2021). Self-supervised training enhances online continual learning. In *British Machine Vision Conference (BMVC)*.

Gomez-Villa, A., Twardowski, B., Wang, K., and van de Weijer, J. (2024). Plasticity-optimized complementary networks for unsupervised continual learning. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 1690–1700.

Gomez-Villa, A., Twardowski, B., Yu, L., Bagdanov, A. D., and van de Weijer, J. (2022). Continually learning self-supervised representations with projected functional regularization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3867–3877.

Grill, J.-B., Strub, F., Altché, F., Tallec, C., Richemond, P., Buchatskaya, E., Doersch, C., Avila Pires, B., Guo, Z., Gheshlaghi Azar, M., et al. (2020). Bootstrap your own latent-a new approach to self-supervised learning. *Advances in Neural Information Processing Systems (NeurIPS)*, 33:21271–21284.

Gu, Y., Yang, X., Wei, K., and Deng, C. (2022). Not just selection, but exploration: Online class-incremental continual learning via dual view consistency. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7442–7451.

Gutmann, M. and Hyvärinen, A. (2010). Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 297–304. JMLR Workshop and Conference Proceedings.

Hadsell, R., Chopra, S., and LeCun, Y. (2006). Dimensionality reduction by learning an invariant mapping. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 1735–1742. IEEE.

Hayes, T. L., Kafle, K., Shrestha, R., Acharya, M., and Kanan, C. (2020). Remind your neural network to prevent catastrophic forgetting. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 466–483. Springer.

He, K., Chen, X., Xie, S., Li, Y., Dollár, P., and Girshick, R. (2022). Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16000–16009.

He, K., Fan, H., Wu, Y., Xie, S., and Girshick, R. (2020). Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9729–9738.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778.

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 (CVPR)*, pages 831–839.

Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning (ICML)*, pages 448–456. PMLR.

Kim, G., Xiao, C., Konishi, T., Ke, Z., and Liu, B. (2022). A theoretical study on solving continual learning. *Advances in Neural Information Processing Systems (NeurIPS)*, 35:5065–5079.

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 (PNAS)*, 114(13):3521–3526.

Koh, H., Seo, M., Bang, J., Song, H., Hong, D., Park, S., Ha, J.-W., and Choi, J. (2023). Online boundary-free continual learning by scheduled data prior. In *International Conference on Learning Representations (ICLR)*.

Krizhevsky, A. (2009). Learning multiple layers of features from tiny images. *University of Toronto*.

Kurby, C. A. and Zacks, J. M. (2008). Segmentation in the perception and memory of events. *Trends in Cognitive Sciences*, 12(2):72–79.

Le, Y. and Yang, X. (2015). Tiny imagenet visual recognition challenge. *CS 231N*, 7(7):3.

Li, X., Zhou, Y., Wu, T., Socher, R., and Xiong, C. (2019). Learn to grow: A continual structure learning framework for overcoming catastrophic forgetting. In *International Conference on Machine Learning (ICML)*, pages 3925–3934. PMLR.

Lin, L.-J. (1992). Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine Learning*, 8:293–321.

Lopez-Paz, D. and Ranzato, M. (2017). Gradient episodic memory for continual learning. *Advances in Neural Information Processing Systems (NeurIPS)*, 30.

Madaan, D., Yoon, J., Li, Y., Liu, Y., and Hwang, S. J. (2022). Representational continuity for unsupervised continual learning. In *International Conference on Learning Representations (ICLR)*.

Masana, M., Van de Weijer, J., Twardowski, B., et al. (2021). On the importance of cross-task features for class-incremental learning. *arXiv preprint arXiv:2106.11930*.

McCloskey, M. and Cohen, N. J. (1989). Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of Learning and Motivation*, volume 24, pages 109–165. Elsevier.

Misra, D. (2020). Mish: A self regularized non-monotonic activation function. In *British Machine Vision Conference (BMVC)*.

Misra, I. and Maaten, L. v. d. (2020). Self-supervised learning of pretext-invariant representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Oord, A. v. d., Li, Y., and Vinyals, O. (2018). Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*.

Ostapenko, O., Puscas, M., Klein, T., Jahnichen, P., and Nabi, M. (2019). Learning to remember: A synaptic plasticity driven framework for continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11321–11329.

Ostapenko, O., Rodriguez, P., Caccia, M., and Charlin, L. (2021). Continual learning via local module composition. *Advances in Neural Information Processing Systems (NeurIPS)*, 34:30298–30312.

Parisi, G. I., Kemker, R., Part, J. L., Kanan, C., and Wermter, S. (2019). Continual lifelong learning with neural networks: A review. *Neural Networks*, 113:54–71.

Pham, Q., Liu, C., and Steven, H. (2021). Continual normalization: Rethinking batch normalization for online continual learning. In *International Conference on Learning Representations (ICLR)*.

Qiao, S., Wang, H., Liu, C., Shen, W., and Yuille, A. (2019). Micro-batch training with batch-channel normalization and weight standardization. *arXiv preprint arXiv:1903.10520*.

Rao, D., Visin, F., Rusu, A., Pascanu, R., Teh, Y. W., and Hadsell, R. (2019). Continual unsupervised representation learning. *Advances in Neural Information Processing Systems (NeurIPS)*, 32.

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 (CVPR)*, pages 2001–2010.

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

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

Schwarz, J., Czarnecki, W., Luketina, J., Grabska-Barwinska, A., Teh, Y. W., Pascanu, R., and Hadsell, R. (2018). Progress & compress: A scalable framework for continual learning. In *International Conference on Machine Learning (ICML)*, pages 4528–4537. PMLR.

Serra, J., Suris, D., Miron, M., and Karatzoglou, A. (2018). Overcoming catastrophic forgetting with hard attention to the task. In *International Conference on Machine Learning (ICML)*, pages 4548–4557. PMLR.

Smith, J., Taylor, C., Baer, S., and Dovrolis, C. (2021). Unsupervised progressive learning and the stam architecture. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*.

Tian, Y., Krishnan, D., and Isola, P. (2020). Contrastive multiview coding. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 776–794. Springer.

van de Ven, G. M., Tuytelaars, T., and Tolias, A. S. (2022). Three types of incremental learning. *Nature Machine Intelligence*, 4(12):1185–1197.

Verwimp, E., De Lange, M., and Tuytelaars, T. (2021). Rehearsal revealed: The limits and merits of revisiting samples in continual learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9385–9394.

Vitter, J. S. (1985). Random sampling with a reservoir. *ACM Transactions on Mathematical Software (TOMS)*, 11(1):37–57.

Wang, L., Xie, J., Zhang, X., Huang, M., Su, H., and Zhu, J. (2023). Hierarchical decomposition of prompt-based continual learning: Rethinking obscured sub-optimality. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Wang, T. and Isola, P. (2020). Understanding contrastive representation learning through alignment and uniformity on the hypersphere. In *International Conference on Machine Learning (ICML)*, pages 9929–9939. PMLR.

Wang, Z., Li, Y., Shen, L., and Huang, H. (2024). A unified and general framework for continual learning. In *International Conference on Learning Representations (ICLR)*.

Wu, Y., Chen, Y., Wang, L., Ye, Y., Liu, Z., Guo, Y., and Fu, Y. (2019). Large scale incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 374–382.

Wu, Y. and He, K. (2018). Group normalization. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 3–19.

Wu, Z., Xiong, Y., Yu, S. X., and Lin, D. (2018). Unsupervised feature learning via non-parametric instance discrimination. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3733–3742.

Xiao, T., Wang, X., Efros, A. A., and Darrell, T. (2021). What should not be contrastive in contrastive learning. In *International Conference on Learning Representations (ICLR)*.

Yoon, J., Madaan, D., Yang, E., and Hwang, S. J. (2021). Online coreset selection for rehearsal-based continual learning. In *International Conference on Learning Representations (ICLR)*.

Yoon, J., Yang, E., Lee, J., and Hwang, S. J. (2018). Lifelong learning with dynamically expandable networks. In *International Conference on Learning Representations (ICLR)*.

You, Y., Gitman, I., and Ginsburg, B. (2017). Large batch training of convolutional networks. *arXiv preprint arXiv:1708.03888*.

Zbontar, J., Jing, L., Misra, I., LeCun, Y., and Deny, S. (2021). Barlow twins: Self-supervised learning via redundancy reduction. In *International Conference on Machine Learning (ICML)*, pages 12310–12320. PMLR.

Zenke, F., Poole, B., and Ganguli, S. (2017). Continual learning through synaptic intelligence. In *International Conference on Machine Learning (ICML)*, pages 3987–3995. PMLR.

Zhang, H., Cisse, M., Dauphin, Y. N., and Lopez-Paz, D. (2018). mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations (ICLR)*.

Zhou, B., Lapedriza, A., Khosla, A., Oliva, A., and Torralba, A. (2017). Places: A 10 million image database for scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*.

A Proof of Proposition 1

Proof. Let $\nu \sim \text{Beta}(\alpha, \alpha)$. Define $\boldsymbol{z}_i \coloneqq g(f(\boldsymbol{x}_i))/\|g(f(\boldsymbol{x}_i))\|_2$, $\boldsymbol{u}_i \coloneqq g(f(\boldsymbol{y}_i))/\|g(f(\boldsymbol{y}_i))\|_2$, and $\tilde{\boldsymbol{z}}_i \coloneqq g(f(\tilde{\boldsymbol{x}}_i))/\|g(f(\tilde{\boldsymbol{x}}_i))\|_2$ for all $i \in \{0, \dots |X|\}$. Assume that the representations are also linearly mixed in the representation space, i.e., $\tilde{\boldsymbol{z}}_i = \nu \boldsymbol{z}_i + (1-\nu)\boldsymbol{u}_i$.

Let $\tilde{\boldsymbol{z}}_i$ be the anchor, the representation of its other augmented view $\tilde{\boldsymbol{z}}_i{}'$ as the positive, and a set of representations of mixed examples $\{\tilde{\boldsymbol{z}}_j\}_{j \neq i}$ as negatives. Then we can express the LUMP loss as

$$\mathcal{L}^{\text{LUMP}}(X, Y; \nu, f_\Theta, g_\Phi) = \mathbb{E}_\nu \left[-\log \frac{\exp(\tilde{\boldsymbol{z}}_i{}^\top \tilde{\boldsymbol{z}}_i{}')}{\exp(\tilde{\boldsymbol{z}}_i{}^\top \tilde{\boldsymbol{z}}_i{}') + \sum_{j \neq i} \exp(\tilde{\boldsymbol{z}}_i{}^\top \tilde{\boldsymbol{z}}_j)} \right]. \tag{11}$$

For convenience, define scalar quantities p_i, p_j's as the softmax probability of predictions:

$$p_i = \frac{\exp(\tilde{\boldsymbol{z}}_i{}^\top \tilde{\boldsymbol{z}}_i{}')}{\exp(\tilde{\boldsymbol{z}}_i{}^\top \tilde{\boldsymbol{z}}_i{}') + \sum_{j \neq i} \exp(\tilde{\boldsymbol{z}}_i{}^\top \tilde{\boldsymbol{z}}_j)}, \quad \text{and} \tag{12}$$

$$p_j = \frac{\exp(\tilde{\boldsymbol{z}}_i{}^\top \tilde{\boldsymbol{z}}_j)}{\exp(\tilde{\boldsymbol{z}}_i{}^\top \tilde{\boldsymbol{z}}_i{}') + \sum_{k \neq i} \exp(\tilde{\boldsymbol{z}}_i{}^\top \tilde{\boldsymbol{z}}_k)}, \quad \text{if } j \neq i, \tag{13}$$

where $0 \leq p_i, p_j \leq 1$. Then gradient of the LUMP loss in Eq. 11 w.r.t. the anchor is

$$\mathbb{E}_\nu \left[\frac{\partial \mathcal{L}^{\text{LUMP}}}{\partial \tilde{\boldsymbol{z}}_i} \right] = \mathbb{E}_\nu \left[(p_i - 1)\tilde{\boldsymbol{z}}_i{}' + \sum_{j \neq i} p_j \tilde{\boldsymbol{z}}_j \right] \tag{14}$$

$$= \mathbb{E}_\nu \left[(p_i - 1)\left(\nu \boldsymbol{z}_i' + (1-\nu)\boldsymbol{u}_i'\right) + \sum_{j \neq i} p_j \left(\nu \boldsymbol{z}_j + (1-\nu)\boldsymbol{u}_j\right) \right] \tag{15}$$

$$= \mathbb{E}_\nu \left[(p_i - 1)(\nu)\boldsymbol{z}_i' + (p_i - 1)(1-\nu)\boldsymbol{u}_i' + \sum_{j \neq i}(p_j)(\nu)\boldsymbol{z}_j + \sum_{j \neq i}(p_j)(1-\nu)\boldsymbol{u}_j \right], \tag{16}$$

where the equality holds between lines 14 and 15 because of our linearity assumption. Therefore, the gradient w.r.t. the current-task example's representation is

$$\mathbb{E}_\nu \left[\frac{\partial \mathcal{L}^{\text{LUMP}}}{\partial \boldsymbol{z}_i} \right] = \mathbb{E}_\nu \left[\frac{\partial \mathcal{L}^{\text{LUMP}}}{\partial \tilde{\boldsymbol{z}}_i} \frac{\partial \tilde{\boldsymbol{z}}_i}{\partial \boldsymbol{z}_i} \right] = \mathbb{E}_\nu \left[\nu \frac{\partial \mathcal{L}^{\text{LUMP}}}{\partial \tilde{\boldsymbol{z}}_i} \right] \tag{17}$$

$$= \mathbb{E}_\nu \left[(p_i - 1)(\nu^2)\boldsymbol{z}_i' + (p_i - 1)(1-\nu)(\nu)\boldsymbol{u}_i' + \sum_{j \neq i}(p_j)(\nu^2)\boldsymbol{z}_j + \sum_{j \neq i}(p_j)(1-\nu)(\nu)\boldsymbol{u}_j \right] \tag{18}$$

$$= (p_i - 1)\mathbb{E}[\nu^2]\boldsymbol{z}_i' + (p_i - 1)\left(\mathbb{E}[\nu] - \mathbb{E}[\nu^2]\right)\boldsymbol{u}_i' + \sum_{j \neq i}(p_j)\mathbb{E}[\nu^2]\boldsymbol{z}_j + \sum_{j \neq i}(p_j)\left(\mathbb{E}[\nu] - \mathbb{E}[\nu^2]\right)\boldsymbol{u}_j \tag{19}$$

$$= (p_i - 1)\mathbb{E}[\nu^2]\boldsymbol{z}_i' + \sum_{j \neq i}(p_j)\mathbb{E}[\nu^2]\boldsymbol{z}_j + \sum_{j \neq i}(p_j)\left(\mathbb{E}[\nu] - \mathbb{E}[\nu^2]\right)\boldsymbol{u}_j + (p_i - 1)\left(\mathbb{E}[\nu] - \mathbb{E}[\nu^2]\right)\boldsymbol{u}_i'. \tag{20}$$

Notice that this is a weighted sum of $\boldsymbol{z}_i'$, $\{\boldsymbol{z}_j\}_j$, $\{\boldsymbol{u}_j\}_j$, and $\boldsymbol{u}_i'$. The first and second terms are the weighted representations of views of $\boldsymbol{x} \in X$ the other two terms are the weighted representations of views of $\boldsymbol{y} \in Y$.

Now, the coefficients involve p_i, p_j, which are the softmax probabilities given by Eq. 12 and Eq. 13; they also involve $\mathbb{E}[\nu], \mathbb{E}[\nu^2]$, which are the first and second moments of Beta(α, α) where $\mathbb{E}[\nu^2], \mathbb{E}[\nu] - \mathbb{E}[\nu^2] > 0$. Therefore, it is easy to see that the coefficients on $\boldsymbol{z}_i', \boldsymbol{u}_i'$ are negative, and the coefficients on $\{\boldsymbol{z}_j\}_j$ and $\{\boldsymbol{u}_j\}_j$ are positive. This conclusion holds as long as $\mathbb{E}[\nu^2], \mathbb{E}[\nu] - \mathbb{E}[\nu^2] > 0$ are satisfied; i.e., ν does not necessarily need to be sampled from a symmetric Beta distribution as chosen by LUMP.

This concludes our proof for the first equality. The proof for $\mathbb{E}_\nu \left[\frac{\partial \mathcal{L}^{\text{LUMP}}}{\partial \boldsymbol{u}_i} \right]$ is similar due to symmetry. □

Table 6: Results on standard Split-CIFAR-100 with five or 20 tasks. The ground truth task identity for each example is given to the model at test time, so the model predicts the most probable class within the task. The best model in each column is made **bold** and the second-best model is underlined. †Improved DER. §Online version of POCON. We separate replay-based (top) and distillation-based methods (bottom) for easier comparisons.

	5-Task Split-CIFAR-100					20-Task Split-CIFAR-100				
	A (↑)	**F** (↓)	**K** (↑)	**C** (↑)	**T** (↑)	**A** (↑)	**F** (↓)	**K** (↑)	**C** (↑)	**T** (↑)
FT	72.4 (± 0.3)	1.7 (± 0.3)	7.9 (± 0.0)	59.1 (± 0.2)	36.3 (± 0.5)	84.4 (± 0.1)	4.2 (± 0.1)	6.6 (± 0.0)	47.2 (± 0.1)	28.6 (± 0.1)
ER	72.9 (± 0.2)	1.1 (± 0.2)	6.9 (± 0.3)	59.7 (± 0.4)	36.9 (± 0.3)	85.1 (± 0.2)	3.0 (± 0.2)	5.1 (± 0.4)	48.2 (± 0.5)	29.6 (± 0.3)
DER†	72.5 (± 0.3)	1.6 (± 0.4)	**8.1** (± 0.2)	59.0 (± 0.4)	36.2 (± 0.4)	84.2 (± 0.2)	4.2 (± 0.2)	6.7 (± 0.5)	47.2 (± 0.2)	28.5 (± 0.2)
LUMP	71.2 (± 0.5)	0.9 (± 0.5)	5.8 (± 0.4)	58.4 (± 0.4)	35.8 (± 0.3)	85.3 (± 0.6)	2.4 (± 0.7)	4.3 (± 0.3)	49.1 (± 1.0)	29.3 (± 0.1)
ER+	73.0 (± 0.5)	1.7 (± 0.3)	7.9 (± 0.2)	60.1 (± 0.3)	36.8 (± 0.5)	85.1 (± 0.2)	3.9 (± 0.1)	6.4 (± 0.2)	48.0 (± 0.1)	29.2 (± 0.1)
ER++	72.5 (± 0.5)	1.3 (± 0.6)	7.0 (± 0.0)	59.8 (± 0.3)	36.8 (± 0.4)	85.3 (± 0.2)	2.8 (± 0.2)	4.9 (± 0.1)	49.0 (± 0.5)	29.6 (± 0.0)
Osiris-R (Ours)	73.3 (± 0.2)	1.3 (± 0.3)	6.7 (± 0.4)	60.1 (± 0.1)	**37.7** (± 0.1)	86.5 (± 0.2)	2.5 (± 0.2)	5.4 (± 0.5)	50.5 (± 0.5)	**30.2** (± 0.4)
EWC	65.6 (± 0.7)	1.4 (± 0.8)	3.7 (± 0.5)	53.9 (± 0.7)	32.8 (± 0.7)	79.0 (± 0.2)	**1.9** (± 0.1)	3.7 (± 0.1)	39.4 (± 0.3)	24.0 (± 0.2)
CaSSLe	72.6 (± 0.2)	**0.5** (± 0.4)	6.7 (± 0.5)	59.5 (± 0.3)	36.1 (± 0.4)	85.4 (± 0.1)	2.5 (± 0.1)	5.1 (± 0.2)	49.2 (± 0.2)	28.8 (± 0.2)
POCON§	72.1 (± 0.7)	2.2 (± 0.8)	**8.0** (± 0.5)	59.3 (± 0.3)	36.4 (± 0.4)	84.0 (± 0.2)	4.4 (± 0.4)	**6.9** (± 0.3)	46.8 (± 0.3)	28.3 (± 0.3)
Osiris-D (Ours)	**73.9** (± 0.1)	0.7 (± 0.3)	6.7 (± 0.5)	**60.5** (± 0.1)	37.3 (± 0.2)	**86.7** (± 0.3)	2.2 (± 0.3)	5.1 (± 0.2)	**51.3** (± 0.1)	30.0 (± 0.2)
Offline	74.0 (± 0.2)	-	-	60.4 (± 0.4)	-	88.7 (± 0.1)	-	-	53.9 (± 0.2)	-

B Implementation Details

General details. For GroupNorm, we set the number of groups to $\min(32, \lfloor \#\text{channels}/4 \rfloor)$. We replace the ReLU activation functions following GroupNorm with Mish activation (Misra, 2020) to avoid dying ReLUs, or *elimination singularities* (Qiao et al., 2019). We use the same data augmentation procedure and parameters as Zbontar et al. (2021); Grill et al. (2020), and change the resized image size to 32×32 for CIFAR-100 and 64×64 for Tiny-ImageNet. We set $\tau = 0.1$ for all contrastive losses. We train all our models for 200 epochs with SGD optimizer, learning rate of 0.03, weight decay of $5e^{-4}$, and batch size of $|X| = 256$ on all benchmarks, following Madaan et al. (2022). We do not use LARS (You et al., 2017) because we do not find it make significant differences on batches of size 256. We train `Offline` for the same number of steps as continual models. The models are trained on two NVIDIA Quadro RTX 8000 GPUs for all experiments. For KNN evaluation, we follow He et al. (2020); Wu et al. (2018)'s set up, where we set $k = 200$ and temperature $\tau_{\text{KNN}} = 0.1$.

UCL methods. All hyperparameters are fine-tuned manually with at least three values. We adapt the official implementations of `CaSSLe`, `LUMP` to our codebase. Since `POCON` is a concurrent work and its code has not been made public yet, we re-implement it. For online `EWC`, we normalize the diagonal Fisher information matrix for each task following the authors (Schwarz et al., 2018) and set $\gamma = 1$; we use 50, 100 as the weight on the regularization loss on the five and 20 task sequences of Split-CIFAR-100, respectively. For `LUMP`, we set the Beta distribution parameter $\alpha = 0.4$ on all experiments; it's the same value used by the authors on Tiny-ImageNet, and we find it performs better than the authors' parameter $\alpha = 0.1$ on CIFAR-100. For online `POCON`, we save the model checkpoint every $ds = 2,000$ steps for CIFAR-100 and $ds = 4,000$ steps for Tiny-ImageNet; we set the weights to 1 on all its loss terms. For `DER`, `ER`, and `ER+`, we set the weights to 1 on the additional loss terms. For all replay-based methods except for `LUMP` (which requires $|Y| = |X| = 256$), we uniformly sample $|Y| = \frac{3}{4}|X| = 192$ examples from the memory at each step for replay.

C Evaluation Results Given Task Identities

We do not provide access to the task identity for each example at test time in Sec. 4. In the scenario where task labels are given during evaluation (Madaan et al., 2022), it remains an open question whether consolidation provides benefits. We hypothesize that the consolidation term is beneficial because it potentially learns a

Table 7: Results on the 10-task sequences on structured CIFAR-100 and Tiny-ImageNet. The ground truth task identity for each example is given to the model at test time, so the model predicts the most probable class within the task. The two best models are marked. `Osiris-D` performs the best, surpassing `Offline` on Structured Tiny-ImageNet.

	Structured CIFAR-100					Structured Tiny-ImageNet				
	A (↑)	**F** (↓)	**K** (↑)	**C** (↑)	**T** (↑)	**A** (↑)	**F** (↓)	**K** (↑)	**C** (↑)	**T** (↑)
FT	64.6 (± 0.8)	7.6 (± 0.8)	10.0 (± 0.2)	59.2 (± 0.5)	29.9 (± 0.3)	57.8 (± 0.2)	6.2 (± 0.3)	6.4 (± 0.2)	43.5 (± 0.2)	34.4 (± 0.1)
LUMP	66.6 (± 0.4)	**3.7** (± 0.2)	7.2 (± 0.4)	62.7 (± 0.7)	30.6 (± 0.1)	59.5 (± 0.8)	**2.9** (± 1.1)	4.9 (± 0.2)	45.2 (± 0.8)	34.1 (± 0.3)
CaSSLe	66.4 (± 0.2)	4.2 (± 0.4)	7.7 (± 0.3)	61.6 (± 0.1)	30.5 (± 0.1)	59.0 (± 0.6)	4.3 (± 0.3)	5.4 (± 0.6)	43.9 (± 0.3)	34.7 (± 0.3)
POCON[§]	64.5 (± 0.4)	7.6 (± 0.6)	**9.8** (± 0.3)	59.6 (± 0.8)	30.0 (± 0.3)	58.1 (± 0.7)	5.5 (± 0.9)	**6.6** (± 0.1)	43.6 (± 0.5)	34.0 (± 0.3)
Osiris-R (Ours)	67.3 (± 0.2)	4.9 (± 0.2)	7.6 (± 0.1)	62.8 (± 0.2)	**32.3** (± 0.1)	59.9 (± 0.2)	4.4 (± 0.3)	5.9 (± 0.3)	45.1 (± 0.1)	35.2 (± 0.4)
Osiris-D (Ours)	**67.8** (± 0.1)	4.6 (± 0.1)	8.0 (± 0.2)	**64.2** (± 0.1)	32.0 (± 0.1)	**61.5** (± 0.1)	3.2 (± 0.3)	5.6 (± 0.5)	**46.5** (± 0.2)	**35.6** (± 0.1)
Offline	69.2 (± 0.3)	-	-	67.9 (± 0.1)	-	60.7 (± 0.2)	-	-	46.4 (± 0.2)	-

Table 8: Ablation of Osiris-D's components on 20-task Split-CIFAR-100. The ground truth task identity for each example is given to the model at test time, so the model predicts the most probable class within the task. $\mathcal{L}_{\text{cross}}$ benefits representations even in within-task discrimination when not considering classes from other tasks.

	A (↑)	**F** (↓)	**K** (↑)	**C** (↑)	**T** (↑)
w/o isolated space	85.6 (± 0.3)	2.8 (± 0.6)	4.9 (± 0.2)	49.3 (± 0.2)	29.8 (± 0.1)
w/o $\mathcal{L}_{\text{cross}}$	84.4 (± 0.2)	4.1 (± 0.3)	**6.2** (± 0.4)	47.2 (± 0.2)	28.9 (± 0.2)
w/o $\mathcal{L}_{\text{past}}$	86.3 (± 0.3)	3.1 (± 0.2)	6.1 (± 0.3)	50.4 (± 0.3)	**30.0** (± 0.1)
Full	**86.7** (± 0.3)	**2.2** (± 0.3)	5.1 (± 0.2)	**51.3** (± 0.1)	**30.0** (± 0.2)

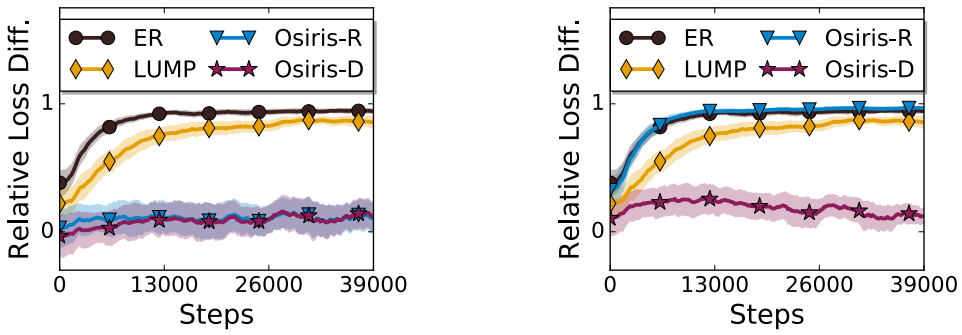
broader set of features than FT by increasing the diversity of batches for the contrastive loss.

To provide some evidence, we perform the same set of evaluations as in Sec. 4, but with task labels given to the model. In other words, the model predicts the most probable class among classes within the same task as the ground truth class i when calculating $A_{\cdot,i}$. The consolidation score (task-level KNN accuracy) is still calculated without task identities, and thus stays the same. The results are shown in Table 6, Table 7, and Table 8. We find that the consolidation score still correlates with the accuracy, and `Osiris-D` consistently outperforms other models. In Table 8, without $\mathcal{L}_{\text{cross}}$, `Osiris-D` experiences a 4.1% consolidation score drop and a 2.3% drop in accuracy. This shows that $\mathcal{L}_{\text{cross}}$ benefits the representation even in within-task discrimination.

Additionally, with consolidation, we expect the representations to separate all classes regardless of task identity. Therefore, they put the least assumptions on what classes we try to discriminate at test time and should appeal for a broader set of downstream use cases. For example, the test data is not required to be a subset of one of the tasks seen during training.

D Additional Plots

In this section, we show four additional sets of plots. First, to complement Fig. 3b, we plot the relative loss difference curves for `Osiris-R` and `Osiris-D` on the outputs of $g \circ f$, i.e., on the space where $\mathcal{L}_{\text{current}}$ is applied, in Fig. 5a. Then, Fig. 6 shows the accuracy versus extra storage for different UCL models on 20-task Split-CIFAR-100. Fig. 7 visualizes the pairwise cosine similarity distributions between examples from different class pairs. Finally, Fig. 8 through Fig. 11 plot the per-task accuracy of `FT`, `LUMP`, `CaSSLe`, and `Osiris-D` throughout training on 20-task Split-CIFAR-100.



(a) `Osiris`'s curves calculated with $g \circ f$.

(b) `Osiris`'s curves calculated with $h \circ f$.

Figure 5: Relative difference between contrastive loss on past-task data and on memory for replay-based methods. (a) All curves are calculated with the projector outputs where $\mathcal{L}_{\text{current}}$ is applied, i.e., with $g \circ f$. (b) Same as Fig. 3b, for `Osiris-D` and `Osiris-R`, we plot the curves calculated with the outputs of $h \circ f$ where $\mathcal{L}_{\text{past}}$ is applied. The curves for `ER` and `LUMP` are still calculated on their only projector branch, i.e, $g \circ f$. `Osiris-D` does not overfit on either branches.

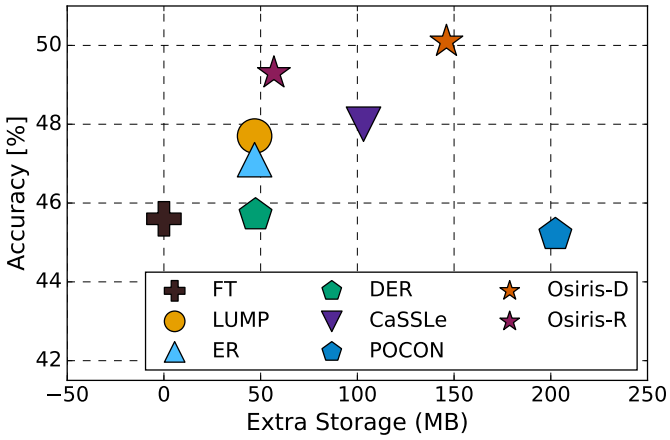


Figure 6: Accuracy on 20-task Split-CIFAR-100 versus additional storage. Storage is calculated by counting each additional model parameter besides the main model as well as each channel of each pixel in memory as a 64-bit float.

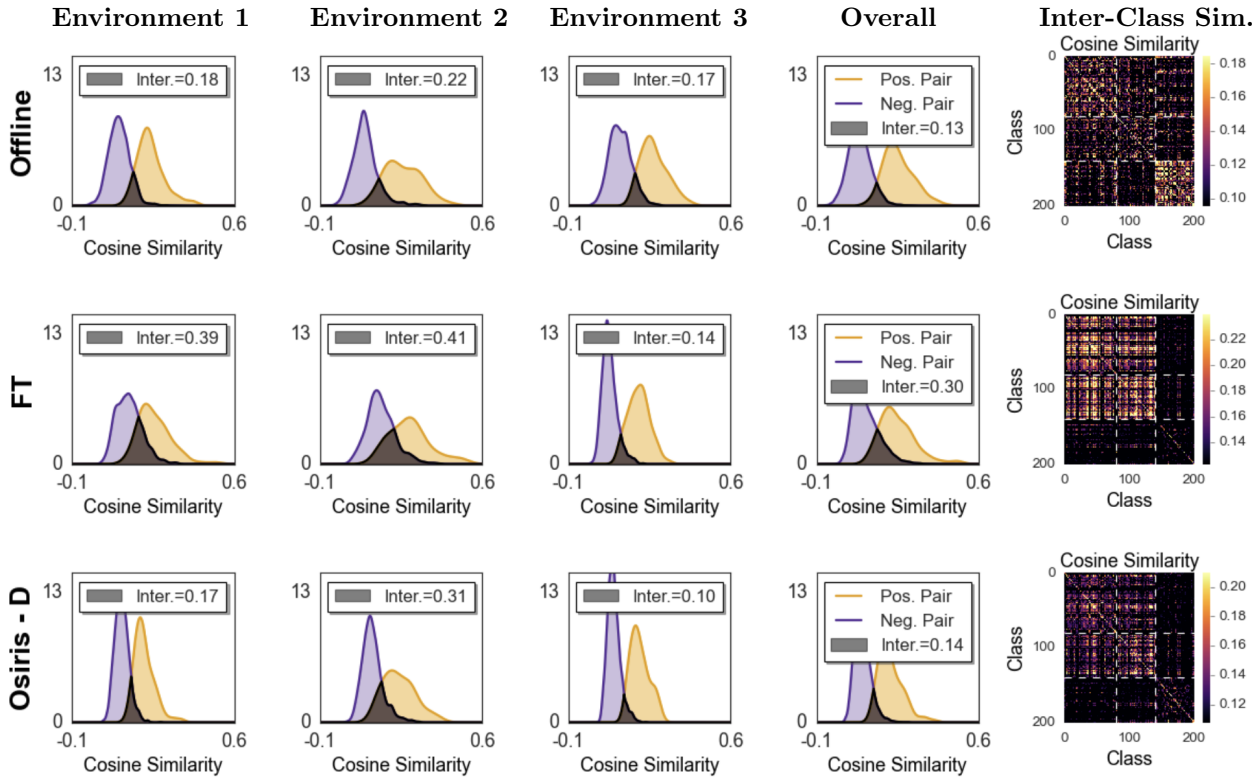


Figure 7: Left: pairwise feature similarity distributions between examples from the same (positive) or different (negative) classes, within each environment of Structured Tiny-ImageNet, as well as on all data (overall). The densities are estimated with a Gaussian kernel. The intersections are marked with a darker shade and the values are obtained by integrating the shaded area. An empty intersection of supports of the two distributions sufficiently entails a KNN classifier with perfect accuracy, but it is not necessary. Right: mean cosine similarity between pairs of examples drawn from pairs of classes.

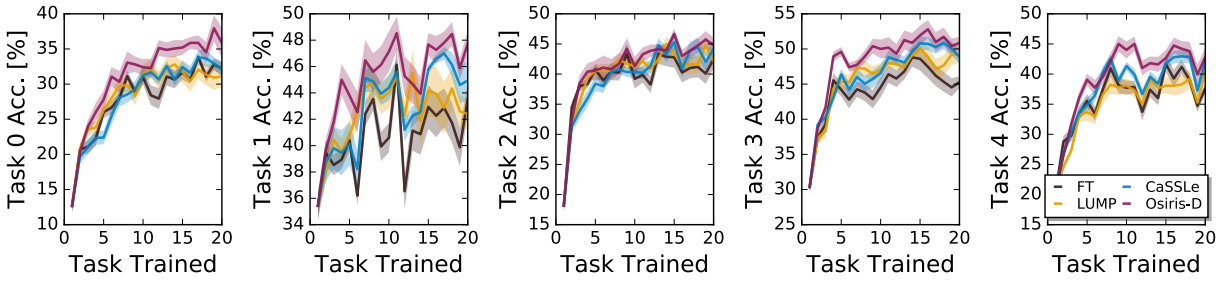


Figure 8: Task 1-5 accuracy throughout training on 20-task Split-CIFAR-100.

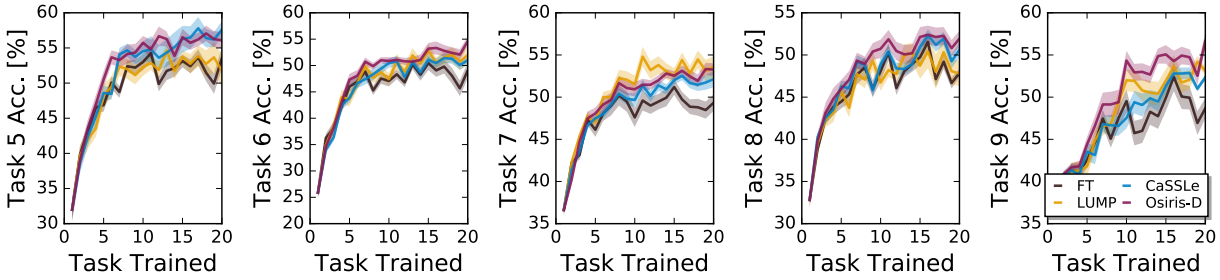


Figure 9: Task 6-10 accuracy throughout training on 20-task Split-CIFAR-100.

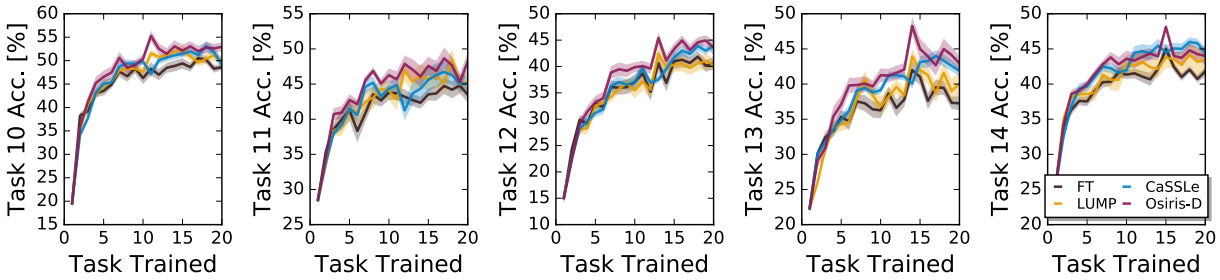


Figure 10: Task 11-15 accuracy throughout training on 20-task Split-CIFAR-100.

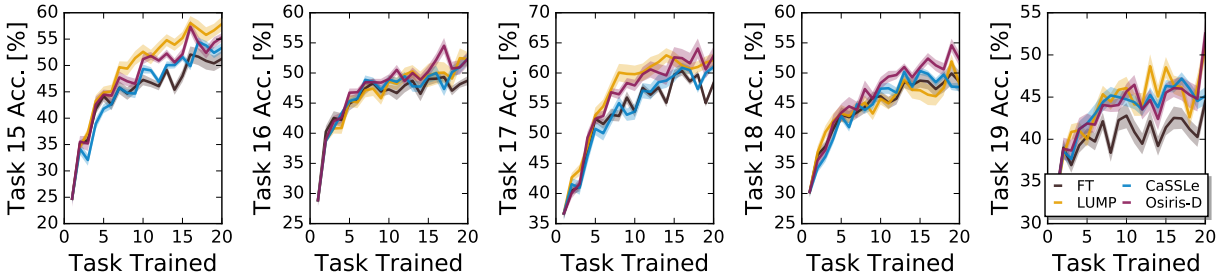


Figure 11: Task 15-20 accuracy throughout training on 20-task Split-CIFAR-100.

E Structured Class Order

We list below the classes within each task in Structured Tiny-ImageNet which we described in Sec. 4. Tasks 1-4 contain classes in an `indoor` environment. Task 5 contains classes in both `indoor` and `city` environments and serves as a soft transition. Tasks 6 and 7 contain classes in a `city` environment. Finally, tasks 9 and 10 contain classes in a natural, `wild` environment.

- Task 1: trilobite, binoculars, American lobster, bow tie, volleyball, banana, fur coat, barbershop, sombrero, water jug, bathtub, beer bottle, bell pepper, hourglass, ice cream, altar, lampshade, boa constrictor, frying pan, Christmas stocking.
- Task 2: turnstile, tabby, potter's wheel, chain, lemon, pill bottle, iPod, cockroach, oboe, punching bag, abacus, refrigerator, sock, bannister, candle, plate, ice lolly, Yorkshire terrier, apron, drumstick.
- Task 3: poncho, dining table, neck brace, guacamole, gasmask, backpack, academic gown, vestment, cash machine, CD player, espresso, potpie, syringe, orange, plunger, desk, Chihuahua, miniskirt, pretzel, bucket.
- Task 4: organ, chest, guinea pig, stopwatch, sandal, broom, pomegranate, barrel, wok, comic book, computer keyboard, meat loaf, pizza, basketball, remote control, teapot, mashed potato, teddy, cardigan, space heater.
- Task 5: Egyptian cat, rocking chair, wooden spoon, pop bottle, sunglasses, magnetic compass, sewing machine, jellyfish, beaker, Labrador retriever, dumbbell, nail, obelisk, lifeboat, steel arch bridge, moving van, gondola, military uniform, pole, beach wagon.
- Task 6: freight car, torch, umbrella, rugby ball, limousine, projectile, brass, go-kart, confectionery, pay-phone, German shepherd, reel, trolleybus, crane, fountain, jinrikisha, convertible, tractor, butcher shop, thatch.
- Task 7: suspension bridge, bullet train, kimono, picket fence, water tower, school bus, maypole, birdhouse, sports car, beacon, parking meter, bikini, swimming trunks, flagpole, triumphal arch, cannon, Persian cat, scoreboard, police van, lawn mower.
- Task 8: dragonfly, scorpion, American alligator, tarantula, lion, golden retriever, mantis, bullfrog, African elephant, snail, bighorn, baboon, sea cucumber, brown bear, cougar, seashore, king penguin, koala, lady-bug, tailed frog.
- Task 9: black widow, ox, grasshopper, acorn, fly, Arabian camel, coral reef, cliff dwelling, goldfish, goose, spider web, brain coral, barn, monarch, black stork, spiny lobster, standard poodle, sulphur butterfly, viaduct, albatross.
- Task 10: sea slug, chimpanzee, snorkel, slug, gazelle, dam, European fire salamander, hog, centipede, lesser panda, walking stick, lakeside, bee, mushroom, dugong, cauliflower, bison, alp, orangutan, cliff.