The Analysis and Continual Learning of Deep Feature Representations

Alexandra Gomez Villa

Universitat Autònoma de Barcelona

The Analysis and Continual Learning of Deep Feature Representations

A dissertation submitted by **Alexandra Gomez Villa** to the Universitat Autònoma de Barcelona in fulfilment of the degree of **Doctor of Philosophy** in the Departament de Ciències de la Computació.

Bellaterra, September 09, 2024

Director

Dr. Joost van de Weijer
Centre de Visió per Computador
Universitat Autònoma de Barcelona

Dr. Andrew D. Bagdanov
Media Integration and Communication Center
University of Florence

Dr. Bartlomiej Twardowski
Centre de Visió per Computador
Universitat Autònoma de Barcelona

Thesis committee

Dr. Simone Calderara
Department of Engineering Enzo Ferrari
University of Modena and Reggio Emilia

Dr. Pau Rodriguez Lopez
Machine Learning Research
Apple

Dr. Sebastian Cygert
IDEAS NCBR & Gdańsk University of Technology

This document was typeset by the author using $\LaTeX 2_{\varepsilon}$.

ISBN: 978-84-128712-1-0

Printed by Ediciones Gráficas Rey, S.L.

“It is impossible to be a mathematician without being a poet in soul.”
—Sofya Kovalevskaya (1850 - 1891)

Acknowledgements

First and foremost, I would like to thank my supervisors, Joost van de Weijer, Andrew Bagdanov, and Bartlomiej Twardowski, for generously sharing their extensive academic knowledge and playing a pivotal role in helping me become a better researcher. I am genuinely thankful for their guidance and understanding, which not only facilitated significant contributions to publications during my PhD but also showed me what it means to dedicate one's life to pursuing knowledge. Moreover, thanks to Kai Wang, he was always there to give a hand, discuss ideas or suggest solutions.

Second, I want to express my deepest gratitude to my partner, Mar. Thank you for always being my rock and accompanying me through the ups and downs of this journey. Your love, support, and companionship have been a source of strength and motivation.

Third, thanks to all the friends who have helped me make Barcelona my home: Hector, Diego, Laura, Javier, Adrian, and Nico. I must also thank German and Jhony, whose wise words have been a light in some of my darkest times. Special thanks also go to Fabio, Jefferson, and Geo, who taught me what it means to have a family abroad. I must also thank my furry friends Koko, Fry, and Jazz, who fill my days with drops of joy and companionship.

Four, I thank all the support received from the CVC administrative staff: Eva, Montse, Gigi, Mireia, Kevin, Joan, and so on. Thanks to all the CVC fellows who made my time there fun and exciting: Albin, Dipam, Fei, Pau, Adri, Sanket, and so on.

Last but not least, I would like to acknowledge my family in my own language: Gracias a mi abuelo Gonzalo, por llenar mis días con profundas discusiones científicas desde la más temprana edad, si he decidido seguir este camino ha sido sin duda por su influencia. Gracias a mi abuela Amparo, por cuidarme y darme su amor incondicional día a día. Gracias a mi madre Alina, la mujer más fuerte que conozco, quien me ha brindado su apoyo y cariño siempre.

Abstract

Artificial Intelligence (AI) is having an enormous impact across diverse application fields, from autonomous driving to drug discovery. This progress is predominantly driven by deep learning, a technology whose performance scales with data availability. As the demand for large datasets grows, researchers have developed transfer learning as an alternative strategy to leverage pre-trained models. However, transfer learning faces several challenges, including the need for extensive annotated data, difficulty in adapting to highly disjoint domains, and the lack of knowledge accumulation when adapting to new domains. Continual learning emerges as a promising approach to address these issues, enabling models to learn from shifting data distributions without forgetting previously acquired knowledge.

Self-supervised learning has shown remarkable success in alleviating data scarcity issues by leveraging unlabeled data to learn meaningful representations. This paradigm offers interesting challenges and opportunities for continual learning, potentially leading to more generalizable and adaptable models. Therefore, this thesis explores the combination of self-supervised and continual learning. While existing continual learning theory has predominantly focused on supervised learning, we extend continual learning theory to unsupervised scenarios. This allows learning of high-quality representations from a stream of unlabeled data.

In the first part of the thesis, we introduce a method that leverages feature distillation to combine the paradigms of continual learning and self-supervised learning to enable continual unsupervised representation learning. We then delve into the stability-plasticity dilemma, proposing strategies to optimize this trade-off in the context of exemplar-free unsupervised continual learning, particularly for scenarios involving numerous tasks and heterogenous architectures.

In the realm of prototype-based continual learning approaches, we find that feature drift is a primary cause of performance decline. Thus, we develop a prototype correction technique based on a projector that maps between the feature spaces of consecutive tasks. We show that our method can be applied to any self-supervised continual learning method to create the first exemplar-free, semi-supervised continual learning method.

Finally, we investigated how the contrastive self-supervised methods invariant to data augmentations can be improved for more transferable representations. We explore the impact of color augmentations on self-supervised learning and introduce a physics-based color augmentation method. This technique proves effective for tasks where intrinsic object color is crucial. We improve overall representation learning by combining color and shape information through different self-supervision modalities.

Key words: *continual learning, continual unsupervised representation learning, representation learning, self-supervised learning, lifelong learning*

Resumen

La Inteligencia Artificial (IA) está teniendo un impacto enorme en diversos campos de aplicación, desde la conducción autónoma hasta el descubrimiento de fármacos. Este progreso está impulsado predominantemente por el aprendizaje profundo, una tecnología cuyo rendimiento escala con la disponibilidad de datos. A medida que crece la demanda de grandes conjuntos de datos, los investigadores han desarrollado el aprendizaje por transferencia como una estrategia alternativa para aprovechar los modelos preentrenados. Sin embargo, el aprendizaje por transferencia enfrenta varios desafíos, incluyendo la necesidad de datos anotados extensos, la dificultad para adaptarse a dominios altamente disyuntos y la falta de acumulación de conocimiento al adaptarse a nuevos dominios. El aprendizaje continuo surge como un enfoque prometedor para abordar estos problemas, permitiendo que los modelos aprendan de distribuciones de datos cambiantes sin olvidar el conocimiento adquirido previamente.

El aprendizaje autosupervisado ha mostrado un éxito notable en la mitigación de problemas de escasez de datos al aprovechar datos no etiquetados para aprender representaciones significativas. Este paradigma ofrece interesantes desafíos y oportunidades para el aprendizaje continuo, potencialmente conduciendo a modelos más generalizables y adaptables. Por lo tanto, esta tesis explora la combinación del aprendizaje autosupervisado y continuo. Mientras que la teoría existente del aprendizaje continuo se ha centrado predominantemente en el aprendizaje supervisado, nosotros extendemos la teoría del aprendizaje continuo a escenarios no supervisados. Esto permite el aprendizaje de representaciones de alta calidad a partir de un flujo de datos no etiquetados.

En la primera parte de la tesis, introducimos un método que aprovecha la destilación de características para combinar los paradigmas del aprendizaje continuo y autosupervisado para permitir el aprendizaje continuo de representaciones no supervisadas. Luego, profundizamos en el dilema de estabilidad-plasticidad, proponiendo estrategias para optimizar este equilibrio en el contexto del aprendizaje continuo no supervisado sin ejemplos, particularmente para escenarios que involucran numerosas tareas.

En el ámbito del aprendizaje continuo basado en prototipos, encontramos que el desplazamiendo de características es una causa principal de la disminución del rendimiento. Por lo tanto, desarrollamos una técnica de corrección de prototipos basada en un proyector que mapea entre los espacios de características de tareas consecutivas. Demostramos que nuestro método puede aplicarse a cualquier método de aprendizaje continuo autosupervisado para crear el primer método de aprendizaje continuo semisupervisado sin ejemplos.

Finalmente, exploramos el impacto de las transformaciones de color en el aprendizaje autosupervisado e introducimos un método de transformación de color basado en la física. Esta técnica demuestra ser efectiva para tareas donde el color intrínseco del objeto es crucial. Mejoramos el aprendizaje general de representaciones combinando información de color y forma a través de diferentes modalidades de autosupervisión.

Palabras clave: *aprendizaje continuo, aprendizaje continuo no supervisado, aprendizaje de representaciones, aprendizaje autosupervisado, aprendizaje a lo largo de la vida*

Resum

La Intel·ligència Artificial (IA) està tenint un impacte enorme en diversos camps d'aplicació, des de la conducció autònoma fins al descobriment de fàrmacs. Aquest progrés està impulsat predominantment per l'aprenentatge profund, una tecnologia el rendiment de la qual escala amb la disponibilitat de dades. A mesura que creix la demanda de grans conjunts de dades, els investigadors han desenvolupat l'aprenentatge per transferència com una estratègia alternativa per aprofitar els models preentrenats. No obstant això, l'aprenentatge per transferència s'enfronta a diversos reptes, incloent la necessitat de dades anotades extenses, la dificultat d'adaptar-se a dominis altament disjunts i la manca d'acumulació de coneixement en adaptar-se a nous dominis. L'aprenentatge continu sorgeix com un enfocament prometedor per abordar aquests problemes, permetent que els models aprenguin de distribucions de dades canviants sense oblidar el coneixement adquirit prèviament.

L'aprenentatge autosupervisat ha mostrat un èxit notable en la mitigació de problemes d'escassetat de dades en aprofitar dades no etiquetades per aprendre representacions significatives. Aquest paradigma ofereix interessants reptes i oportunitats per a l'aprenentatge continu, potencialment conduint a models més generalitzables i adaptables. Per tant, aquesta tesi explora la combinació de l'aprenentatge autosupervisat i continu. Mentre que la teoria existent de l'aprenentatge continu s'ha centrat predominantment en l'aprenentatge supervisat, nosaltres estenem la teoria de l'aprenentatge continu a escenaris no supervisats. Això permet l'aprenentatge de representacions d'alta qualitat a partir d'un flux de dades no etiquetades.

A la primera part de la tesi, introduïm un mètode que aprofita la destil·lació de característiques per combinar els paradigmes de l'aprenentatge continu i autosupervisat per permetre l'aprenentatge continu de representacions no supervisades. Després, aprofundim en el dilema d'estabilitat-plasticitat, proposant estratègies per optimitzar aquest equilibri en el context de l'aprenentatge continu no supervisat sense exemples, particularment per a escenaris que involucren nombroses tasques.

En l'àmbit dels enfocaments d'aprenentatge continu basats en prototips, trobem que la deriva de característiques és una causa principal de la disminució del rendiment. Per tant, desenvolupem una tècnica de correcció de prototips basada en un projector que mapeja entre els espais de característiques de tasques consecutives. Demostrem que el nostre mètode es pot aplicar a qualsevol mètode d'aprenentatge continu autosupervisat per crear el primer mètode d'aprenentatge continu semisupervisat sense exemples.

Finalment, explorem l'impacte de les transformacions de color en l'aprenentatge autosupervisat i introduïm un mètode de transformacion de color basat en la física. Aquesta tècnica demostra ser efectiva per a tasques on el color intrínsec de l'objecte és crucial. Millorem l'aprenentatge general de representacions combinant informació de color i forma a través de diferents modalitats d'autosupervisió.

Paraules clau: *aprenentatge continu, aprenentatge continu de representacions no supervisat, aprenentatge de representacions, aprenentatge autosupervisat, aprenentatge al llarg de la vida*

Contents

List of Figures

List of Tables

1 Introduction

We live in an extraordinary era in which artificial intelligence (AI) is driving development across a wide range of fields, including autonomous driving, multimedia [39, 41, 86], drug discovery [10], astrophysics [101], material discovery [102], and ecological preservation [45]. Consequently, the impact of AI extends beyond academia, with significant adoption in the private sector. In fact, it is estimated that investments in AI research and applications are set to hit $500 billion by 2024 [118].

The current AI boom is predominantly based on end-to-end feature learning (deep learning), which had a major breakthrough in 2012 with the AlexNet architecture [77]. This method outperformed existing hand-crafted features on classification tasks. A well-known drawback of deep learning is its data-hungry nature; to learn effective features, models must consume large quantities of data. Furthermore, some models are known to scale as a function of dataset size and number of parameters (see Fig. 1.1). This has resulted in a scaling competition among big tech companies, where the player with the largest model and dataset is poised to win. The question remains: is there a limit to dataset and model size? To address this data-intensive approach, researchers have developed alternative strategies, such as transfer learning of pre-trained models.

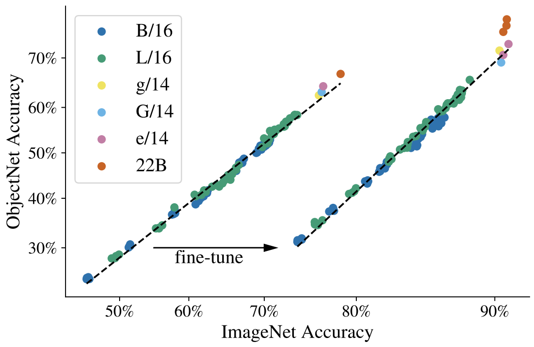


Figure 1.1: Scaling a vision transformer from 4 billion of parameters to 22 billion was proved to improve overall performance (image taken from [28]).

Transfer learning in deep neural networks facilitates the use of large, pretrained models on domains with limited annotated data. Generally, it involves fine-tuning the pretrained network on the data from the new domain. However, transfer learning has several drawbacks. The training of large general models (the pretrained models) requires massive amounts of annotated data, which is labor-intensive to obtain. Moreover, transfer learning of models may struggle with domain-specific nuances and rare edge cases not present in their original training data [78]. Finally, transfer learning focuses on the best performance on a new downstream task, the model may entirely forget the old task. In this sense, there is no knowledge accumulation and valuable features from past tasks will not be available for future use.

In response to the challenges of obtaining large annotated datasets, the machine learning community has made significant strides in developing self-supervised learning algorithms. These methods leverage unlabeled data to learn meaningful representations, reducing the reliance on human-annotated samples. Techniques such as contrastive learning, masked language modeling, and autoregressive prediction have shown remarkable success across various domains. For instance, models like BERT [30] have revolutionized natural language processing tasks, while approaches like SimCLR [22] and Barlow twins [157] have advanced computer vision. Self-supervised learning has not only improved the efficiency of data utilization but has also led to more robust and generalizable models, often surpassing their supervised counterparts in transfer learning scenarios [22–24, 157].

As the field evolves, researchers are actively working on developing more efficient and adaptable transfer learning techniques that can overcome transfer learning limitations, such as continual learning algorithms that can adapt to changing data distributions without forgetting previously learned information. Various strategies have been proposed to mitigate catastrophic forgetting, including regularization-based methods, memory-based approaches, and architectural solutions [27]. These techniques aim to preserve knowledge about previous tasks while learning new ones, enabling models to accumulate knowledge over time. Despite these improvements, challenges remain in scaling continual learning to complex high-dimensional problems, multi-modal domains, and in achieving a balance between stability and plasticity in neural networks.

In conclusion, significant progress has been made in both continual learning and self-supervised learning, addressing key challenges in modern machine learning. Self-supervised techniques have drastically reduced the need for labeled data, while continual learning methods have improved the ability to adapt models to new information without catastrophic forgetting. However, the intersection of these two promising fields has received relatively little attention from the research community. Therefore, this thesis aims to explore the synergies between self-supervised and continual learning, proposing novel approaches that combine the data efficiency of self-supervised methods with the adaptability of continual learning algorithms. By integrating these two paradigms, we seek to develop more robust, efficient, and adaptable AI systems capable of learning continuously from unlabeled data streams, potentially opening new avenues for autonomous learning in dynamic real-world environments.

1.1 Research Context

Here I will shortly elaborate on the two main research topics of this thesis, namely continual learning and self-supervised learning.

1.1.1 Continual Learning

Continual learning (CL) relaxes the IID assumption that underlies most learning methods and studies the design of algorithms that learn from data with shifting distributions. This assumption has a significant impact on learning dynamics, as naively training a machine learning model on such data leads to catastrophic forgetting [99]. Continual learning has seen a notable increase in research [27, 96], especially due to its importance for many applications such as: 1) Systems with memory restrictions that cannot store all previously seen data and must learn incrementally (edge devices or robotics); 2) Applications with data security/privacy restrictions that prevent long-term storage of data, a core requirement current European laws (GDPR [40]); 3) Sustainable artificial intelligence, where retraining large models on all data is computationally expensive and has a large carbon footprint. Incremental learning provides more efficient algorithms that only require processing new data when updating the system; 4) Personal digital assistants that continuously learn from user interactions to improve their responses; 5) Incorporate new styles/concepts to the current growing topic of generative AI; and 6) Recommender systems: Online platforms for e-commerce, streaming services, or social media that use continual learning to update their recommendation algorithms based on evolving user interests and new content.

Continual learning can be categorized into task or class-incremental[*], and offline or online continual learning, depending on factors such as data availability, metadata, computational budget, and the nature of the task (see Fig. 1.2). Task-Incremental Learning (TIL) involves learning a series of distinct tasks sequentially, with clear boundaries between tasks and task identities provided at both training

[*]Some works consider the domain-incremental scenario, where the set of classes is shared among the tasks and the input data distribution is shifting [143]. In this thesis, this scenario is not considered.

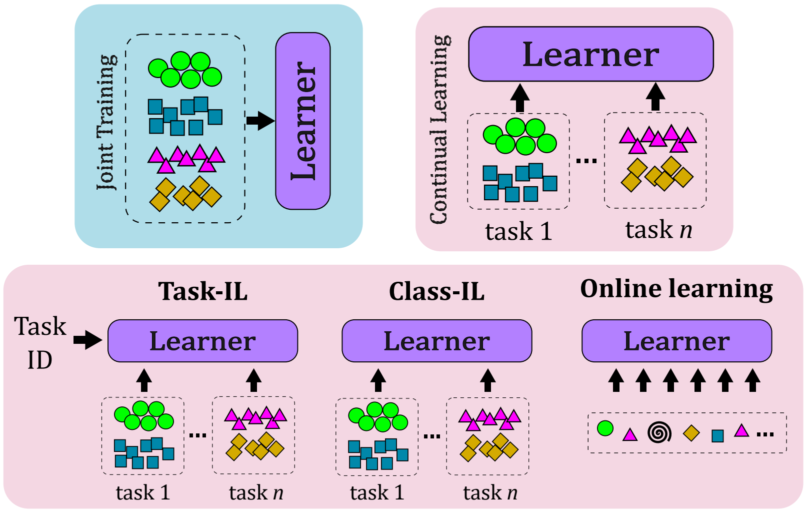


Figure 1.2: Joint training (blue square) vs continual learning (red square). Continual learning studies the design of algorithms that learn from data with shifting distributions. CL algorithms can be categorized into task-incremental, class-incremental, and online learning.

and testing time. Class-incremental learning (CIL) focuses on gradually introducing new classes to a model while maintaining performance on previously learned classes; no task identities are provided at inference. This setting is more challenging and realistic, as it requires the model to perform task-agnostic inference. Online continual learning represents a scenario where the model must learn from a continuous stream of data in real-time, adapting to new information. The challenge is that the algorithm is not allowed to revisit data multiple times. This setting closely mimics real-world applications where data arrives sequentially and must be processed immediately.

CL methods can be further divided into replay-based, architecture-based, and regularization-based methods:

Replay-based methods aim to mitigate catastrophic forgetting by storing and revisiting data from previously learned tasks [20, 122]. These approaches maintain a memory buffer of past experiences, which are then interleaved with new data during training. This allows the model to retain knowledge of earlier tasks while learning new ones. Replay can be implemented through various techniques, such as reservoir sampling to select representative examples, generative models to synthesize past data, or episodic memory systems. Some common replay-based methods experience replay [124], gradient episodic memory (GEM) [20], and iCaRL [122].

Architecture-based methods focus on modifying the structure of neural networks to accommodate new tasks while preserving knowledge of previously learned ones [127, 130]. These approaches typically involve dynamically expanding the network, allocating new neurons or layers for each new task, or creating task-specific pathways within a larger network. Some common techniques include progressive neural networks [127], which add new columns for each task while maintaining lateral connections to previous knowledge, and dynamically expandable networks that grow in capacity as needed [155]. Other methods involve selective parameter freezing or gating mechanisms to protect significant weights for old tasks [135].

Regularization-based methods aim to prevent catastrophic forgetting by constraining the updates to model parameters when learning new tasks. These approaches typically add regularization terms to the loss function that penalize significant changes to parameters deemed crucial for previously learned tasks. Learning Without Forgetting [83] preserves knowledge of previous tasks through knowledge distillation while learning new tasks. Another prominent example is Elastic Weight Consolidation [73], which uses Fisher information to estimate parameter importance and slow down updates to critical weights. Other techniques include Synaptic Intelligence [158], which tracks the contribution of each parameter to the task performance, and Memory Aware Synapses [2], which estimates parameter importance based on how much they affect the model's output.

The vast majority of work on continual learning has focused on the supervised learning scenario [27]. However, recent years have witnessed several breakthroughs in self-supervised learning, which enables the learning of high-quality feature representations from unlabeled data. In the following section, we will elaborate on recent advances in self-supervised learning.

1.1.2 Self-supervised Learning

Self-supervised learning aims to learn high-quality image representations without the need for human annotations. To do so, a pretext task is used; that is, a proxy task that will help the model learn features from the input data without explicit supervision. Many pretext tasks were investigated for learning image representations, including rotation prediction [50], solving jigsaw puzzles [106], determining relative patch positions [32], predicting surrogate classes [34], and image colorization [163]). The most successful pretext tasks in the last few years are based on data augmentation and contrastive-like learning in which two samples are considered either similar or different from each other.

Most recent self-supervised learning (SSL) methods can be categorized into four families based on how they manage image pairs [48]: deep metric learning, self-distillation, canonical correlation analysis, and masked image modeling (see Fig. 1.3).

The deep metric learning family aims to push similar samples together while pushing dissimilar ones apart in the embedding space. This approach has roots in earlier contrastive methods used in metric learning [59] and some extensions using triplet losses [146]. In SimCLR [22], similar samples are created by applying random distortions to an input image, while dissimilar ones are chosen randomly. NNCLR [37] extends this approach by using nearest neighbors from a support set as additional positives, enabling the model to learn from semantically similar samples beyond mere augmentations of the same image. This increases the diversity of positive examples and helps the model learn more robust and generalizable representations.

Self-distillation methods feed two different views of an image into two encoders and map one onto another using a predictor. This technique is prone to collapse to a constant solution for any input, so several strategies are employed to prevent this phenomenon. The BYOL approach proposed by [57] uses an asymmetric network with an additional MLP predictor between the outputs of two branches. One branch is kept "offline" and updated by a momentum encoder. SimSiam [24] offers a simplified solution without a momentum encoder and works well without requiring a very large mini-batch size. The MoCo method [23, 61] uses a memory bank for learned embeddings, enabling efficient sampling. This memory is kept synchronized with the rest of the network during training using a momentum encoder. DINO [16] employs a centering operation in the output of one of the encoders and discretizes the representations using a softmax. iBOT [166] builds on DINO but adds a masked image modeling objective in the latent space. DINOv2 [107] improves the iBOT baseline by incorporating more regularization into the training recipe and curating the pretraining dataset.

The canonical correlation analysis approach aims to infer the relationship between two variables by analyzing their cross-covariance matrices [64]. Barlow Twins [157] uses a loss function based on correlations between each pair in a current training mini-batch, with negatives implicitly assumed to be available in each mini-batch. VicReg [9] balances three objectives based on the cross-correlation matrix: variance, invariance, and covariance. Other methods in this family include SwAV [15] and W-MSE [38].

Finally, the masked image modeling (MIM) family masks out a portion of an image and trains a model to reconstruct it. This technique is inspired by the masked language modeling paradigm commonly used in language models such as BERT [30].

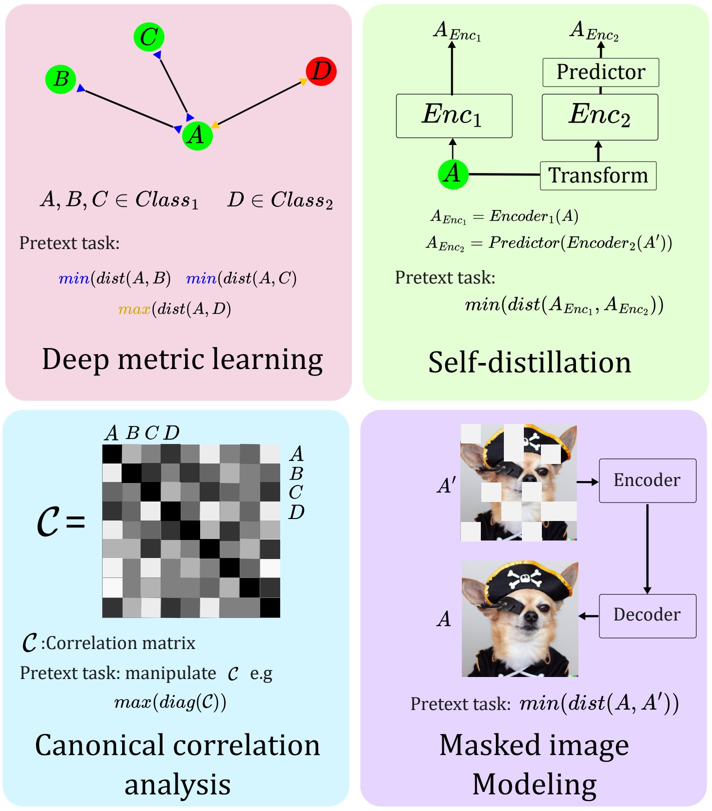

C
B
D
A
$A, B, C \in Class_1 \quad D \in Class_2$
Pretext task:
$min(dist(A,B) \quad min(dist(A,C)$
$max(dist(A,D)$
Deep metric learning
A_{Enc_1}
A_{Enc_2}
Predictor
Enc_1
Enc_2
A
Transform
$A_{Enc_1} = Encoder_1(A)$
$A_{Enc_2} = Predictor(Encoder_2(A'))$
Pretext task:
$min(dist(A_{Enc_1}, A_{Enc_2}))$
Self-distillation
$A\ B\ C\ D$
$\mathcal{C} =$
A
B
C
D
$\mathcal{C}$:Correlation matrix
Pretext task: manipulate $\mathcal{C}$ e.g
$max(diag(\mathcal{C}))$
Canonical correlation analysis
A'
Encoder
A
Decoder
Pretext task: $min(dist(A, A'))$
Masked image Modeling

Figure 1.3: Self-supervised learning families. Deep metric learning (red) pushes similar samples together and dissimilar ones apart. Self-distillation (green) maps two different views using a predictor. Canonical correlation (blue) infers the relationships using cross-covariance matrices. Masked image modeling (purple) reconstructs a masked image

MAE [60] and SimMIM [153] directly reconstruct masked image patches (instead of tokens) extracted from a vision transformer encoder. Although MIM is a relatively new approach, it has already achieved competitive performance across a wide variety of vision tasks [68, 142]. Furthermore, it has been combined with other SSL families to produce variations such as iBOT [166] and DINOv2 [107].

The advantages of self-supervised learning extend beyond its initial applications, particularly in the realm of continual learning. Unsupervised learning tends to lead to more generalizable feature representations since features that are not relevant to the specific discriminative task are not automatically discarded. This can potentially lead to more general representations that can quickly incorporate new tasks without incurring significant amounts of forgetting. Consequently, the integration of self-supervised techniques into continual learning presents an exciting and promising path forward, offering the potential to create AI systems that can learn and adapt more efficiently.

1.2 Challenges in Continual Learning and Self-Supervised Learning of Deep Feature Representations

Here we elaborate the main challenges which we have identified with respect to the self-supervised continual learning of feature representations.

1.2.1 Continual unsupervised representation learning

Feature representation learning aims to learn high-quality representations of the input signal. This learning process can be implemented using labels (supervised learning), reward signals (reinforcement learning), or only input signal information (unsupervised learning). Most of the data available have few labels, making the use of supervised or reinforcement paradigms impossible or really expensive (paying a data labeling service). On the contrary, unsupervised learning exploits large quantities of unlabeled data to learn features that can be fine-tuned using few labels or be used by a linear classifier as general representations.

Also, continual learning can be applied in a supervised and unsupervised manner; nevertheless, most of the CL approaches follow the supervised paradigm [96]. Unsupervised CL presents useful applications in real-world settings, where it is expected to have high quantities of unlabeled data. Recent works have revolutionized unsupervised learning with so-called self-supervised learning algorithms. These methods obtain representations that rival results obtained from supervised approaches, making use of image augmentation and contrastive learning techniques (see Fig. 1.4).

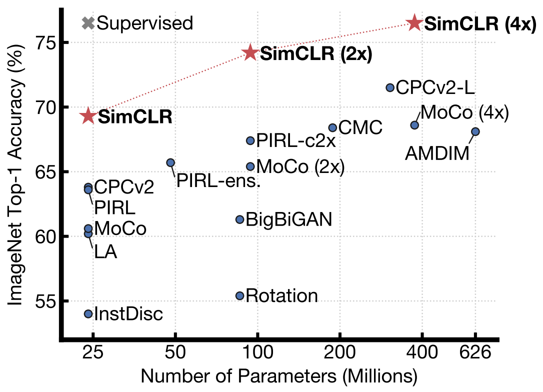


Figure 1.4: ImageNet top-1 accuracy comparing supervised and self-supervised (SimCLR) methods (image from [22]).

Self-supervised learning methods have in common that they assume that all training data is available during the training process. However, in many real-world applications, the learner must cope with non-stationary data in which they are exposed to tasks with varying distributions of data. *Therefore, in this thesis, we aim to extend self-supervised learning theory to continual learning paradigm.*

1.2.2 Improving the Plasticity-Stability trade-off in continual unsupervised representation learning

The Plasticity-Stability trade-off (see Fig. 1.5) is a fundamental problem in any continual learning system [103]. A model must address the trade-off between acquiring new knowledge (plasticity) and preventing the forgetting of previous knowledge (stability); each proposed continual learning method continuously addresses this dilemma.

Unsupervised representations tend to be more general, not specific to a certain discriminative task. This fact can lead to representations that can easily incorporate new knowledge (plastic) without incurring a lot of forgetting (stable). If samples (exemplars) can be stored during the CL process, general representations can be maintained over time without overspecializing, leading to an excellent plasticity-stability trade-off (as previous CURL methods have shown [88]).

In the exemplar-free landscape, there have been two proposed approaches. PFR (our paper [53]) uses a projection head after the feature extractor of an SSL framework to predict past representations. Projected representations are motivated to be

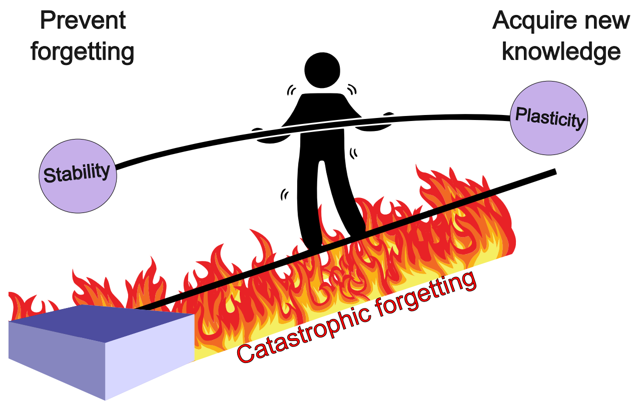


Figure 1.5: The Plasticity-Stability trade-off implies that a model must address the trade-off between acquiring new knowledge (plasticity) and preventing the forgetting of previous knowledge (stability). Failure to balance these two properties will lead to catastrophic forgetting.

close to past representations; therefore, the new model is encouraged to remember past knowledge. CaSSLe [43] uses a strategy similar to PFR, but the projection head is used after the projector of the SSL approach. Even though these methods obtain satisfactory results, they struggle to adapt to new tasks (plasticity) without jeopardizing the vast knowledge already accumulated by the network (stability) since both are based on regularization approaches. *Therefore, in this thesis, we will explore how to mitigate the negative effects of regularization in exemplar-free continual self-supervised methods.*

1.2.3 Semantic drift compensation in unsupervised continual learning

Semantic drift (SD) is the shift/displacement of learned representations when a model is updated during the training of new tasks [156]. This drift will cause a considerable amount of forgetting in exemplar-free methods, as the model is not encouraged to retain current knowledge (see Fig. 1.6).

Class Prototype (CP) accumulation has been shown to be among the most successful approaches to exemplar-free continual learning [54, 70, 100, 111]. CP methods store class prototypes computed using training data and then use them to classify using a distance classifier in feature space. Class Prototype methods are attractive since they can be agnostically applied to any feature extractor, such as

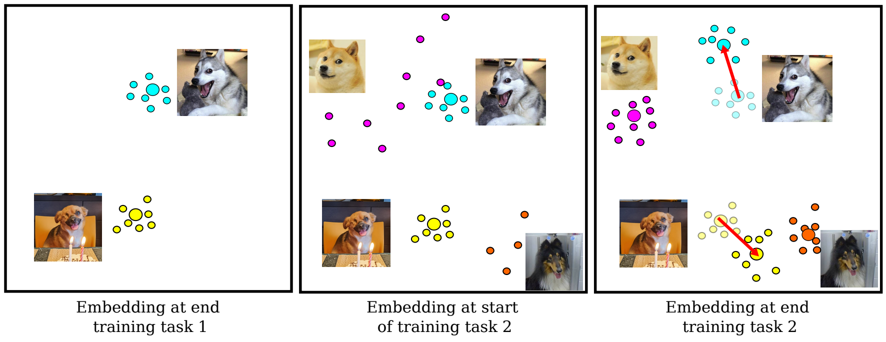

Embedding at end training task 1
Embedding at start of training task 2
Embedding at end training task 2

Figure 1.6: Semantic drift of old classes (cyan, yellow) occurs when training new classes (pink, orange) due to drift of features in the network backbone. (left) The two classes of task 1 have been learned correctly and are nicely clustered in feature space. (middle) The new classes (of task 2) are not clustered with the network trained on task 1. (right) While learning a feature representation that clusters the new classes, the old classes experience semantic drift.

multilayer perceptrons, convolutional neural networks, or transformers.

An approch to mitigating semantic drift is *drift compensation*. Semantic Drift Compensation (SDC) [156] addresses this by computing the drift of features using new task data and approximating the drift of old class prototypes based on this information. Subsequent work [140] extended SDC by also considering feature drift. Memory efficient CIL through feature adaptation (MEA) [69] addresses the problem of feature drift. It stores the features after the backbone and corresponding labels. A learned mapping is applied to translate stored features into the new space to compensate for drift. This method requires labels of features, making it unsuitable for unsupervised and semi-supervised settings. Additionally, they store an extensive amount of features for all old classes, leading to higher memory requirements.

Current approaches to semantic drift compensation require exemplars and labels to work, making them unsuitable for unsupervised and semi-supervised, and exemplar-free settings. *Therefore, in this thesis, we will explore how to correct semantic drift without labels and/or an exemplar buffer.*

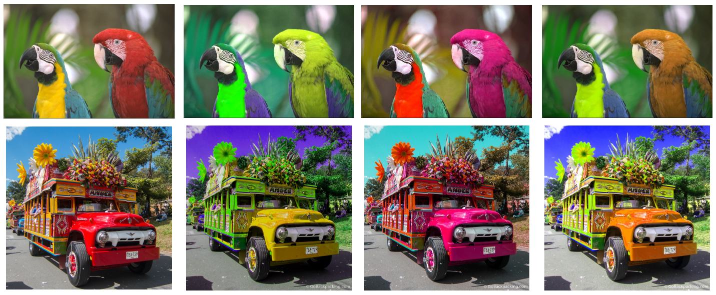


Figure 1.7: Augmentations based on color jitter lead to unrealistic images from which it is difficult to learn good color representations. We present here the default color jitter operation applied to two images showing color out of the distribution (columns 2 to 4) of the real object (first column).

1.2.4 Enhancing color quality for self-upservised representation learning

Image augmentations are a key ingredient in modern machine learning. As contemporary learning algorithms become increasingly data-hungry, the ability to artificially increase the available data allows these massive models to learn feature representations and generalize better over the learning task. For self-supervised learning (SSL), image augmentations are particularly important [13], as this training paradigm aims to learn similarities between an image and its augmented versions.

In SSL, there is a set of "standard" augmentations (proposed in [22]): rotation, cutout, flip, color jitter, blur, and grayscale. Color jitter (CJ) induces a certain level of color invariance (invariance to hue, saturation, brightness, and contrast), which is consequently transferred to the downstream task. This invariance, however, is expected to negatively impact downstream tasks where color is important. For instance, Fig. 1.7 shows how CJ produces unrealistic colors for a flower. Although color invariance has proven beneficial for object recognition, the invariance to hue and saturation changes induced by color jitter can be detrimental to object recognition for classes in which color characteristics are fundamentally discriminative. *Therefore, in this thesis, we will explore how to generate realistic color augmentations for self-supervised learning.*

1.3 Objectives and approach

In this thesis we explore deep feature representations in the context of continual and self-supervised learning. Here, we define our objectives and approach to address the challenges identified in the previous section.

1.3.1 Continual unsupervised representation learning

Current approaches in self-supervised learning rely on the joint training paradigm (all training data is available during the training process). However, in a shifting distribution setting (i.e. a non-IID data stream), current methods will lead to catastrophic forgetting. To address this, we consider the problem of continual unsupervised representation learning, and define the following objective:

> **Regularization of self-supervised learned features:** Propose an exemplar-free regularization-based CL method for the problem of continual unsupervised representation learning. Analyze and review feature knowledge distillation within the context of rehearsal-free class incremental learning.

To address the challenges of continual unsupervised representation learning, we propose a new method, called *projected functional regularization* (PFR), to alleviate forgetting during unsupervised representation learning without the need for exemplars. This technique is an extension of Learning without Forgetting (LwF) [83] and feature distillation. To improve the plasticity of the method, we introduce a temporal projection network that provides more freedom to learn features from the current task.

1.3.2 Plasticity-Stability trade-off in continual unsupervised representation learning

Continual unsupervised representation learning methods rely on regularization methods to avoid forgetting in the absence of exemplars [43]. As a consequence, these methods limit the plasticity of the model and risk forgetting the accumulated knowledge, we therefore define the following objective:

> **Plasticity-optimized networks :** Develop a multi-network method capable of managing plasticity and stability. Study complementary learning systems applied to continual learning

To improve the plasticity-stability trafe-off in continual unsupervised representation learning, we apply complementary learning systems theory to improve continual learning from unsupervised data streams. we propose to train an expert network that is relieved of the task of keeping the previous knowledge and can focus on the task of performing optimally on new tasks. In the second phase, we combine this new knowledge with the old network in an adaptation-retrospection phase to avoid forgetting.

1.3.3 Semantic drift in unsupervised continual learning

Drift compensation methods in continual learning rely on exemplars and labels to retain the knowledge of previous classes. However, these two conditions constrain the application of drift compensation to supervised exemplar-based CL, we therefore, define the following objective:

> **Class-prototype mapping between tasks:** We propose a supervision-free method to map class prototypes between consecutive tasks. Furthermore, we analyze and review class prototype accumulation strategies for continual learning.

To map class prototypes between consecutive tasks, we propose our method called Learnable Drift Compensation (LDC). LDC maps the previous task features to the current feature space via a forward projector network. This compensation can be learned from only the stored old class prototypes and does not require class labels.

1.3.4 Color-crippling effects of color jitter on representation learning

Color jitter is a core image augmentation strategy in most self-supervised learning methods. Nevertheless, the induced color invariance effect of this augmentation is expected to negatively impact tasks where color is important. We therefore define the following objective:

> **Realistic color augmentations:** Develop a color augmentation strategy that follows natural color variations. Evaluate the impact of the proposed realistic color augmentation in mainstream self-supervised learning methods.

In order to produce realistic color augmentations, we draw on the existing color imaging literature on designing features invariant to illuminant changes commonly encountered in the real world. Our augmentation, called Planckian Jitter, applies physically-realistic illuminant variations. We consider the illuminants described by Planck's Law for black-body radiation, which are known to be similar to illuminants encountered in real-life.

2 Continually Learning Self-Supervised Representations with Projected Functional Regularization*

2.1 Introduction

Self-supervised learning aims to learn high-quality image representations without the need for human annotations. A recent set of works has shown that self-supervised learning can achieve performance close to that of supervised learning [15, 22, 24, 57], and that learned representations transferred to downstream tasks are sometimes even superior to fully-supervised representation learning [16]. These methods learn representations that are invariant with respect to a set of data augmentations. They are typically trained with contrastive losses in which multiple views of the same image (computed by applying different data augmentations) are mapped close together, whereas representations of other images are mapped far away. However, several methods show that only encouraging similarity between views from the same image (without any explicit loss to promote the distancing of negative pairs) can also obtain excellent performance [24, 57]. These methods apply various mechanisms to prevent trivial solutions, including asymmetric architectures and the use of momentum updates of the model.

Recent works on self-supervised learning have in common that they assume that all training data is available during the training process. However, in many real-world applications the learner must cope with non-stationary data in which they are exposed to tasks with varying distributions of data. Continual learning relaxes the IID assumption that underlies most learning methods and studies the design of algorithms that learn from data with shifting distributions. Naively training a learner on such data, for example by simply continuing stochastic gradient descent, leads to catastrophic forgetting [99]. A variety of approaches have been proposed including various types of regularization [2, 73, 83, 158], data replay [17, 66, 122, 149], pseudo replay [134, 148], and growing architectures [127]. Even though there is some work on unsupervised continual learning [42, 82, 94, 127], the vast majority of existing work is on supervised continual learning [112, 116].

*This chapter is based on a publication in the 3rd CLVISION workshop in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition 2022 [53]

Earlier works on self-supervised learning was based on pretext tasks like predicting rotation [50], determining patch position [32], or solving jigsaw puzzles in images [106]. Labels for these discriminative pretext tasks can be automatically computed and allow learning of meaningful feature representations of images. Recently, researchers are adapting contrastive methods for unlabeled data and operating more at an instance-level augmentation while looking for similarity or contrastive samples [15, 22, 57, 157]. These methods rely heavily on stochastic data augmentation to produce enough similar examples to learn representations. Negative examples are randomly sampled or not used at all [24]. The results are impressive and are competitive with many supervised methods on downstream tasks [16].

We propose an approach to *continual* self-supervised learning that is able to learn high-quality visual feature representations from non-IID data. The learner is exposed to a changing distribution and, while learning new features on current task data, should prevent forgetting of previously acquired knowledge. These representations should, at the end of training, be applicable to a wide range of downstream tasks. We focus on the more restrictive, memory-free continual learning setting in which the learner is not allowed to store any samples from previous tasks. This scenario is realistic in many scenarios where data privacy and security is fundamental and often legislatively regulated.

The main contributions of this work are twofold. First, we propose a new method, called *projected functional regularization*, to alleviate forgetting during unsupervised representation learning without the need for an external memory of samples from previous tasks. This technique is an extension of Learning without Forgetting (LwF) and distillation in feature space. To improve the plasticity of the method we introduce a *temporal projection network* that provides more freedom to learn features from the current task. Secondly, we propose a set of experiments over benchmark datasets to compare with other state-of-the-art methods and use different scenarios to evaluate the functional projection role in the context of continual self-supervised representation learning. We show that the additional projection to past tasks results in better representation learning during class incremental training sessions. Without any adjustment, evaluation on a truly class incremental scenario – with only a single class per task, where many class incremental methods cannot be directly applied – our method still prevents forgetting and is able to progressively learn new features. Furthermore, we confirm that our method is generic and the results are not restricted to a particular self-supervised learning approach. In a variety of experimental settings the transferability of the learned features to different downstream tasks is maintained, confirming that the network is incrementally learning more robust representations. The influence of the regularization strength is analyzed for different regularization methods applied to self-supervised continual learning and results clearly shows the benefit of the proposed additional projector resulting both in improved plasticity (i.e. lower intransigence) and less forgetting.

2.2 Related Work

Both self-supervised and continual learning have gathered increasing interest in recent years. We briefly review the literature on both topics before articulating our contribution which combines elements of both in the form of continual self-supervised representation learning.

Self-supervised learning. Self-supervised learning has proved useful for many applications. In order to learn representations useful for a downstream task, a self-supervised pretext task can be introduced to avoid supervision. Many pretext tasks were investigated for learning image representations, including rotation prediction [50], solving jigsaw puzzles [106], determining relative patch positions [32], predicting surrogate classes [34], and image colorization [163]).

In the last few years, the gap between supervised and self-supervised learning is being closed. This is primarily due to methods based on data augmentation and contrastive-like learning in which two samples are considered either similar or different to each other. This has links to earlier contrastive methods used in metric learning [59] and some extensions using triplet losses [146]. However, in the unsupervised setting without labels, different approaches must be used for creating such pairs. In SimCLR [22], similar samples are created by augmenting an input image with a random distortion, while dissimilar ones are chosen by random. To make contrastive training more efficient, the MoCo method [23, 61] uses a memory bank for learned embeddings which enables efficient sampling. This memory is kept synchronized with the rest of the network during training by using a momentum encoder. The SwAV approach uses online clustering over the embedded samples [15]. SwAV does not sample negative exemplars, however, other cluster prototypes can play the role of negative examples.

Interesting are methods without any explicit contrastive pairs. The BYOL approach proposed by [57] is based on an asymmetric network with an additional MLP predictor between two outputs of the two branches. One branch is kept "offline" and updated by a momentum encoder. SimSiam [24] goes even further and offers a simplified solution without a momentum encoder and moreover works well without a very large mini-batch size. BarlowTwins is another simplified solution like SimSiam which uses a loss function based on correlations between each pair in a current training mini-batch [157]. Negatives are implicitly assumed to be available in each mini-batch. No asymmetry is used by the BarlowTwins network, but a larger embedding size and bigger mini-batches are preferred in this method in comparison to SimSiam.

Continual learning. Existing continual learning methods can be broadly divided into replay-based, architecture-based, and regularization-based methods [27, 96]. Replay-based methods save a small amount of data from previously seen tasks [8,21] or generate synthetic data with a generative model [145, 160]. Architecture-based method activate different subsets of network parameters for different tasks by allowing model parameters to grow linearly with the number of tasks. Previous works following this strategy include DER [154], Piggyback [93], PackNet [94], DAN [125], HAT [131], Ternary Masks [97] and PathNet [42]. Regularization-based methods add an additional regularization term derived from knowledge of previous tasks to the training loss. This can be done by either regularizing the weight space (constraining important parameters) [133, 137] or the functional space (constraining predictions or intermediate features) [25, 35, 67]. EWC [73], MAS [2], REWC [84], SI [158], and RWalk [19] constrain the importance of network parameters to prevent forgetting. Methods such as LwF [83], LwM [31] and BiC [149] instead leverage knowledge distillation to regularize features or predictions.

Our approach, called *Projected Functional Regularization* is a functional regularization approach. Normally, these approaches distill information at the class-prediction level between an old and new model. However, in self-supervised learning this has to be applied to the embedding output. Regularizing the embedding layer is known to undermine plasticity [35], we therefore propose an additional temporal projection network that maps between the latent spaces of current and previous model. We show that this regularization prevents forgetting while obtaining improved plasticity.

Continual representation learning. Continual unsupervised representation learning was investigated by [121] with an approach based on variational autoencoders and a Gaussian mixture model. Still, detection of new clusters and model expansion is necessary. In a very recent paper, the authors used contrastive self-supervised learning with a memory buffer for storing exemplars [88]. They proposed the LUMP method in which images from the current task and previous tasks are combined with CutMix for continual training. One of the main differences between LUMP and our approach is that ours does not require storing any data from previous tasks.

Our contribution is fundamentally different from methods using self-supervised learning to improve the learning of a sequence of supervised tasks [58, 168]. Their objective is not to learn from unlabeled data, but rather to use self-supervised learning to further enrich the feature representation. The hypothesis of these works is that, for class incremental learning scenario, the features learned via self-supervision will be more generic than ones learned from task-bounded discrimination problems.

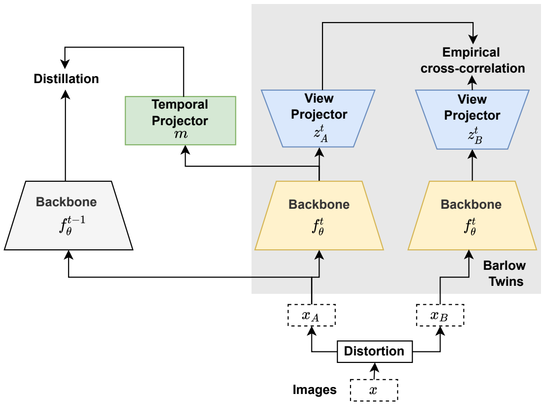


Figure 2.1: Self-supervised continual learning with Projected Functional Regularization. Instead of performing feature distillation directly between the previous task backbone and the new one, we use a *learned temporal projection* between the two feature spaces.

2.3 Continual Self-supervised Representation Learning

We begin with a discussion of self-supervised representation learning, and then describe our proposed Projected Functional Regularization (PFR) approach for continual learning of self-supervised representations without the need of any memory or replay.

2.3.1 Self-supervised representation learning

In recent works on self-supervised learning the aim is to learn a network $f_\theta : \mathscr{X} \rightarrow \mathscr{F}$ that maps from input space $\mathscr{X}$ to output feature representation space $\mathscr{F}$. This network is learned on unlabeled input data x drawn from distribution $\mathscr{D}$. The aim is then to exploit the learned feature representation to perform any variety of downstream tasks. As an example, for the downstream task of classification on some target domain, we have training data $\mathscr{D}^t = \{x_i^t, y_i^t\}$ on which we learn a classifier $g_\phi : \mathscr{F} \rightarrow \mathscr{Y}$ (with $\mathscr{Y}$ being the output space) that minimizes a loss $\mathscr{L} = \ell\left(y^t, \hat{y}^t = g_\phi\left(f_\theta\left(x^t\right)\right)\right)$. Adaptation to the target domain might only optimize the weights ϕ while keeping θ fixed on the target data, or instead might also allow θ

to be fine-tuned on the target data.

In this chapter we apply the BarlowTwins [157] approach to self-supervised learning of the representation network f_θ. However, the proposed method is general and can be applied to other self-supervised methods. BarlowTwins does not require explicit negative samples and achieves competitive performance while remaining computationally efficient, assuming that negatives are available in each mini-batch to calculate correlation between all samples in it. The BarlowTwins architecture has two branches (see the shaded area in Fig. 2.1). In both branches a projector network $z : \mathscr{F} \rightarrow \mathscr{Z}$ is used. For the sake of notational simplicity, we do not make explicit the parameters of the network z since it is not used by downstream tasks. The parameters in the backbone and projector layer are shared between the branches.

The network is trained by minimizing an invariance and a redundancy reduction term in the loss function [157]. Here, different augmented views X_A and X_B of the same data samples X are taken from the set of data augmentations $\mathscr{D}^*$. This leads to the loss defined as:

$$\mathscr{L}_c = \mathbb{E}_{X_A, X_B \sim \mathscr{D}^*}\left[\sum_i (1 - \mathscr{C}_{ii})^2 + \lambda \sum_i \sum_{j \neq i} {\mathscr{C}_{ij}}^2\right], \tag{2.1}$$

where λ is a positive constant trade-off parameter between both terms, and where $\mathscr{C}$ is the cross-correlation matrix computed between the representations z of all samples X_A and X_B in a mini-batch indexed by b:

$$\mathscr{C}_{ij} = \frac{\sum_b z^A_{b,i} z^B_{b,j}}{\sqrt{\sum_b (z^A_{b,i})^2}\sqrt{\sum_b (z^B_{b,j})^2}}, \tag{2.2}$$

The cross-correlation matrix $\mathscr{C}$ has values in the range of -1.0 (worst) to 1.0 (best) correlation between the projector's outputs: $Z_A = z(f_\theta(X_A))$ and $Z_B = z(f_\theta(X_B))$. The invariance term of the loss function encourages the diagonal elements to be 1. As such the learned embedding will be invariant to the applied data augmentations. At the same time, the second term (redundancy reduction) keeps the off-diagonal elements close to zero and decorrelates the outputs of non-related images.

2.3.2 Projected Functional Regularization

Current work on self-supervised learning considers the above scenario where the learner has access to a single, large dataset which can be revisited multiple times to learn the optimal feature extractor f_θ. However, for many real-world scenarios this is an unrealistic setup and the learner will have to learn the optimal feature extractor f_θ from a stream of data drawn from a distribution that varies over time.

We consider scenarios in which the learner must learn from a set of tasks, each containing data drawn from a different distribution. We consider the tasks $T = \{1...c\}$ where c is the current task and the data of task t follows the distributions $\mathscr{D}_t$. In this case we would like to find the parameters θ of the feature extractor f_θ that minimize the summed loss over all tasks up to the current one c:

$$\arg\min_\theta \sum_{t=1}^{c} \mathscr{L}_c^t, \tag{2.3}$$

where $\mathscr{L}_c^t = \mathbb{E}_{X_A, X_B \sim \mathscr{D}_t^*}[\mathscr{L}_c]$ and $\mathscr{L}_c$ is defined as in Eq. 2.1. Again, $\mathscr{D}_t^*$ refers to the set of augmented samples from $\mathscr{D}_t$ (i.e. the data from task t). However, during continual training we only have access to the data of one task, meaning that the optimal parameters must be found using only the current data $\mathscr{D}_c$. Naive fine-tuning results in parameters optimal for task c, however leads to catastrophic forgetting of knowledge acquired during previous tasks.

Regularization methods are among the most successful at addressing catastrophic forgetting, especially for scenarios where storing of any data from previous tasks is prohibited (which is the objective in this chapter). Regularization methods can be divided into two important groups: weight regularization approaches [2, 73, 158], which aim to find a set of weights that is both good for the current task while incurring only a small increase in loss on previous tasks, and functional regularization methods (also known as data regularization methods) which optimize weights for new tasks while incurring only minimal changes in the network outputs on previous tasks [65, 110, 139].

The canonical example of functional regularization, called Learning without Forgetting (LwF), was introduced in [65] and is based on knowledge distillation [63]. It was proposed for supervised continual learning and introduces an additional loss that prevents the class predictions of previous tasks on the current data from undergoing large changes while training on the current task data. This loss cannot be directly applied to self-supervised learning since it requires class predictions. However, several continual learning works have extended this idea to feature layers by replacing the modified cross-entropy distillation loss with a distance (typically L1 or L2) which can be applied to any layer output [35, 85, 156]. We will refer to this as *feature distillation* (FD) and it leads to the following loss when training task t:

$$\mathscr{L}_c^t + \lambda_{fd} \mathbb{E}_{x_a, x_b \sim \mathscr{D}_i^*}[\| f_{\theta^t}(x_a) - f_{\theta^{t-1}}(x_a) \| + \| f_{\theta^t}(x_b) - f_{\theta^{t-1}}(x_b) \|], \tag{2.4}$$

where θ^{t-1} refers to the parameters learned after training up to task $t-1$, and λ_{fd} defines the importance of the regularization term.

The regularization imposed on class predictions in the original LwF paper [65] is not very restrictive: the weights can still significantly vary as long as the final network predictions do not significantly vary. It has been observed in the literature, however, that feature distillation is very restrictive and leads to continual learning methods with low plasticity [35]. In addition, this loss directly penalizes the learning of new features since these would lead to a difference between the new and old model output $\| f_{\theta^t}(x) - f_{\theta^{t-1}}(x) \|$. To address this problem we propose *Projected Functional Regularization* (PFR).

We would like the network to retain previous feature representation while allowing it to learn new features learned on new tasks. These new features should not be directly penalized by regularization. To do so, we introduce a *temporal projection network* $m : \mathcal{Z} \rightarrow \mathcal{Z}$ that maps the embedding learned on the current task back to the embedding learned on the previous ones (see Figure 2.1). The new loss is:

$$\mathcal{L}_c^t + \lambda_{pfr} \mathbb{E}_{x_a, x_b \sim \mathcal{D}_i^*} [\mathcal{S}(m(f_{\theta^t}(x_a)), f_{\theta^{t-1}}(x_a)) + \mathcal{S}(m(f_{\theta^t}(x_b)), f_{\theta^{t-1}}(x_b))] \tag{2.5}$$

where $\mathcal{S}(\cdot, \cdot)$ is a cosine similarity:

$$\mathcal{S}(a, b) = -\frac{a^T b}{||a||_2 ||b||_2}. \tag{2.6}$$

New features learned in $f_{\theta^t}(x)$ do not directly result in an increased loss as long as they lie in the null-space of m. As a consequence this loss prevents forgetting of information of previous tasks while maintaining plasticity to adapt to new tasks.

2.4 Experimental Results

Here we report on a variety of experiments performed to evaluate the performance of Projected Functional Regularization for continual self-supervised representation learning without the need of an exemplar memory and replay.

2.4.1 Datasets

We use the following datasets in our evaluation:

- **CIFAR-100**: Proposed by [76], this dataset consists 100 object classes in 45,000 images for training, 5,000 for validation, and 10,000 for test with 100 classes. All images are 32×32 pixels.

- **Tiny ImageNet**: A rescaled subset of 200 ImageNet [29] classes used in [136] and containing 64×64 pixel images. Each class has 500 training images, 50 validation images and 50 test images.

- **SVHN**: contains 32×32 pixel images of from house numbers. There are 10 classes with 73,257 training images and 26,032 test images. From we split 5% of the training images to use as a validation set.
- **Cars**: Was introduced in [75]. contains 16,185 images of 196 cars classes which includes 8,144 as train set and 8,041 as test set.
- **Aircraft**: Was proposed in [91] and consists 6,667 images for training and 3,333 for testing of 100 classes.

The last three datasets are used for evaluating our proposed method on downstream tasks. We downscale images to 64×64 for Cars and Aircraft in our experiments.

2.4.2 Training procedure and baseline methods

In all experiments, we train a ResNet-18 [62] using SGD with an initial learning rate of 0.06 and a weight decay of 0.0001. The network is trained with cosine annealing for the first 1500 epochs. After these epochs of cosine annealing, the learning rate is reduced by a factor of 0.8 for the both projectors (view and temporal) and 0.4 for the backbone. The data augmentation process is the same as in BarlowTwins (which was taken from SimCLR [22]). As a temporal projector, we use an MLP with a linear layer of 512 neurons followed by a batch normalization, Relu and a second linear layer of 256 neurons for CIFAR-100 and a linear layer with 512 neurons followed by a ReLu for TinyImageNet.

Downstream task classifiers are by default linear with a cross-entropy loss and are trained with Adam optimizer with a learning rate $5e$-3 for CIFAR-100 and $5e$-2 for TinyImageNet. We use validation data to implement a patience scheme that lowers the learning rate by a factor of 0.3 up to three times while training a downstream task classifier.

In our experiments we compare with the following baseline methods:

- **Fine-tuning (FT)**: The network is trained sequentially on each task without access to previous data and with no mitigation of catastrophic forgetting.
- **Single Task**: We perform joint training with fine-tuning on all data which provides an upper bound.
- **Continual Joint Training (CJ)**: We continually perform joint training on the entire dataset seen so far. This is provides a tighter upper bound than Single Task [6].

- **Feature Distillation (FD)**: Knowledge distillation is used as in LwF [83] to retain representation from previous tasks. We use the L2 distance as the regularization term, as is also proposed by other methods performing knowledge distillation on feature embeddings [35, 85, 156].
- **Elastic Weight Consolidation (EWC)**: We use the regularization method from [73] with a contrastive loss used to estimate the diagonal of the Fisher Information Matrix.

Note that we only compare to exemplar-free methods and exclude methods that require replay from our comparison. [†]

2.4.3 Continual representation learning

In this experiment we evaluate all methods in the incremental representation learning setting. The most straightforward way of doing this is to use the class incremental learning setting without access to labels. Specifically, we split datasets into four equal task as done in [122]. In each task we learn a self-supervised representation and in the evaluation phase we train a linear classifier using the trained backbone encoder. In order to assess the learned representation, we use all available test data to obtain the overall task-agnostic performance evaluation[‡]

In Table 2.1 we present the results for all methods. After the final task, the upper bound CJ obtains 60.6%, while a simple fine-tuning (FT) method reaches 56.8%. This is the gap where methods using regularization can improve. Joint training on all data at once outperforms CJ by 2.4%. PFR obtains an accuracy after the final task of 60.1%, while other regularization methods FD and EWC reach 57.2% and 55.8%, respectively.

In addition, we show the task-aware results on CIFAR 100 of the incrementally learned representations in Figure 2.2. Here the models are the same as those used in Table 2.1. Note that all other results in the chapter are task-agnostic (no task-ID given during inference). Here, we also use training data from future tasks to train the classifier: this allows us to also evaluate the performance of the tasks that have not been seen by the feature extractor (see above diagonal elements). We observe that all methods, including FT, incrementally improve results. Only our proposed PFR method considerably outperforms FT in this setting. It is also interesting to observe that PFR obtains positive backward transfer, since the performance on task 1 and 2 improves during the consecutive training sessions.

[†] code available at https://github.com/alviur/CVPR_PFR.git

[‡] Note that we use *task agnostic* in this chapter to refer to the class-incremental learning evaluation [96].

Table 2.1: (top) Accuracy on CIFAR-100 with 4 tasks of 25 classes for incremental, self-supervised training. The learned representation is evaluated using a linear classifier over all classes after each task. (bottom) Evaluation of the same trained models on a different downstream task - classification using SVHN dataset. Mean and standard deviation over five runs are provided.

CIFAR100

Method	**Task 1**	**Task 2**	**Task 3**	**Task 4**
Single	-	-	-	65.4±1.4
CJ	53.3±0.7	58.4±0.8	60.8±1.1	63.6±1.5
FD	53.3±0.4	55.6±0.5	56.8±1.0	57.8±0.5
EWC	53.0±0.3	53.1±0.3	53.8±0.6	55.0±0.4
PFR	53.2±0.4	56.4±0.4	58.2±0.3	**59.7**±0.3
FT	53.4±0.5	53.0±0.7	54.6±0.2	54.8±1.0

SVHN

Method	**Task 1**	**Task 2**	**Task 3**	**Task 4**
Single	-	-	-	64.0±1.4
CJ	60.6±2.6	63.2±1.7	63.4±1.3	63.0±1.3
FD	60.4±2.7	62.7±2.2	64.3±1.7	65.8±1.5
EWC	61.3±2.3	62.4±1.8	64.1±1.2	64.5±1.5
PFR	60.4±2.7	63.5±2.3	66.2±1.9	**68.0**±1.5
FT	60.4±2.7	61.0±1.4	63.8±1.1	64.5±1.1

Learned representations. In addition to evaluating accuracy on downstream classification tasks, we compare learned representation similarity with a Centered Kernel Alignment (CKA) [74]. The results are given in Figure 2.3. When the task is learned and immediately evaluated, the similarity is equal to one. When we finetune the model with new data, we start experiencing representation degradation – seen in decreasing values in the columns in Figure 2.3, left. With PFR, representation forgetting progresses much more slowly. FT is the worst, having the first task similarity after the last one with CKA equal 0.69, next is FD with value 0.72, and the best is PFR with 0.82.

Many task scenario. Here we consider a challenging setting with longer sequences – i.e. with more tasks. We experimented with our PFR method, fine-tuning(FT)

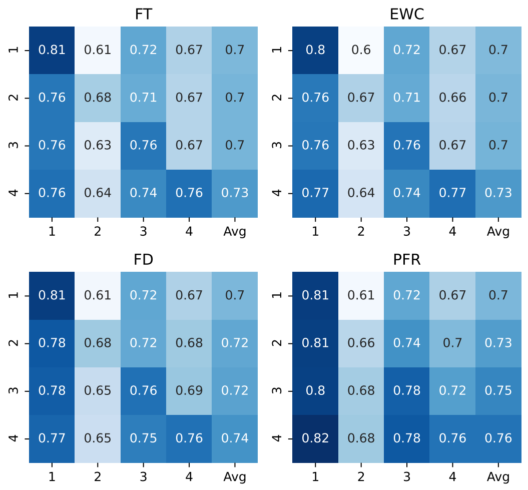


Figure 2.2: Task-aware performance after the four consecutive tasks on CIFAR-100 for several methods. Each row reports the results after each task, and columns represent on which task the model is evaluated. The last column reports average accuracy after each trained task.

and feature distillation(FD) on CIFAR-100 split into 50 or 100 tasks. In the case of 100 tasks, we only have a single class per task, which is an interesting setting since there are no negative classes, forcing the network to learn representations that are discriminative at the instance level. Results are presented in Figure 2.4a, where accuracy over all classes is given per training session for each method. Without any mitigation of forgetting, FT cannot maintain the learned representations in longer tasks sequences, dropping closely to the level of a randomly initialized backbone (23%) in the extreme case of 100 tasks. FD is also struggling on the longer sequence of 100 tasks. Only, our method shows stable results, preventing forgetting of the learned representation and progressing steadily. Similar results can be observed for 50 tasks, however the differences are not as pronounced as for the 100 task sequence.

2.4.4 The influence of regularization

Each regularization method is applied differently to the self-supervised network. To assess the influence of regularization and quantify its effect, we use the *forgetting*

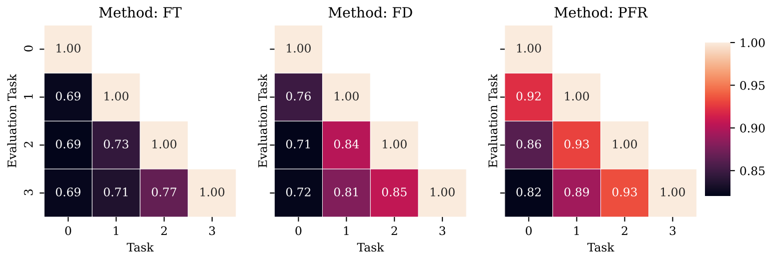


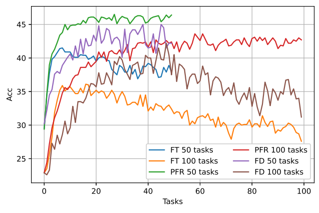
Figure 2.3: Representation similarity compared with CKA for FT, FD, and PFR during incremental training.

and *intransigence* measures defined in [19]. Forgetting measures the average drop in accuracy per-task during continual learning. Intransigence describes the inability of a model to learn a new task. Formally, it is the difference between a referential model accuracy at task t – for us jointly trained self-supervised model up to task t – and the current task accuracy measured using held-out data.

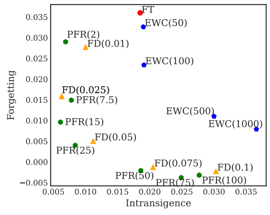
In Figure 2.4b all methods with different λ parameter values are shown for CIFAR-100 dataset. FT is a point of reference here since it uses no regularization and maximum forgetting is expected. EWC is a weight regularization method that regularizes network weight changes using the Fisher Information Matrix. With larger lambda, forgetting is lower, but at the same time we pay the price of larger intransigence. As in the other experiments, we found that weight regularization does not obtain satisfactory results when applied to continual self-supervised learning. Feature distillation represents a better trade-off. The closer the results are to the bottom left corner, the better they are, and here PFR is the clear winner. The PFR results are consistently better than FD by some margin, which implies that the additional flexibility of the model introduced by the temporal projection network indeed leads to higher plasticity while at the same time keeping forgetting low. These results are confirmed on the larger Tiny ImageNet dataset (see Figure 2.4c). Here, the gap between FD and PFR is much larger, showing that projected functional regularization suffers from much lower forgetting at equal intransigence values.

2.4.5 Generality of the Approach

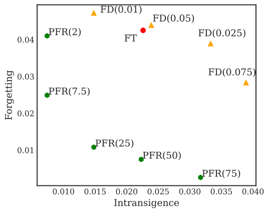
In order to verify that PFR generalizes to other self-supervised approaches, we conducted a series of experiments with SimCLR [22], SimSiam [24], and BarlowTwins [157].



(a)



(b)



(c)

Figure 2.4: (a): Performance of several methods for different numbers of tasks on CIFAR-100. (b, c): Influence of regularization on forgetting and intransigence for EWC, FD, and PFR. FT uses no regularization and represents a point of reference. The regularization hyperparameter λ is given in parentheses, (b) presents results for CIFAR-100 dataset, (c) for Tiny ImageNet.

Table 2.2: Accuracy of SimCLR, SimSiam and BarlowTwins on CIFAR-100 split into 10 tasks of 10 classes.

Method	**Task 1**	**Task 2**	**Task 5**	**Task 10**
SimCLR PFR	40.4	44.7	47.2	48.2
SimSiam PFR	43.1	50.2	53.1	55.1
BarlowT PFR	45.1	50.6	54.7	**55.4**
SimCLR FT	40.4	41.2	40.6	42.8
SimSiam FT	43.1	46.0	47.0	46.7
BarlowT FT	45.1	48.8	48.9	**47.0**

In Table 2.2 we present results of fine-tuning and PFR for a ten task scenario. PFR results in 5.4%, 8.4%, and 8.1% improvement over FT after the final task. For this longer task sequence scenario the gain of our method with respect to FT is much larger when compared with results in Table 2.1. Starting from the second task, the effect of projected regularization is clear. In the ten task scenario, SimSiam after the first five tasks begins to outperform BarlowTwins, which is reflected by the results in the Task 10 column.

2.4.6 Transfer to downstream tasks

To better asses the quality of the trained representation, we evaluated all methods on a series of different downstream datasets. This allows us to evaluate the trans-

Table 2.3: Accuracy on TinyImageNet split into four tasks. The learned representation is evaluated using a linear classifier over all classes after each task.

Method	**Task 1**	**Task 2**	**Task 3**	**Task 4**
Joint (no CL)	-	-	-	46.0
FD	35.3	35.9	36.6	36.6
EWC	35.3	37.4	36.9	38.2
PFR	35.3	38.6	39.9	**42.3**
FT	35.3	38.3	38.5	39.1

ferability of the learned features during the continual training process. The results for the smaller sized (32x32 images) CIFAR-100 and SVHN datasets are in Table 2.1 (bottom table). We observe a similar outcome as in the source dataset evaluation. The best results use our PFR method, followed by FD, EWC, and finally FT. The results are consistent during incremental learning: the better the representation is on a source classification task, the better it is on the target (SVHN) dataset after each task.

Furthermore, we trained all the methods on a larger dataset (Tiny ImageNet). We use the same procedure as for CIFAR-100 with four tasks, but with bigger images (64×64) and more classes (200). The results are presented in Table 2.3. In this dataset our method (PFR) surpasses FD, which is followed by the modified EWC and FT.

As in CIFAR100 we evaluate our networks trained on Tiny Imagenet on different downstream datasets (Cars and Aircraft). The results are given in Table 2.4. For these datasets, as in CIFAR100, the accuracy of the various techniques follows the same pattern: PFR yields the best results, followed by FD, EWC, and finally FT.

2.5 Conclusion

In this chapter, we proposed a method for incremental self-supervised learning without the need for any stored examples of previous tasks. Most existing regularization methods for continual learning are applied to class predictions or logits. Such approaches applied to self-supervised representation learning result in low plasticity. To address this, we propose Projected Functional Regularization via a temporal projection network that ensures that the newly learned feature space preserves information of the previous one, while still allowing for the learning of

Table 2.4: Transfer Learning to downstream tasks.

Cars				
Method	**Task 1**	**Task 2**	**Task 3**	**Task 4**
Joint (no CL)	-	-	-	34.5
FD	27.1	30.6	31.8	31.1
EWC	27.1	28.3	27.7	27.8
PFR	27.1	31.1	33.1	**33.8**
FT	27.1	29.5	29.2	31.5

Aircraft				
Method	**Task 1**	**Task 2**	**Task 3**	**Task 4**
Joint (no CL)	-	-	-	30.0
FD	24.6	25.6	27.0	27.0
EWC	24.6	23.8	25.3	25.1
PFR	24.6	26.2	27.0	**28.4**
FT	24.6	25.6	26.9	27.0

new features, resulting in higher plasticity. Extensive results on CIFAR100 and Tiny ImageNet demonstrate that our approach outperforms standard feature distillation by a considerable margin.

Finally, there are several limitations and future directions we discuss here. First, due to the high computational demands, experiments have been performed on low-resolution images, and they need to be confirmed for higher resolution data. Next, our method assumes access to task boundaries and cannot be directly applied in the task-free setting (without task boundaries) [3]. We think that this can be addressed by replacing the regularization based on the model from the previous task, with a regularization model that is updated with momentum. Next, continual learning of transformer architectures has only recently started [36, 114]. Self-supervised learning is a key ingredient of the transformer network training, and integrating our theory with these attention-based architectures for their continual learning is especially interesting. Finally, extending the theory with a limited replay buffer is of interest and would allow to directly report class-incremental learning results where the buffer is used to compute the classifier layer.

3 Plasticity-Optimized Complementary Networks for Unsupervised Continual Learning[*]

3.1 Introduction

Continual learning (CL) designs algorithms that can learn from shifting distributions (non-IID data), generally this is modeled by learning from a sequence of tasks [27]. The main challenge for these methods is the problem of catastrophic forgetting [99], which is a dramatic drop in performance on previous tasks. Most CL approaches, therefore, need to address the trade-off between acquiring new knowledge (plasticity) and preventing forgetting of previous knowledge (stability). The vast majority of existing methods in continual learning have focussed on supervised learning, where the incoming data stream is fully labeled. In this chapter, we focus on continual learning on unsupervised data.

Only recently, some works have explored continual learning on unsupervised non-IID data-streams [43,53]. Motivated by the tremendous progress in unsupervised learning, notably of contrastive learning approaches [22,157], methods aim to extend these methods to the continual setting. An additional motivation is the fact that unsupervised learning tends to lead to more generalizable feature representations since features that are not relevant to the specific discriminative task are not automatically discarded. This can potentially lead to representations that can faster incorporate new tasks without incurring significant amounts of forgetting. PFR [53] uses a projection head after the feature extractor of an SSL framework to predict past representations. Projected representations are motivated to be close to past representations; therefore, the present model is encouraged to remember past knowledge. CaSSLe [43] uses a similar strategy as PFR, but the projection head is used after the projector of the SSL approach. Even though these methods obtain satisfactory results, they struggle to adapt to new tasks without jeopardizing the vast knowledge already accumulated by the network. We hypothesize that the regularization imposed by these methods to avoid forgetting hurts the learning process of CURL in the following ways: 1) the SSL component cannot fully adapt to the

[*]This chapter is based on a publication in the Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision 2024 [52]

incoming data (low plasticity), 2) The model will have a significant drift (forgetting) when the current model is unable to perform the SSL pretext-task. These effects increase as the number of tasks increases, and consequently, the data for training reduces.

Complementary learning systems (CLS) theory [79, 98] proposes a computational framework in which the interplay between a fast (episodic memory/specific experience) and a slow (semantic/general structured) memory system is the core of the mechanism of knowledge consolidation. Several existing CL methods have taken inspiration from CLS as a way to find a good stability-plasticity trade-off (see [108, 112] for a review). The fast learner can quickly adapt to new knowledge, which then is carefully absorbed by the slower learner. DualNet [117] proposes to use a self-supervised method to train the slow, more generic learner, which weights can be quickly adapted for solving a supervised task with exemplars from the replay buffer. Recently, in [5] the authors proposed to use a CLS-based approach and maintain separate plastic and stable models for online CL with experience replay. However, existing methods that exploit CLS for continual learning have in common that they only consider a supervised learning scenario.

This chapter aims to apply complementary learning systems theory to improve continual learning from unsupervised data streams. The existing methods [43, 53] can suffer from suboptimal stability and plasticity on longer sequences since they have difficulty adapting to the new knowledge required to address the latest task, while maintaining the vast knowledge already learned on earlier tasks. Instead, we propose to train an expert network that is relieved of the task of keeping the previous knowledge and can focus on the task of performing optimally on new tasks. In the second phase, we combine this new knowledge with the old network in an adaptation-retrospection phase to avoid forgetting.

In conclusion, the main contributions of this work are :

- A new exemplar-free continual unsupervised representation learning (CURL) method called *Plasticity-Optimized COmplementary Networks* (POCON). Existing CURL methods learn new knowledge while imposing regularization to prevent forgetting. Instead, POCON separates the learning of new knowledge from the knowledge integration part. Analysis confirms that this leads to a better stability-plasticity trade-off.

- Extensive experiments confirm that POCON outperforms state-of-the-art CURL on various settings (e.g., a 5-9 % performance gain over CaSSLe on ImageNet100 for a 20-100 task-split). Unlike previous CURL methods, POCON can thrive in low-data regimes (such as small-task incremental learning) and setups without task boundaries. We also demonstrate the application of POCON to semi-supervised continual learning.

- We propose and evaluate a *heterogeneous* version of POCON, where the main network can have a different network architecture than the expert. This opens up the possibility for interesting applications where a slow/big network can be deployed in a cloud environment, while a fast/slow learner can be utilized on an edge device, such as a mobile phone.

3.2 Related work

Continual Learning and Class Incremental Learning. Existing continual learning methods can be broadly categorized into three types: replay-based, architecture-based, and regularization-based methods [27, 96]. Replay-based methods either save a small amount of data from previously seen tasks [8, 21] or generate synthetic data with a generative model [145, 160]. The replay data can be used during training together with the current data, such as in iCaRL [122] and LUCIR [66], or to constrain the gradient direction while training, such as in AGEM [20]. Architecture-based methods activate different subsets of network parameters for different tasks by allowing model parameters to grow with the number of tasks. Previous works following this strategy include HAT [131], Piggyback [93], PackNet [94], DER [154] and Ternary Masks [97]. Regularization-based methods add a regularization term derived from knowledge of previous tasks to the training loss. This can be done by either regularizing the weight space, which constrains important parameters [133, 137], or the functional space, which constrains predictions or intermediate features [25, 35, 67]. EWC [73], MAS [2], REWC [84], SI [158], and RWalk [19] constrain the importance of network parameters to prevent forgetting. Methods such as LwF [83], LwM [31], and BiC [149] leverage knowledge distillation to regularize features or predictions. DMC [162] work is more related to POCON as the authors proposed to train the expert network without any regularization for a classification task. However, after that, in a distillation phase an additional auxiliary dataset is used to integrate old and new knowledge.

Self-supervised representation learning. In recent years, unsupervised methods based on self-supervision have become dominant in learning representation for computer vision systems. The aim of self-supervised learning is to acquire high-quality image representations without explicit labeling. Initially, these methods addressed some well-defined pretext tasks, such as predicting rotation [50], determining patch position [32], or solving jigsaw puzzles in images [106], and labels for these discriminative pretext tasks can be automatically computed to enable learning of meaningful feature representations of images. Recently, researchers have adapted contrastive methods for use with unlabeled data and have placed more emphasis on instance-level data augmentation to find similar or contrasting

Method	*Labels*	*Exemplars*	*Regularization*
DMC [162]	✓	✓*	✗
DualNet [117]	✓	✓	✓
CLS-ER [5]	✓	✓	✓
LUMP [89]	✗	✓	✓
PFR [53]	✗	✗	✓
CaSSLe [43]	✗	✗	✓
POCON	✗	✗	✗

Table 3.1: Comparison of SSL-based CURL. Only POCON does not use any regularization during the training, what results in higher plasticity for the current task. *DMC uses an auxilary dataset for distillation, instead of exemplars.

samples [9,15,22,57,157]. These methods heavily rely on stochastic data augmentation [151,171] to generate sufficient similar examples for learning representations, and negative examples are either randomly sampled or excluded entirely [24].

Self-supervised learning has also been used to improve the learning of a sequence of supervised tasks [157,168]. Their objective is not to learn from unlabeled data, but rather to use self-supervised learning to further enrich the feature representation. Similarly, pre-trained models with self-supervision have been used to improve incremental average classification metrics [47] with data augmentation, distillation, and even exemplars. Training self-supervised models directly on class-IL setting without exemplars was also proposed in [43,53], where both present results that self-supervised learning mitigates the problem of forgetting.

Complementary learning systems. There are CLS-based methods that use several networks in addition to a rehearsal memory or pseudo-sample generator. FearNet [72] uses a hippocampal network for recalling new examples, a PFC network for long-term memories, and a third network to select between the PFC or hippocampal networks for a particular instance. In [113], they propose a G-EM network that performs unsupervised learning from spatiotemporal representations and a G-SM network that uses signals from G-EM and class labels to learn from videos incrementally. Closer to our work are DualNet [117] and CLS-ER [5] (previously explained); however, both models are supervised and use exemplars. Table 3.1 presents a summary of similar CL methods.

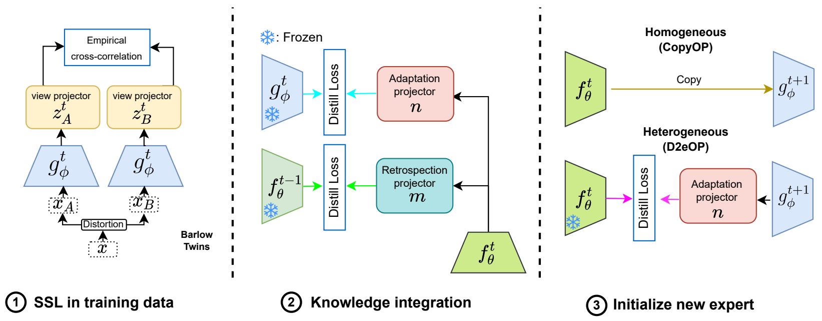


Figure 3.1: Overview of the POCON method that uses multi-stage training of complementary learning system for unsupervised continual representation learning. Three stages of training allow POCON to maintain fast and efficient adaptation of the fast expert network while maintaining stable representation in a slow learner – the main network. To avoid forgetting, knowledge from the new task expert is integrated (adaptation) with the previous state of the slow network (retrospection). The last stage of the training is dedicated for the best preparation of the new expert for the new task.

3.3 Method

In this section, we describe our approach for continual learning of self-supervised representations, referred to as Plasticity-Optimized COmplementary Networks (POCON), which eliminates the need for memory or replay. Our method is based on the complementary learning system (CLS) framework, and involves an interplay between a fast-expert learner and a slow-main network. Our work is motivated by the recognition that fast adaptation to new data is crucial in constructing more robust representations during continual unsupervised representation learning. Rather than attempting to maintain network stability in its old state through strict or relaxed distillation methods, as suggested in recent works [43, 53], POCON allows the expert network to learn a new task freely without any restrictions. After acquiring the new knowledge, we integrate it into the main network. In turn, the main network serves as a good starting point for the new expert. Before presenting the details of the POCON method, we first provide a brief introduction to the problem of continual self-supervised representation learning.

3.3.1 Self-supervised representation learning

In recent research on self-supervised learning, the objective is to train a network $f_\theta : \mathscr{X} \to \mathscr{F}$, which maps input data from $\mathscr{X}$ to output feature representations in $\mathscr{F}$. The network is trained using unlabeled input data x sampled from a distribution $\mathscr{D}$. The learned feature representation is subsequently used to facilitate various downstream tasks. In this chapter, we employ the BarlowTwins approach [157] for self-supervised learning of the representation network g_θ. This approach serves as a common baseline for previous works [43, 53]. However, the proposed method is versatile and can be extended to other self-supervised techniques.

In Fig. 3.1 (left) the BarlowTwins architecture is presented. Both branches use a projector network z and both share the same parameters, but while computing the empirical cross-correlation loss operates on different views of the same sample x created by data augmentation techniques. For simplicity of notation, we omit the explicit mention of z network parameters, since it is not utilized by downstream tasks. The BarlowTwins method eliminates the need for explicit negative samples and achieves comparable performance while maintaining computational efficiency. It assumes that negatives are accessible in each mini-batch to estimate correlations among all samples in it. The network is trained by minimizing an invariance and a redundancy reduction term in the loss function [157]. Here, different augmented views X_A and X_B of the same data samples X are taken from the set of data augmentations $\mathscr{D}^*$. This leads to the loss defined as:

$$\mathscr{L}_c = \mathbb{E}_{X_A, X_B \sim \mathscr{D}^*}\Big[\sum_i (1 - \mathscr{C}_{ii})^2 + \lambda \sum_i \sum_{j \neq i} {\mathscr{C}_{ij}}^2\Big], \tag{3.1}$$

where λ is a positive constant trade-off parameter between both terms, and where $\mathscr{C}$ is the cross-correlation matrix computed between the representations z of all samples X_A and X_B in a mini-batch indexed by b: $\mathscr{C}_{ij} = \sum_b z^A_{b,i} z^B_{b,j} / (\sqrt{\sum_b (z^A_{b,i})^2} \sqrt{\sum_b (z^B_{b,j})^2})$. The cross-correlation matrix $\mathscr{C}$ contains values ranging from -1.0 (worst) to 1.0 (best) for the correlation between the projector's outputs: $Z_A = z(g_\phi(X_A))$ and $Z_B = z(g_\phi(X_B))$. The invariance term of the loss function encourages the diagonal elements to have a value of 1. This ensures that the learned embedding is invariant to the applied data augmentations. Meanwhile, the second term (redundancy reduction) maintains the off-diagonal elements close to zero and decorrelates the outputs of non-related images.

3.3.2 Continual SSL Problem Definition

In this chapter, we consider a CL scenario in which the feature extractor f_θ must learn from a set of task $\{1,\ldots,T\}$ from different distributions, where each task t from that set follows the distributions $\mathscr{D}_t$. We would like to find the parameters θ of the feature extractor f_θ that minimizes the summed loss over all tasks T:

$$\arg\min_{\theta} \sum_{t=1}^{T} \mathscr{L}_c^t, \tag{3.2}$$

where $\mathscr{L}_c^t = \mathbb{E}_{X_A,X_B\sim\mathscr{D}_t^*}[\mathscr{L}_c]$ and $\mathscr{L}_c$ is defined as in Eq. 3.1. However, finding the right θ poses the main problem in continual learning, as previous data $D_1,...,D_{t-1}$ is not available at time t and Eq. 3.2 cannot be minimized directly.

3.3.3 Plasticity-Optimized Complementary Networks

We propose Plasticity-Optimized Complementary Networks (POCON) based on CLS framework. POCON is composed of three stages training (see Fig. 3.1 for details): ① learn expert knowledge from current data task, ② integrate new expert knowledge to the main network, and ③ initialize the new expert from the updated main network. Each stage is explained in the details in the next sections.

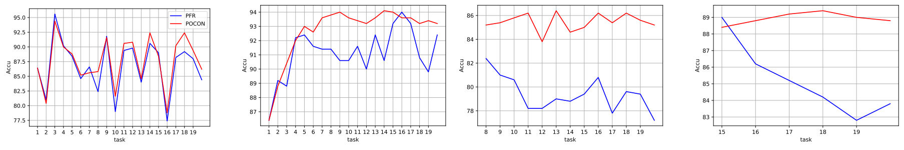
***Stage* ①: SSL of expert for the current task.** In this step, we are interested to fully-adapt to the input training data of the current task. Hence, a feature extractor g_θ^t is used to learn in a self-supervised way (following Eq. 3.1) on the data D_t. Note that unlike previous methods (PFR and CaSSLe), we do not constraint in any way our expert during the training (like imposing regularization to prevent forgetting). We allow the expert network to be fully plastic and optimal for learning representation for the current task.

***Stage* ②: Knowledge integration.** Once we absorb the knowledge of the current task in the expert network at *Stage* ①, it needs to be transferred and accumulated to the main feature extractor f_θ without forgetting previous tasks (see Fig.3.1 ②). To do so, we employ an additional *adaptation projector* $n : \mathcal{Z} \rightarrow \mathcal{W}$, which maps the embedding space from the main network f_θ to the embedding space learned in the expert network g_{ϕ^t}. Then, to avoid forgetting, we use a *retrospection projector* $m : \mathcal{Z} \rightarrow \mathcal{Z}$ that maps the embedding space learned on the current task back to the embedding space learned on the previous ones. The final loss function for the knowledge integration stage consists of an adaptation and retrospection component:

$$\mathscr{L}_{INT}^{t} = \mathbb{E}_{X_A \sim \mathscr{D}_t^*}\left[\sum_{i=0}^{|X_A|} \parallel n(g_{\phi^t}(x_a)) - f_{\theta^t}(x_a) \parallel + \parallel m(f_{\theta^t}(x_a)) - f_{\theta^{t-1}}(x_a) \parallel \right] \tag{3.3}$$

where both sources from which we integrate our knowledge: previous main network $f_{\theta^{t-1}}$ and the current expert g_{ϕ^t} are frozen, and only f_{θ^t} and adaptor networks n and m are being updated by this loss function. The goal is to integrate knowledge using distillation and current task data.

***Stage* ③: New expert initialization.** In order to begin the training on the next task, the expert g_{ϕ}^{t+1} must be prepared to solve the new task with the best prior knowledge for training new task representation efficiently. In this stage, we need to initialize a new expert (see Fig. 3.1 ③) in the best way. Improper initialization can influence the training epochs in *Stage* ① and make the problem of adaptation for projector n more difficult in *Stage* ②. We looked into two potential initialization setups based on the similarities between the expert and main backbones. The homogeneous setup, where both the main network and the expert network share the same architecture. This is the default setup in our experiments. In addition, we will also consider the heterogeneous setup, where the expert has another architecture than the main network. This allows, for example, to apply smaller networks when per task data is limited, or computation should be performed on edge devices.
Homogeneous setup (CopyOP): In order to begin the training of g_{ϕ}^{t+1} with all the accumulated knowledge till t we can just copy the weight of the main network f_{θ}^{t} into g_{ϕ}^{t+1}. This operation avoids the recency bias of (g_{ϕ}^{t}) and provides an excellent initialization point for g_{ϕ}^{t+1} to continue learning. Furthermore, CopyOP makes the problem of the adaptation projector n easier since g_{ϕ}^{t+1} will have a similar representation to f_{θ}^{t}. The main drawback of CopyOP is that it constrains of POCON to use the same architecture for the main f_{θ}^{t} and the expert g_{ϕ}^{t+1} networks.
Heterogeneous setup (D2eOP): This distillation-based initialization method allows the use of heterogeneous network architectures for f_{θ}^{t} and g_{ϕ}^{t+1} in POCON. In order to transfer the knowledge, we propose a projected distillation as in *Stage* ② using a fresh adaptation projector n. Despite being more computationally demanding, this way of transferring knowledge offers one big advantage – different architecture for g_{ϕ}^{t+1} allows the use of a smaller backbone network or even very different ones, e.g. ViT. Using a smaller backbone is useful for low-data regimes (as we will show later) or for devices low in computational power at the time of learning the expert



(a) Plasticity (b) Stability task 1 (c) Stability task 8 (d) Stability task 15

Figure 3.2: Plasticity-stability trade-off in POCON and PFR: (a) accuracy of a current task during the continual learning session, (b-d) accuracy of task 1, 8, and 15 in the following incremental steps of learning the representation with POCON and PFR. POCON presents superior stability to PFR at the same time still maintaining better plasticity, with its non-restrictive training of the expert network.

(robotics, edge devices). The loss function for D2eOP in *Stage* ③ is given as:

$$\mathcal{L}^t_{D2eOP} = \mathbb{E}_{X_A \sim \mathcal{D}^*_t}\left[\sum_{i=1}^{|X_A|} \| n(g_\phi^{t+1}(x_a)) - f_{\theta^t}(x_a) \|\right] \tag{3.4}$$

where n is a projector that adapts the embedding space of new expert g_ϕ^{t+1} to the previous one from the main network with current task data D^t. The difference of D2eOP and CopyOP is presented in the right column of Fig. 3.1.

Other initialization options exists. We discuss them and provide more results in the Appendix.

3.3.4 Plasticity-stability trade-off in CURL with SSL

Our work is motivated by two key observations:

- Regularized-based CURL models (PFR and CaSSLe) have an implicit handicapped plasticity. During the training, they learn new tasks data using SSL while maintaining the backbone representations close to the previous tasks to avoid forgetting. Therefore, the backbone network cannot fully adapt to current data due to the regularization imposed by the CL method.

- As the number of tasks increases, current CURL models lose stability. In this case, the data distribution changes more abruptly and the backbone is pushed to follow an imperfect estimation of the new distribution restricted by the regularization of the old model (see first observation). Hence, these models are not as plastic as finetuning and do not present stable behavior.

In Fig. 3.2 we present the stability-plasticiy tradeoff for twenty task partition of

CIFAR100 in PFR and our method (POCON). For the plasticity plot, we evaluate the accuracy at the end of each task t using task t data for training and evaluation. The stability plots were made by training and evaluating in a fixed task partition (1, 8, and 15 for Fig. 3.2) after the training of each task t.

As Fig. 3.2 *a*) shows POCON has a higher accuracy (more plastic) in most of the task during the whole training. As training an expert in stage ① is not constrained by any regularization. Morover, Fig. 3.2 *b*), *c*), and *d*) display how POCON is able to retain previous representations over the whole incremental learning session, where our double distillation (adaptation and retrospection) is able to retain the learned representation correctly.

3.4 Experimental Results

3.4.1 Experimental setup

Datasets We use the following datasets: *CIFAR-100* [76], consists of 100 object classes in 45,000 images for training, 5,000 for validation, and 10,000 for test with 100 classes. All images are 32×32 pixels; *TinyImageNet* a rescaled subset of 200 ImageNet [29] classes used in [136] and containing 64×64 pixel images. Each class has 500 training images, 50 validation images, and 50 test images; *ImageNet100* a subset of one hundred classes from ImageNet [29] that contains 130k images of size 224×224 pixels.

Training procedure In all experiments, we train ResNet-18 [62] (or ResNet-9 for the heterogeneity experiment) for expert and main network using SGD with an initial learning rate of 0.01 and a weight decay of 0.0001 for 250 epochs (200 for ImageNet100)in *Stage* ①. For *Stage* ② same optimization procedures as *Stage* ① is followed but for 500 epochs (400 for ImageNet100).

The data augmentation used in *Stage* ① is the same as in BarlowTwins [22]). Based on the self-supervised distillation ideas of [104, 129], we use a four-layer MLP projector as adaptation and retrospection projectors following the architecture of [104].

Downstream classifiers are by default linear and trained with a CE-loss and Adam optimizer with a learning rate 5e-2 on CIFAR-100, and 3 on TinyImageNet. We use validation data to implement a patience scheme that lowers the learning rate by a factor of 0.3 and 0.06 up to three times while training a downstream task classifier. For ImageNet100 we use the same training and evaluation procedure as [43].

Baseline methods

We only compare to exemplar-free methods and exclude methods that require replay from our comparison[†].

*Fine-tuning (**FT**)*: The network is trained sequentially on each task without access to previous data and with no mitigation of catastrophic forgetting.

Joint: We perform joint training with fine-tuning on all data which provides an upper bound. Equivalent to having a single-task scenario.

PFR [53] and ***CaSSLe*** [43] with Barlow Twins: We use the code and hyperparameters provided by the authors, in PFR we used $\lambda = 25$ for all the experiments.

In continual semi-supervised learning (section 3.4.5), we consider the following methods. Regularization-based methods: Learning without Forgetting (LwF) [83], online Elastic Weight Consolidation (oEWC) [73], Replay-based methods: Experience Replay (ER) [124], iCaRL [122] and GDumbciteprabhu2020gdumb, and Continual semi-supervised learning methods: CCIC [11], PAWS [7] and NNCSL [71].

3.4.2 Continual representation learning

In this experiment, we evaluate all methods in the continual representation learning setting, where each task consist of a distinct set of classes from a single dataset. Splits are prepared similarly to the class incremental learning setting, but without access to labels. Specifically, we split datasets into four, ten, twenty, fifty, and one hundred equal tasks as done in [122]. In each task, we perform SSL (*Stage* ①), knowledge integration (*Stage* ②), and a new expert initialization (*Stage* ③). In the evaluation phase, we train a linear classifier using the learned representation of the main network encoder. Please note that the expert network is never used for evaluation (unless specified). We use all available test data to obtain the overall task-agnostic performance evaluation[‡]. In all our tables we report the accuracy in the last task.

Homogeneous setup. Table 3.2 presents the results for all the methods on three commonly used dataset - CIFAR100, TinyImageNet, and ImageNet100. For CIFAR100 the upper bound of Joint training for a single task is 65.4%. Note, that the gap between FT and *Joint* is getting bigger with more tasks, from 10.6% for four task up to 38.4% for extreme case of one hundred tasks. POCON is significantly better than CaSSLe and PFR for different number of tasks. In four tasks setting, POCON is only 1.7% points lower than Joint, while second next, CaSSLe reaches 3.3% pints lower accuracy. With the increasing number of tasks, the CaSSLe performance drops faster than PFR, while POCON maintains superior results against others and presents the lowest decrease in performance.

[†]Code available at https://github.com/alviur/pocon_wacv2024

[‡]We use *task-agnostic* evaluation in this chapter to refer to the class-incremental learning evaluation [96] as in CaSSLe and PFR methods

CIFAR-100 (32x32)					
Method	**4 tasks**	**10 tasks**	**20 tasks**	**50 tasks**	**100 tasks**
FT	54.8	50.94	44.95	38.0	27.0
CaSSLe	59.80	52.5	49.6	45.3	42.10
PFR	59.70	54.33	44.80	46.5	43.30
POCON	**63.7**	**60.5**	**56.8**	**48.9**	**48.94**
Joint	65.4				

TinyImagenet (64x64)					
Method	**4 tasks**	**10 tasks**	**20 tasks**	**50 tasks**	**100 tasks**
FT	41.95	36.55	32.29	22.34	2.80
CaSSLe	**46.37**	**41.53**	38.18	28.08	25.38
PFR	42.23	39.20	31.22	25.87	21.20
POCON	40.97	41.06	**41.14**	**37.20**	**30.24**
Joint	50.18				

ImageNet100 (224x224)					
Method	**5 tasks**	**10 tasks**	**20 tasks**	**50 tasks**	**100 tasks**
FT	56.10	48.13	42.73	39.64	21.03
CaSSLe	**67.56**	59.78	53.92	46.64	36.44
PFR	66.12	60.46	54.84	42.18	38.34
POCON	66.30	**61.36**	**59.32**	**53.50**	**45.40**
Joint	71.06				

Table 3.2: Accuracy of a linear evaluation on equal split various datasets and different number of tasks with ResNet-18. POCON present better result than other regularization-based methods and maintain high accuracy even with increasing number of tasks.

The results for TinyImagenet and ImageNet100 follow the same procedure as for CIFAR-100, but here we have larger images (64x64 and 224x224) and more classes for TinyImagenet (200). In this case, POCON outperforms other methods whenever the number of tasks is higher, 20 for TinyImangeNet and 10 for ImageNet100. For a few tasks scenario, POCON will be close to other CL methods due to high data availability. The accuracy gap between POCON and other methods increases with

the number of tasks.
Heterogeneous setup. An expert in POCON can use a different network architecture than the main network. That opens the possibility of using a smaller network for the expert whenever this can be beneficial, e.g., tasks are small with not enough data to train a large ResNet-18 network, or the device where the expert network is being trained is not powerful enough (robot, edge). We investigated heterogeneous architecture use in POCON with a smaller network, ResNet-9 which has 6.5M number of parameters instead of 11M in ResNet-18. The results are presented in Table 3.3.

The different combinations of POCON are presented for using smaller network in expert only, or for both, expert and main networks. With increasing number of tasks it is more beneficial to use smaller expert (20 tasks). And having less data per task (50 and 100 tasks) we see improved results when as well the main network is smaller, we gain 4.6% changing from ResNet-18 to ResNet-9 for one hundred tasks.

Method	**Arch**	**4 tasks**	**10 tasks**	**20 tasks**	**50 tasks**	**100 tasks**
POCON	R18-R18	**63.7**	**60.5**	56.8	48.9	48.94
POCON	R18-R9	62.05	60.5	**57.48**	49.7	47.94
POCON	R9-R9	58.34	58.32	56.07	**53.3**	**51.22**

Table 3.3: Accuracy of linear evaluation with heterogeneous POCON architecture using ResNet-18 (R18) and ResNet-9 (R9) on split CIFAR-100. With increasing number of tasks and lower data POCON benefits from a smaller ResNet-9 network, first, only for the expert (20 tasks) and later (50 tasks) for both networks, expert and main.

3.4.3 Other possible expert initializations

We investigated two additional initialization regimes for the expert network:

ScratchOP - which is a trivial way to initialize g_ϕ^{t+1} with random weights. This initialization is fast and completely clean of any bias (either good or bad) of previous data. The main drawback is that previous knowledge could help g_ϕ^{t+1} to learn better representation. There is no knowledge transfer to the new expert.

FtOP - which begin the training of the next task using the expert from the current one. That is a reasonable assumption if the distributions of consecutive tasks are similar. This initialization copies the weights of g_ϕ^t into g_ϕ^{t+1}, allowing a good initialization point for the training of the task $t+1$. This approach has two disadvantages: first, the knowledge of g_ϕ^t has a lot of recency bias (g_ϕ^t has already forgotten knowledge from tasks $t-1$); second, the architectures of g_ϕ^t and g_ϕ^{t+1} need

Method	exemplar	CIFAR-100 0.8%	CIFAR-100 5.0%	CIFAR-100 25.0%
Exemplar-based methods				
ER [124]	✓	8.2 ± 0.1	13.7 ± 0.6	17.1 ± 0.7
iCaRL [122]	✓	3.6 ± 0.1	11.3 ± 0.3	27.6 ± 0.4
GDumb [119]	✓	8.6 ± 0.1	9.9 ± 0.4	10.1 ± 0.4
CCIC [11] (500)	✓	11.5 ± 0.7	19.5 ± 0.2	20.3 ± 0.3
PAWS [7] (500)	✓	16.1 ± 0.4	21.2 ± 0.4	19.2 ± 0.4
NNCSL [71] (500)	✓	**27.4** ± 0.5	**31.4** ± 0.4	**35.3** ± 0.3
Exemplar-free methods				
Fine-tuning	✗	1.8 ± 0.2	5.0 ± 0.3	7.8 ± 0.1
LwF [83]	✗	1.6 ± 0.1	4.5 ± 0.1	8.0 ± 0.1
oEWC [73]	✗	1.4 ± 0.1	4.7 ± 0.1	7.8 ± 0.4
Prototypes	✗	19.2 ± 0.9	23.5 ± 0.4	24.1 ± 0.1
Prototypes+*SDC*	✗	**22.7** ± 0.6	**27.6** ± 0.4	**28.5** ± 0.1

Table 3.4: Semi-supervised continual learning comparison on CIFAR100 dataset. The number between brackets indicates the size of the memory buffer. We highlight the best method in each group with **bold** fonts.

to be homogenous to perform the copy operation.

Table 3.5 show results of ScratchOP and FtOP expert initialization for POCON in CIFAR-100. The results for CaSSLe and PFR are recall once again here for an easy comparison.

3.4.4 Task-free setting

To show the ability of POCON to handle varying data streams, we test it in the task-free setting. In this setting, there is no explicit boundary between tasks, the data from one task changes smoothly to the data from the next [81, 159]. This prevents methods from having a fixed point where the network can change and prepare for the new task; the adaptation is ongoing. For instance, when we receive data D_t there is a mix with the data D_{t-1}. At some point, we only get data from D_t, but later on, we will get a mix with D_{t+1}. For this experiment, we employ the data partition of [159] for the CIFAR100 dataset with beta equal to 4.

Fig. 3.3 depicts a data partition for 10 tasks and beta 4 based on [159]. The y-axis denotes the probability of selecting a sample in an iteration. Hence, the batches of data will contain a mixture of tasks data proportional to the probability of taking

Table 3.5: Results of ScratchOP and FtOP expert initialization for POCON in CIFAR-100.

CIFAR-100 (32x32)					
Method	**4 tasks**	**10 tasks**	**20 tasks**	**50 tasks**	**100 tasks**
FT	54.8	50.94	44.95	38.0	27.0
CaSSLe	59.80	52.5	49.6	45.3	42.10
PFR	59.70	54.33	44.80	46.5	43.30
POCON					
ScratchOP	57.64	46.87	39.69	30.42	30.50
FtOP	62.0	58.13	55.57	48.64	47.06
Joint	65.4				

a sample for each class (beta= 4 produces a mixture of two tasks). Fig. 3.3 shows where the mixed batches are taken, for instance at 8000 iteration for this particular instance. In such created setting, there are no explicit tasks' boundaries that could be used in the continual training session to initiate additonal step for training a new task.

Since there is no clear boundary, we cannot perform distillation without losing data from the stream of mini-batches. In this setting, POCON performs *Stage* ② in parallel with *Stage* ①. In order to do so, a frozen copy of the expert g_ϕ is used for the *Stage* ② while a new expert is learning on the current data. After s steps, a copy of the expert g_ϕ is passed to perform a distillation for ds steps. Note that we do not store any data; distillation for *Stage* ② and SSL in *Stage* ① is always performed using the current mini-batch data. *Stage* ③ is omitted, as the expert network is not changed and initialized, as there is no task switch.

To compare to other method, we adapted PFR to work in the task-free setting similarly. In this case, the feature extractor of past data is updated after s steps (copy of the current feature extractor), and the regularization is performed during the whole training as in normal PFR. We also present results with a simple fine-tuning (FT).

Fig. 3.4 presents the results of linear evaluation of the learned representation in continual learning for the ten, twenty, and fifty data partitions settings.

Only at the first steps at the beginning, for 10 and 20 tasks, PFR and FT accuracy is above POCON. However, that changes significantly in the following steps in favor of POCON, which continues improving learned representations. Unlike PFR and FT,

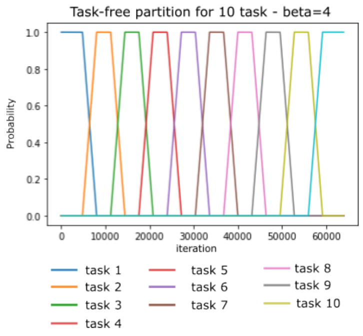
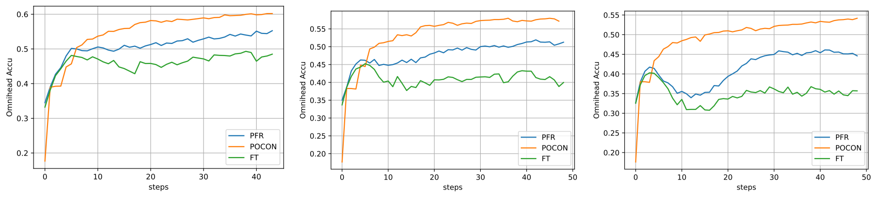


Figure 3.3: Sample data partition for 10 task and beta 4



(a) 10 blurred data partitions (b) 20 blurred data partitions (c) 50 blurred data partitions

Figure 3.4: Task-task setting for CIFAR-100 with different length of incremental learning sequence. Plots present linear evaluation of learned representation for the blurred 10, 20, and 30 data partitions respectively (like in [81, 159]). POCON presents significantly better results with stable learning accuracy curves.

POCON has a more stable learning curve for all task splits. The improvements over PFR and FT are better for more tasks, since the regularization-based method hurts plasticity while learning new tasks with PFR.

3.4.5 Continual semi-supervised learning

We propose a simple extension of POCON to the semi-supervised setting, where a small percentage of the data is labeled. We train the method in the same fashion as the unsupervised case, and we initially ignore available labels. After updating f_θ^t until convergence with POCON, we initialize the prototypes of each class as the average of the labeled samples. Then we assign all unlabeled data to the nearest prototype center, and we again compute the average of all samples assigned to each prototype. We perform the same procedure after each of the tasks. We call this method *Prototypes*. We also show for *Prototypes+SDC* where we use Semantic Drift Compensation [156] to prevent forgetting - we estimate and compensate for learned class prototypes drift.

In Table 3.4 we present the results for continual semi-supervised learning under three levels of supervision, namely with only 0.8%, 5.0% and 25.0% of samples labeled (following the settings of NNCSL [71] and CCIC [11]). Simply applying prototypes with POCON achieves much better results than the other exemplar-free methods (LwF [83] and oEWC [73]). The method outperforms the continual learning methods which only exploit the labeled data (such as ER, iCaRL, and GDumb). Furthermore, it also outperforms the semi-supervised methods CCIC and PAWS. Note that our method, without storing any exemplars, is only outperformed by the recent NNCSL method which requires exemplars and is dedicated to the semi-supervised learning setting.

3.5 Conclusions and Future directions

We proposed a method for exemplar-free continual unsupervised representation learning called *Plasticity-Optimized COmplementary Networks*. POCON trains an expert network that performs optimally on the current data without any regularization (optimizing plasticity). The knowledge is transferred to the main network through distillation while retaining similarity with the old main network to avoid forgetting. Experiments of CIFAR100, TinyImagenet, and ImageNet100 show that our approach outperforms other methods for SSL exemplar-free CL learning (PFR and CaSSLe), and it is especially good for many tasks scenarios.

The POCON method presents several opportunities for improvement. One promising direction is to extend the method to the scenario where multiple experts can be trained in parallel on different tasks, similar to the clients in federated learning. Secondly, heterogeneity of the fast and slow learner can be better investigated – how we can benefit from having different architectures (even mixed ones with transformer-based network).

Limitations. Although we show how to squeeze (and retain) the knowledge learned by the SSL method, when the number of samples per tasks is too low, the knowledge transfer (stages ② and ③) and the expert training (stage ①) degrades. Note that in this extreme scenario, we are going towards online continual learning.

4 Exemplar-free Continual Representation Learning via Learnable Drift Compensation*

4.1 Introduction

The standard supervised learning paradigm assumes that data are available in a single training session. This belies, however, the reality of many real-world scenarios in which machine learning methods must deal with data that arrive in a continuous and non-stationary stream of tasks. These shifting distributions lead to catastrophic forgetting if approached naively (for instance, with continuing stochastic gradient descent) [99]. Continual Learning (CL) considers the design of algorithms capable of learning from such streams of non-stationary data.

Several techniques have been proposed for continually training models, such as regularization [2, 73, 83, 158], exemplar replay [17, 36, 66, 122, 149], and growing architectures [127]. One group of methods focuses on the more difficult exemplar-free continual learning scenario where the storing of exemplars is prohibited [54, 115, 156, 168, 169]. This setting is relevant in applications where storing previous-task samples is not feasible due to memory limitations, legal constraints arising from new data privacy regulations (like GDPR), or when handling sensitive data such as medical imaging where patient identity must be protected.

One of the main limitations of exemplar-free CL methods is that, as the backbone is updated during the training of new tasks, the network experiences semantic feature drift [156]. This feature drift results in catastrophic forgetting and a significant drop in performance. As a consequence, most exemplar-free methods are evaluated in Warm Start settings (starting from a large first task or from a pretrained network) to start from a strong backbone and thus avoid significant semantic drift of backbone network [54, 66, 85, 100, 149, 156, 167–169]. In fact, a recent paper [115] showed that a simple strategy combined with a frozen backbone after learning the first task can lead to very competitive results in this Warm Start settings. In this chapter, we focus on the exemplar-free Cold Start setting [90] in which we train a network from scratch, starting with a small task, and continually update the

*This chapter is based on a publication in Proceedings of the IEEE/CVF European Conference on Computer Vision 2024 [51] backbone.[†]

Class Prototype (CP) accumulation is among the most successful approaches to exemplar-free continual learning [54, 70, 100, 111]. CP methods store class prototypes computed using training data and then use them to classify using a distance classifier in feature space. Class Prototype methods are attractive since they can be agnostically applied to any feature extractor, such as multilayer perceptrons, convolutional neural networks, or transformers. CP methods have been successfully applied in Warm Start settings [54, 100, 115] but are expected to struggle in Cold Start settings due to significant semantic drift.

In this chapter, we aim to apply class prototype methods in Cold Start settings. The main challenge in this setting is the semantic drift of the feature backbone, due to which class prototypes become progressively invalidated since the mean position of each class moves with each new task and cannot be corrected without access to old data in exemplar-free settings. However, in an initial analysis of feature drift (see Fig. 4.1), we observe that an oracle that perfectly compensates for feature drift leads to a significant recovery of performance. This is an important observation since it shows that semantic drift has not diminished the discriminative power of the feature representations. Consequently, we show that if drift is correctly estimated, feature drift compensation can correct for most of the performance drop resulting from it.

Existing drift estimation methods [140, 156] assume that the drift can locally be approximated with a translation. Instead, we propose a simple Learnable Drift Compensation (LDC) approach to update old class prototypes which has not prior assumptions on the nature of the drift. The learnable compensation maps the previous task features to the current feature space via a forward projector network. This compensation can be learned from only the stored old class prototypes and does not require class labels. Owing to the label-agnostic nature of LDC, we propose the first semi-supervised exemplar-free approach, which uses the labeled data only to compute prototypes and all available data to learn a robust forward projector. The main contributions of this work are as follows:

- We show that a large part of forgetting in prototype-based methods can be addressed by prototype compensation. Very little actual forgetting of features crucial for previous tasks is occurring, which motivates the need to develop drift compensation methods for continual representation learning.
- We propose a new exemplar-free continual learning method, which we call Learnable Drift Compensation (LDC), that is based on class prototype accu-

[†]This setting is predominant in exemplar-based continual learning but hardly explored for exemplar-free methods.

mulation, can compensate the drift of a moving backbone, and is fast and easy to add on top of CP-based methods.

- We experimentally demonstrate that LDC is better than previous drift compensation methods for supervised exemplar-free CL setup. We show that LDC can be applied to any self-supervised CL method to create the first, to the best of our knowledge, exemplar-free semi-supervised CL method.

4.2 Related Work

Continual Learning [73, 99] addresses the issue of *catastrophic forgetting* when training model on a non-i.i.d stream of data [96]. Various methods [96, 165] have been proposed to tackle this problem, including parameter isolation methods [94, 127, 131], regularization techniques [2, 73, 83], and rehearsal strategies [20, 122]. In this chapter, we focus on class-incremental learning [143], where the learner has no access to the task oracle during inference, without the need to store any exemplars.

Exemplar-free Class Incremental Learning. In this chapter, we consider the challenging exemplar-free CIL setting. PASS [168] aligns prototypes of new classes with those of previous tasks via augmentation with noise. SSRE [170] focuses on continually expanding the model's representations to accommodate new classes. FeTrIL [115] translates old prototype features using the difference between old and new prototypes to create pseudo-features of old data in the new space. In FeCAM [54], the heterogeneous distribution of features in class-incremental learning is exploited using anisotropic Mahalanobis distance instead of Euclidean metric.

Continual Representation Learning. CURL [121] performs continual unsupervised representation learning and is based on a variational autoencoder whose latent space can be sampled to replay samples of old classes. CURL is applicable only to small-sized image datasets, requires task boundaries, and is evaluated only on simple classification task. Using self-supervised learning to enhance the learning process of a sequence of supervised tasks was introduced in prior works [157, 168]. Their aim is not to learn directly from unlabeled data, but rather to leverage self-supervised learning to enrich the feature representation further. Similarly, several studies [14, 18, 47, 89, 109] have used pre-trained models with self-supervision to enhance incremental average classification accuracy through techniques like data augmentation, distillation, and even exemplars. Furthermore, models are trained in self-supervised way in the class-incremental learning (CL) setting without relying on exemplars, as seen in [43, 53].

Continual unsupervised learning can be straightforwardly extended to address the challenge of semi-supervised continual learning, where training phases abstain from utilizing labels entirely, similar to the approach in CaSSLe [43]. However, a limitation of this approach is mitigated by NNCSL [71], which builds upon the PAWS [7] for continual learning. Nonetheless, it is worth noting that these methods typically rely on exemplars to achieve satisfactory performance.

Drift estimation. An obstacle to exemplar-free methods is the drift of backbone features, which can result in catastrophic forgetting. Semantic drift compensation (SDC) [156] addresses *drift compensation* by computing the drift of features using new task data and approximating the drift of old class prototypes based on this information. Subsequent work [140] extended SDC by also considering feature drift. Elastic feature consolidation (EFC) [90] regularizes the drift in a direction highly relevant to the previous task using the empirical feature matrix. Recently, Adversarial drift compensation [55] was proposed to generate adversarial samples from new task data which behaves as pseudo-exemplars and are used for estimating drift of old classes. Our method shares a similar objective of estimating feature backbone drift, but we propose a different approach by suggesting the use of a dedicated network for drift compensation.

Memory efficient CIL through feature adaptation (MEA) [69] addresses the problem of feature drift. It stores the features after the backbone and corresponding labels. A learned mapping is applied to translate stored features into the new space to compensate for drift. Unlike our method, MEA requires labels of features, making it unsuitable for unsupervised and semi-supervised settings. Additionally, MEA stores an extensive amount of features for all old classes, leading to higher memory requirements.

4.3 Preliminaries

4.3.1 The Continual Learning Problem

We consider a class-incremental learning setup where a model learns the dataset D split into a sequence of tasks $T = \{t_1, ..., t_s\}$. During the training of each task, the model only has access to the data in the split t (exemplar-free setup). Each task contains a set of m classes $C^t = \{c_1^t, c_2^t, ..., c_m^t\}$ and there is no overlap between classes of different tasks: $C^t \cap C^s = \emptyset$ for $t \neq s$.

Samples inside a task t can contain different data depending on the learning method. For supervised learning, each sample is a pair (x_i, y_i) where x_i is an image of class $y_i \in C^t$. In the semi-supervised setting, L samples have an assigned label y_i and U samples are unlabeled images x_i (usually $L_t << U_t$).

4.3.2 Motivation

Not all forgetting is catastrophic.

Semantic feature drift, as described in prior work [156], manifests when a model learns sequentially. In continual learning, the prototypes computed for task t may lose relevance for future tasks because the position of class c after training on task t differs from that in task $t-1$. This phenomenon is independent of any learning paradigm (whether supervised or unsupervised) and impacts all non-frozen models during incremental training.

To demonstrate the impact of feature drift on Class-Prototype (CP) methods, we conduct experiments using a ResNet-18 backbone trained in a Cold Start setting on 10-task CIFAR-100 and evaluate the performance of classes from the initial 3 tasks after training on all tasks. Fig. 4.1(a) illustrates the backbone trained with the supervised continual Learning without Forgetting (LwF) strategy [83], while Fig. 4.1(b) shows the backbone trained with the unsupervised continual learning (CaSSLe) method [43]. We use a Nearest Class Mean (NCM) [122] classifier to predict the labels of the test set for the classes encountered up to each task t.

In Fig. 4.1, we show the NCM accuracy using four types of prototypes:

- **Naive prototypes** (p_{naive}): These are prototypes without any update or correction. More specifically, prototypes of classes C^t are computed at the end of t and used as is afterward.
- **Corrected prototypes** ($p_{corrected}$): At the end of each task, all stored prototype positions are corrected using LDC (see Section 4.3.3), without the use of exemplars. This correction strategy is the main focus of this chapter.
- **Oracle prototypes** (p_{oracle}): The true position of all old class prototypes is computed by utilizing all the training data from old tasks at the end of t.
- **Jointly trained prototypes** (p_{joint}): The prototypes are computed using a model which is based on incremental joint training. This serves as the upper bound.

We observe that while a substantial gap in performance exists between the p_{naive} and p_{joint} accuracy in the two plots, which is generally referred to as catastrophic forgetting, a major part of this forgetting can be mitigated using p_{oracle}. This indicates that a significant amount of forgetting can be attributed to the drift in class prototypes which can be reduced by correcting the position of old prototypes. For example, for the case of task 1 and 2 (red lines) we can observe that the features that are discriminative for the task 1&2 are present in the backbone and have not been significantly impaired by the learning of subsequent tasks.

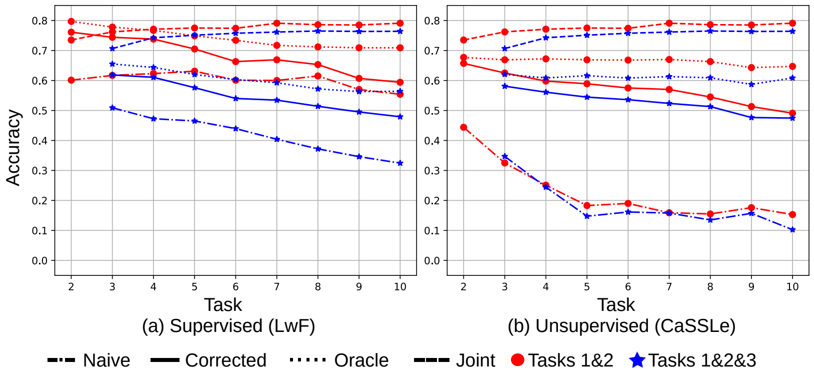


Figure 4.1: Last task accuracy of class-prototype accumulation strategies using an NCM classifier in the 10-task CIFAR-100 scenario. The figure row shows regularization-based methods (a) LwF and (b) CaSSLe. Here we show result for two settings: Tasks 1&2 shows how the performance on the first two tasks evolves while incrementally training all tasks. Analogously, Tasks 1&2&3 shows how the performance on the first three tasks evolves over training of all ten tasks.

However, regularization methods (like LwF and CaSSLe) are not enough for mitigating forgetting. This motivates us to explore drift compensation. We will demonstrate that correcting the prototypes with our method LDC improves the performance of p_{naive} to $p_{corrected}$ by big margins (around 38% for tasks 1&2&3 in unsupervised setting).

Limitations of Semantic Drift Compensation.

The closest methods to LDC rely on SDC. However, this method has an important weakness: it assumes a fixed local transformation for drift (namely locally it can be captured by translations). If semantic drift follows a different type of transformation, such as scaling, this assumption will lead to errors in the estimated drift.

To illustrate this point, Fig. 4.2 presents a toy example showcasing translations, rotations, and scaling applied to three sample distributions. In this experiment, we aim to estimate the true mean of C_1 after it has drifted by using the transformation observed in D_{t_2} as a reference. When only translations are applied (first row), SDC perfectly estimates the distribution's drift. However, under transformations that include rotations and scaling (second row), SDC incorrectly approximates the true mean of the distributions.

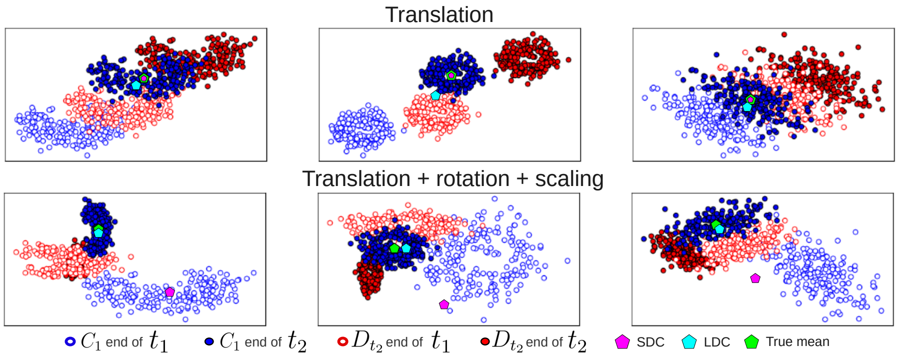


Figure 4.2: Feature drift estimation after applying random translations, rotations and scaling on three sample 2D distributions. We aim to estimate the true mean of C_1 at the end of t_2 using D_{t_2}. SDC assumes locally that transformations can be captured by translations. LDC can handle rotation and scaling in feature space. Note that LDC (cyan) more accurately approximates the real distribution mean (green).

4.3.3 Learnable Drift Compensation

LDC is applied at the end of each training session t. Let f_θ be a feature extractor parameterized by θ. To correct the semantic drift, we learn a forward projector p_F^t to map the features from the old feature space of f_θ^{t-1} to the new feature space of f_θ^t. In order to learn the drift in the feature space, we use the data of the current task D_t and minimize the mean squared error (MSE) between the projected features of f_θ^{t-1} and f_θ^t. Hence, the loss function to learn p_F^t is:

$$\mathcal{L} = \frac{1}{N}\sum_{i=1}^{N}(p_F^t(f_\theta^{t-1}(x_i)) - f_\theta^t(x_i))^2, \tag{4.1}$$

where f_θ^t and f_θ^{t-1} are frozen. After this optimization, the old prototypes P_{t-1}^c for all classes seen till task $t-1$ are updated using Eq. (4.2):

$$P_t^c = p_F^t(P_{t-1}^c). \tag{4.2}$$

This process is repeated each time f_θ is trained in a new task as shown in Fig. 4.3. Hence, if we perform class-incremental learning in T tasks, $T-1$ projectors are trained. We do not need to store any old images or features, or use any labels of the current data; we simply store only the old class prototypes and update them

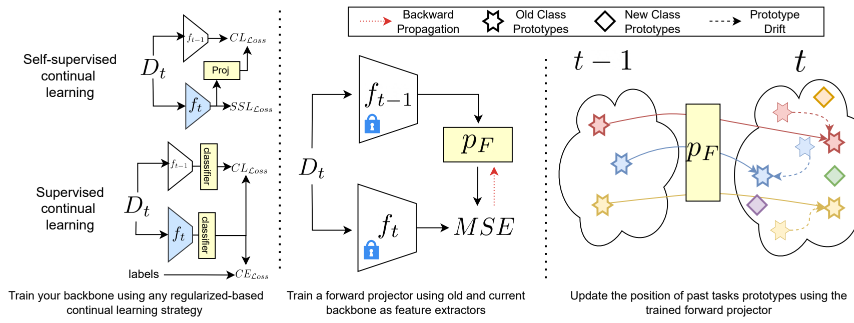

Backward Propagation
Old Class Prototypes
New Class Prototypes
Prototype Drift
Self-supervised continual learning
D_t
f_{t-1}
CL_{Loss}
Proj
f_t
SSL_{Loss}
Supervised continual learning
classifier
CE_{Loss}
labels
p_F
MSE
$t-1$
t
Train your backbone using any regularized-based continual learning strategy
Train a forward projector using old and current backbone as feature extractors
Update the position of past tasks prototypes using the trained forward projector

Figure 4.3: **Learnable Drift Compensation**. We train the model on the current task data using regularization-based, self-supervised or supervised continual representation learning methods. After training the new feature extractor f_t, we learn a forward projector p_F by minimizing the mean squared error between the projected features from f_{t-1} and the features from f_t. We use the learned projector to compensate for the drift of the old class prototypes in the new feature space.

using the learned current task projector. The projector p_F^t can be any trainable mapping function. However, we found that a linear layer performs best in this task (see section 4.4.4). We summarized the LDC algorithm in the supplementary material.

Nearest Class Mean (NCM) Classifier. NCM classifier [54, 70, 122, 156] has been found to be very effective for classification in CL settings. We use a NCM classifier on top of the trained feature extractor f_θ^t which classifies the test images x based on their distance to all the class prototypes P_t^c as follows:

$$y^* = \underset{y=1,\dots,Y^t}{\operatorname{argmin}} \| f_\theta^t(x) - P_t^y \|, \tag{4.3}$$

where Y^t is the set of classes seen up to task t and $P_t^y = p_F^t(P_{t-1}^y)\, \forall y \in Y^{t-1}$.

4.3.4 Continual Training Strategies

Our proposed method for drift compensation is general and can be combined with many existing regularization methods commonly used in CL. LDC can be applied at the end of training session t with the only requirement being access to the old feature extractor f_{t-1} and the old class prototypes P_{t-1}^c. A projector p_F^t must be optimized using Eq. (4.1) with D_t, and then old prototypes are updated using Eq. (4.2) and stored along with current ones. Finally, a new training session begins on D_{t+1} using any CL strategy (e.g., LwF). It is important to note that our method is independent of the CL strategy used to retain knowledge in the backbone, as LDC is a prototype correction method, not a CL approach. Here, we explain how to extend several existing methods with LDC.

Supervised CL.

Regularization-based methods retain knowledge while learning new concepts by preventing weights or activations from drifting far from the old model. For instance, LwF [83] aims to match the output of the previous model softmax layer $h_\theta^{t-1}(x)$ with the current model outputs $h_\theta^t(x)$ using knowledge distillation [63] as follows:

$$\mathscr{L} = \mathscr{L}_{ce}(h_t(x), y) + \lambda \mathscr{L}_{ce}(h_{t-1}(x), h_t(x)), \tag{4.4}$$

where $\mathscr{L}_{ce}$ refers to the cross-entropy loss and λ denotes the regularization strength. After the training session, the prototypes p_t^c of D_t are computed using the labels and added to the prototype pool for NCM classification. This work conducts experiments using LwF as it is a well-established and flexible method in the CL community. However, any exemplar-free supervised CL method can replace the LwF method. Regularization-based methods are particularly well-suited for LDC as they already maintain a copy of the old model at each task.

Self-supervised CL. Exemplar-free unsupervised CL methods are based on a projected regularization strategy at the feature level. These approaches perform self-supervised learning (SSL) in D_t while preserving similarity between old-fixed $f_{t-1}(D_t)$ and new-projected features $p(f_t(D_t))$. PFR [53] project features after the backbone and CaSSLe [43] after a post-backbone layer. POCON [52] uses two networks, one expert network performs SSL in D_t and the other network (main) distills the knowledge from the expert and the old main network using projected features from the current main network.

The CP accumulation strategy cannot be directly applied in exemplar-free self-supervised CL due to the absence of labels during training. However, if labels become available after training, we can compute prototypes, store them, and even correct them as in the supervised case. After the SSL CL session, we train a projector using the frozen SSL feature extractors and correct the old prototype positions accordingly.

Semi-supervised CL. If some labels are accessible during training, we operate within the semi-supervised learning setup. Here, we propose using LDC with exemplar-free self-supervised CL methods, which to the best of our knowledge firstly achieve the exemplar-free continual semi-supervised learning. To implement this, we follow a similar procedure as in the self-supervised CL case but utilize the available labels to compute prototypes. It is important to note that the performance of LDC remains independent of the number of available labels, meaning that the primary source of error would stem from the computation of the prototype, which typically exhibits low variability.

4.4 Experimental Results

4.4.1 Datasets

We use several publicly available datasets for our experiments. CIFAR-100 [76] contains 100 classes of 32×32 images with 500 training and 100 testing images per class. Tiny-ImageNet [80] contains 200 classes of 64×64 images with a total of 100K training and 10K testing images. ImageNet100 is a subset of one hundred classes from ImageNet [126] containing 224×224 images with a total of 130K training and 5K testing images. Stanford Cars [75] consists of 196 finegrained categories of cars and has 8144 images for training and also for testing. We equally split all the datasets into 10 tasks, which is different from the conventional Warm Start settings [54, 115, 156, 168, 169].

4.4.2 Supervised Continual Learning

Baseline methods. Since the existing methods were originally proposed for Warm Start settings, we re-implement all methods in Table 4.1 using PyCIL [165]. We implement a standard distillation baseline method, LwF [83] and also use NCM [122] classification naively. We use SDC [156] with the models trained with LwF. Note that SDC was originally proposed in Warm Start settings with feature distillation. Here, we adapt SDC for Cold Start settings with logits distillation (LwF). We include recent methods like PASS [168], FeTrIL [115], FeCAM [54] and EFC [90] in our comparison. While LwF, PASS and EFC are continual representation learning methods, FeCAM and FeTrIL freeze the model after the first task and continually learn the classifier. We do not compare to the methods [92, 132, 169, 170] proposed in Warm Start settings since they showed poor performance compared to FeTrIL and FeCAM. Following recent practice [56, 100, 161, 164] of using pre-trained ViTs for CIL, we use a ViT-B/16 [33] model pretrained on ImageNet21k [123] and present results with NCM, SDC and LDC in Table 4.2.

Training Settings. We implemented existing methods like LwF [83] (with NCM [122] and SDC [156]), PASS [168], FeTrIL [115] and FeCAM [54] and optimized them for

Table 4.1: Results in supervised settings for 10 tasks on CIFAR-100, Tiny-Imagenet, and ImageNet100. The mean and standard deviation using 5 different seeds are reported. Best results are in **bold** and second-best <u>underlined</u>. †: results excerpted from [90].

Method	**CIFAR-100**		**Tiny-ImageNet**		**ImageNet100**	
	A_{last}	A_{inc}	A_{last}	A_{inc}	A_{last}	A_{inc}
LwF+NCM [122]	40.5 ± 2.7	56.2 ± 3.6	28.6 ± 1.1	44.2 ± 1.3	44.0 ± 1.9	64.4 ± 0.4
LwF+SDC [156]	40.6 ± 1.8	56.2 ± 3.0	29.5 ± 0.8	43.8 ± 1.3	42.6 ± 1.9	$\underline{65.3} \pm 0.6$
PASS [168]	37.8 ± 0.2	52.3 ± 0.1	31.2 ± 0.4	45.3 ± 0.7	38.7 ± 0.5	55.5 ± 0.5
FeTrIL [115]	37.0 ± 0.6	52.1 ± 0.5	24.4 ± 0.6	38.5 ± 1.1	40.7 ± 0.5	56.3 ± 0.8
FeCAM [54]	33.1 ± 0.9	48.1 ± 1.3	24.9 ± 0.5	38.6 ± 1.3	42.4 ± 0.9	57.9 ± 1.5
EFC [90] †	$\underline{43.6} \pm 0.7$	$\underline{58.6} \pm 0.9$	$\underline{34.1} \pm 0.8$	$\mathbf{48.0} \pm 0.6$	$\underline{47.3} \pm 1.4$	59.9 ± 1.4
LwF+LDC	$\mathbf{45.4} \pm 2.8$	$\mathbf{59.5} \pm 3.9$	$\mathbf{34.2} \pm 0.7$	$\underline{46.8} \pm 1.1$	$\mathbf{51.4} \pm 1.2$	$\mathbf{69.4} \pm 0.6$

Table 4.2: Performance on Stanford Cars (10 tasks of 20 classes each) with pre-trained ViT-B/16.

Method	$t1$	$t3$	$t5$	$t7$	$t10$
FT+NCM	85.8	73.3	63.2	58.0	53.4
FT+SDC	85.8	73.5	63.5	57.8	53.3
FT+LDC	85.8	76.0	71.3	67.0	**62.9**

small-start settings. We use the PyCIL [165] framework. Here we share the details of the settings for reproducibility.

For CIFAR-100, we use the CIFAR-10 policy augmentations from PyCIL and use it for all the methods to have a fair comparison. For all datasets, we follow the common practice of using random crop and random horizontal flip for the training set.

- LwF: For all datasets in the first task, we follow PyCIL and use a learning rate of 0.1, momentum of 0.9, weight decay of 0.0005 and train with a batch size of 128 for 200 epochs. The learning rate is reduced by 10 after 60 , 120 and 180 epochs. For all new tasks in CIFAR-100 and ImageNet100, we use a learning rate of 0.05. For TinyImageNet, we use a learning rate of 0.001. We train for 100 epochs and reduce the learning rate by 10 after 45 and 90 epochs. The temperature scale is set to 2. The LwF regularization strength is set to 10 for CIFAR-100 and TinyImageNet and 5 for ImageNet100. We use the models trained with LwF and use them with a NCM classifier and SDC. For FeTrIL and FeCAM, we train the first task in the same way as LwF.

- SDC: For SDC [156], we use $\sigma = 0.3$, which is the standard deviation of the Gaussian kernel for estimating the weights of the drift vectors. For ImageNet100, we set $\sigma = 1.0$.
- PASS: Following the PyCIL implementation of PASS, we set $\lambda_{fkd} = 10$ and $\lambda_{proto} = 10$.
- FeTrIL: For all incremental tasks, we follow the same settings as the FeTrIL implementation from PyCIL [165].
- FeCAM: FeCAM trains the first task and use the frozen model for all subsequent tasks. FeCAM use the stored prototypes and distribution statistics (covariance matrix) for all old classes. Following the implementation from [54], we use the covariance shrinkage hyperparameters of (1,1) and 0.5 as the Tukey's normalization value.

Training procedure. We train a ResNet-18 [62] using an SGD optimizer. For all datasets and methods we train the first task for 200 epochs and the incremental tasks for 100 epochs. In the first task for CIFAR-100, we use a learning rate of 0.1, momentum of 0.9, weight decay of 0.0005, and the learning rate is reduced by 10 after 60, 120, and 160 epochs following the default settings of PyCIL. In the incremental tasks, we use a learning rate of 0.05, which is reduced by 10 after 45 and 90 epochs. For LwF models, we use $\lambda = 10$. For LDC, we learn a linear layer using Adam optimizer and a learning rate of 0.001 for 20 epochs using all the data in the current task. We report all details about experimental settings in the supplementary materials. For table 4.2, we use the same settings and hyperparameters as [161] for finetuning in new tasks and learn a two-layer MLP using Adam optimizer and a learning rate of 0.005 for 100 epochs.

Results. We report the last task accuracy as well as the average incremental accuracy with 5 random seeds and class sequences in Table 4.1. We observe that while the recently proposed methods like PASS [168], FeTrIL [115], and FeCAM [54] performed well in Warm Start settings, LwF with an NCM classifier outperformed these methods in Cold Start settings. While SDC was effective in the originally proposed Warm Start settings, we see that with higher feature drift in our settings, SDC does not improve the performance in most cases. We show that LDC with LwF outperforms all these baselines significantly across all datasets. In CIFAR-100, LDC outperforms SDC by 4.8% and by 3.3% on last task and incremental accuracy respectively. For Tiny-ImageNet, LDC performs similar to the recent EFC method

Table 4.3: Results in semi-supervised settings for 10 tasks on CIFAR-100. The mean and standard deviation using 3 different seeds are reported. The best results for the proposed setting are in **bold**.

Method	**Ex Free**	**Semi-Sup**	**100%**		
LwF+NCM	✓	✗	40.5±2.7		
PASS	✓	✗	37.8±0.2		
			0.8%	**5%**	**25%**
PASS	✓	✓	5.91	15.13	15.84
NNCSL (500)	✗	✓	27.4±0.51	31.4±0.40	35.3±0.30
NNCSL (0)	✓	✓	8.7±0.63	9.0±0.42	9.32±0.39
PFR+naive	✓	✓	20.5±0.44	24.8±0.35	25.7±0.12
CaSSLe+naive	✓	✓	19.4±0.57	23.2±0.21	23.6±0.13
POCON+naive	✓	✓	20.1±0.23	21.7±0.27	22.4±0.31
PFR+SDC	✓	✓	16.2±0.66	22.5±0.57	24.8±0.12
CaSSLe+SDC	✓	✓	22.1±0.69	25.8±0.09	26.5±0.04
POCON+SDC	✓	✓	21.8±0.82	25.7±0.23	26.7±0.09
PFR+LDC	✓	✓	18.6±0.20	29.3±0.12	33.4±0.06
CaSSLe+LDC	✓	✓	**27.0**±0.60	**32.8**±0.30	**35.0**±0.21
POCON+LDC	✓	✓	26.1±0.74	31.1±0.62	33.8±0.25

which is also designed for the cold start setting. However, on the more challenging ImageNet100 LDC improves over EFC by a large margin of almost 4% on last task accuracy. Finally, using pre-trained ViTs on Stanford Cars, we observe that LDC improves over SDC by 9.6% after the last task, see Table 4.2.

4.4.3 Semi-supervised Continual Learning

Baseline methods. Since the proposed method is the first exemplar-free semi-supervised CL method, we can only compare it against drift correction methods (SDC). Nevertheless, we present results for the most close approaches. The naive approach achieves the lowest possible accuracy, no prototype correction is performed. SDC [156] is used as a comparison for drift correction. PASS [168] is not a semi-supervised method, but we present our results with all labels (as a reference) and with reduced training data corresponding to the proposed label sets. NNCSL [71] is a semi-supervised method that uses exemplars; we present here results with the same number of exemplars proposed by the authors (as a reference) and a reduced version. Removing exemplars from exemplar-based methods or reducing data to comply with our setup is not fair, and we present these results as a reference to showcase the performance of our approach. We use the code and hyperparameters provided by the authors.

Training Settings. We use the existing CL exemplar-free methods namely PFR [53], CaSSLE [43], and POCON [52]. We utilize the code provided for each method. Here we include additional details of the settings for reproducibility.

For each dataset, we follow the image augmentation pipeline of SimCLR [22]. All the methods use Barlow Twins [157] as base self-supervised learning strategy.

- PFR: We conduct our experiments with 500 epochs per task on CIFAR100 and 400 for ImageNet100. As for CL hyperparameters, $\lambda = 25$ for both datasets; the learning rate, optimizer, and training schedules are the same as proposed by the authors.
- CaSSLe: We use the same training epochs as PFR. We strictly follow the hyperparameters shared by the authors.
- POCON: For both datasets, we employ the configuration CopyOP. The hyperparameters setup is the same as the authors proposed for the 10-task partition of each dataset.

Training procedure. We train a ResNet-18 [62] using PFR [53], CaSSLe [43], and POCON [52] as SSL CL strategies. We use the same code, training procedure, and hyperparameters provided by the authors in each method. At the end of each task, we compute the prototypes using the available labels: 0.8%, 5%, 25% for CIFAR-100 and 1%, 5%, 25% for ImageNet100. For LDC, we train a linear projector using the Adam optimizer (learning rate $5e-3$) for 100 epochs using all the data in the task. We report the mean last task accuracy in the test set using an NCM classifier.

Results. Table 4.3 presents the results of 10 task split of CIFAR-100. LDC presents the best results for all the pure semi-supervised exemplar-free settings. For 5% and 25% settings, it is reaching the same accuracy as the exemplar-based method NNCSL. Furthermore, LDC is only 3% below the performance of the fully supervised version of PASS. LDC surpasses SDC by at least 7% in all the base SSL CL methods in the pure semi-supervised exemplar-free setting.

Table 4.4 presents the results of 10 task split of ImageNet100. As in CIFAR-100, LDC performs best for the proposed setting, surpassing SDC by at least 7% in all the SSL CL backbones. However, in this dataset, the replay-based method NNCSL is better with the same number of exemplars proposed by the authors. Nevertheless, a reduction in the number of exemplars in NNCSL leads to a significant performance drop, ending below our proposed exemplar-free method.

Table 4.4: Results in semi-supervised settings for 10 tasks on ImageNet100. The mean and standard deviation using 3 different seeds are reported. The best results for the proposed setting are in **bold**.

Method	**Ex Free**	**Semi-Sup**		**100%**	
LwF+NCM	✓	✗		44.0±1.9	
PASS	✓	✗		38.7±0.5	
			1%	**5%**	**25%**
PASS	✓	✓	1.76	10.36	23.9
NNCSL (5120)	✗	✓	35.4±0.21	41.6±0.12	47.1±0.08
NNCSL (500)	✗	✓	9.93±0.22	15.1±0.15	16.6±0.1
PFR+naive	✓	✓	19.6±0.16	21.8±0.38	22.0±0.12
CaSSLe+naive	✓	✓	25.37±0.27	26.9±0.25	27.36±0.11
POCON+naive	✓	✓	24.2±0.32	26.1±0.22	25.8±0.20
PFR+SDC	✓	✓	17.2±0.55	19.1±0.16	19.8±0.08
CaSSLe+SDC	✓	✓	22.2±0.03	25.9±0.23	26.6±0.01
POCON+SDC	✓	✓	21.8±0.08	25.1± 0.30	25.7±0.03
PFR+LDC	✓	✓	26.5±0.09	32.4±0.35	33.5±0.07
CaSSLe+LDC	✓	✓	**29.5**±0.58	**33.9**±0.15	**35.8**±0.39
POCON+LDC	✓	✓	28.3±0.37	33.2±0.30	34.0±0.28

4.4.4 Ablation Studies

In this section we report on a range of ablation studies for LDC. Unless specified, the training and evaluation procedure are the same as in Sections 4.4.2 and 4.4.3. We provide more ablation experiments on comparison of LDC with NME and using stored features for learning the projector, in the supplementary materials.

Architecture of projector. To ablate the complexity of the proposed projector p_F, Table 4.5 presents experiments for LwF+LDC(supervised CL) and CaSSLe+LDC (semi-supervised CL - 25% of labels). We show LDC performance with a linear layer, with and without bias, with an activation layer ReLU after the linear layer and also using a two-layer MLP. We observe that a single linear layer works better in both settings.

Exemplars versus LDC for prototype correction. We compare the performance of LDC against Nearest-Mean of Exemplars (NME) which directly uses exemplars for old class prototype compensation and is explored in exemplar-based methods [35, 122]. While it is easy to predict the updated positions of the old prototypes using

Table 4.5: Ablation study for different complexities of the proposed projector p_F.

Method	Linear	Linear + bias	Linear + ReLu	MLP
LwF + LDC	45.4±2.8	45.2±3.0	41.2±2.1	43.7±2.3
CaSSLe + LDC	35.0±0.2	34.3±0.2	29.1±0.5	34.9±0.4

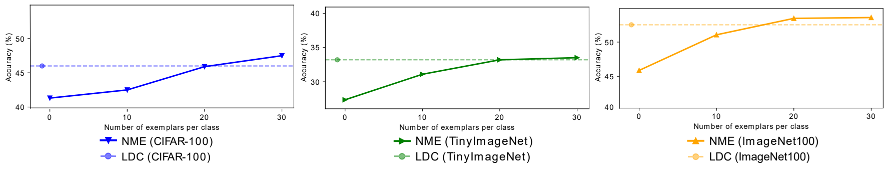


Figure 4.4: Comparison of LDC with NME for varying memory size in the supervised settings on CIFAR-100, Tiny-ImageNet and ImageNet100.

original samples of old classes by NME, it is very challenging without exemplars in our settings. We compare the last task accuracy of LDC (same setting as in Table 4.1, with LwF+NCM) with NME which stores a memory of old class samples. We show in Fig. 4.4 that LDC is on par with using 20 exemplars per class for NME for CIFAR-100 and Tiny-ImageNet. For ImageNet100, LDC is marginally less than NME with 20 exemplars per class. This demonstrates that LDC is able to effectively correct the prototypes without access to exemplars and yet is on par with NME using exemplars.

Drift Compensation Analysis. In order to visualize the drift compensation using LDC, Fig. 4.5 presents a comparison between SDC and LDC. We update all the old prototypes using both SDC and LDC and then compute the cosine distance of the updated prototype to the oracle prototype (computed using all old data for analysis) in the new feature space. We plot the distributions over all old class prototype distances after alternate tasks till the lats task. We observe that SDC struggles to estimate the prototypes even after the first increment (task 2) and is centered around distance of 0.1. As we move to the last task, SDC gets even worse with a higher deviation which implies increasing distance of updated prototypes from their oracle positions. On the other hand, LDC is more robust and the distribution is centered between 0 and 0.05 even after the last task, with much lower deviation. This analysis suggests that the LDC predicts the old prototypes very close to their oracle positions and thus better tracks the prototypes position when moving from one feature space to another.

Feature Storage. In this work, we update only prototypes (one sample mean

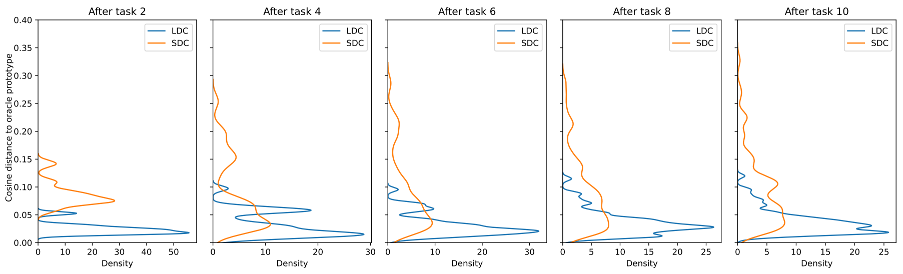


Figure 4.5: Drift compensation analysis using SDC and LDC in supervised settings on CIFAR-100. We compute the cosine distance between the updated and oracle prototypes. We plot the distributions of the cosine distances for all old classes using both methods after alternate tasks. While SDC fails to estimate the prototypes closer to the oracle (with increasing distance for many classes), LDC predictions are very close to the oracle prototypes (mean of distributions using LDC is close to 0 even after the last task and has low standard deviation).

Table 4.6: Accuracy comparison storing and projecting N features per class against storing and projecting only class prototypes.

Method	**Mean**	$N = 5$	$N = 50$	$N = 100$	$N = 500$
CaSSLe + LDC	35.0 ± 0.2	21.1 ± 0.01	32.9 ± 0.03	33.9 ± 0.02	35.05 ± 0.04

per class), but one can argue that storing a set of features per class is better than a single class mean. In Table 4.6, instead of projecting a prototype per class, we project N stored features and use them to compute the class prototype at the classification stage. The results show that storing and projecting features do not present a significant advantage over projecting the class means.

4.5 Conclusions

We showed that performance degradation in exemplar-free CP accumulation methods is largely due to feature drift. We proposed a simple yet effective prototype correction method called Learnable Drift Compensation (LDC). LDC learns a projector that maps between the feature spaces of the consecutive tasks using only the data available at each task along with current and previous model. The trained projector is used at the end of each task to correct the stored old prototype positions. We demonstrate the efficacy of LDC in correcting the drift of any moving backbone by using it on top of already established exemplar-free CL strategies. Extensive experiments on CIFAR-100, Tiny-ImageNet, and ImageNet100 show that LDC outperforms previous drift correction methods in supervised and semi-supervised settings.

Limitations. While the learned projection p_f between tasks is effective for correcting drift, it's not flawless. The gap between $p_{corrected}$ and p_{oracle} in Fig. 4.1 highlights this difference. Without exemplars, p_f only learns the projection from current task data. Consequently, due to biases in the current data, updates to some old class prototypes may not be perfect. We would like to explore methods to further minimize or eliminate this gap in our future work.

5 Planckian Jitter: countering the color-crippling effects of color jitter on self-supervised training*

5.1 Introduction

Self-supervised learning enables the learning of representations without the need for labeled data [32, 34]. Several recent works learn representations that are invariant with respect to a set of data augmentations and have obtained spectacular results [15, 24, 57], significantly narrowing the gap with supervised learned representations. These works vary in their architectures, learning objectives, and optimization strategies, however they are similar in applying a common set of data augmentations to generate different image views. These algorithms, while learning to map these different views to the same latent representation, learn rich semantic representations for visual data. The set of transformations (data augmentations) used induces invariances that characterize the learned visual representation.

Before deep learning revolutionized the way visual representations are learned, features were handcrafted to represent various properties, leading to research on shape [87], texture [95], and color features [44, 49]. Color features were typically designed to be invariant to a set of scene-accidental events such as shadows, shading, and illuminant and viewpoint changes. With the rise of deep learning, feature representations that simultaneously exploit color, shape, and texture are learned implicitly and the invariances are a byproduct of end-to-end training [76]. Current approaches to self-supervision learn a set of invariances implicitly related to the applied data augmentations.

In this chapter, we focus on the currently de facto choice for color augmentations. We argue that they seriously cripple the color quality of learned representations and we propose an alternative, physics-based color augmentation. Figure 5.1 (left) illustrates the currently used color augmentation on a sample image. It is clear that the applied color transformation significantly alters the colors of the original image, both in terms of hue and saturation. This augmentation results in a representation that is invariant with respect to surface reflectance – an invariance beneficial

*This chapter is based on research which was performed together with Simone Zini and is based on the publication at Eleventh International Conference on Learning Representations (ICLR 2023) [171]

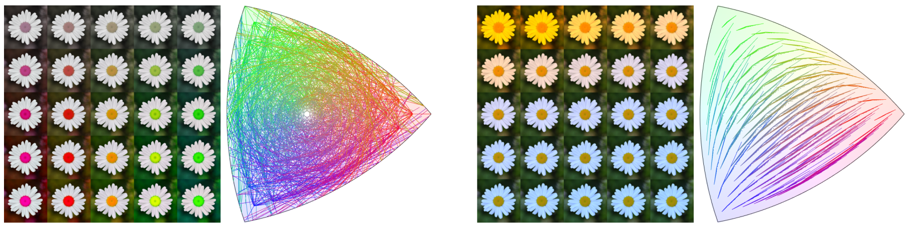


Figure 5.1: Default color jitter (left) and Planckian Jitter (right). Augmentations based on default color jitter lead to unrealistic images, while Planckian Jitter leads to a set of realistic ones. The ARC chromaticity diagrams for each type of jitter are computed by sampling initial RGB values and mapping them into the range of possible outputs given by each augmentation. These diagrams show that Planckian Jitter transforms colors along chromaticity lines occurring in nature when changing the illuminant, whereas default color jitter transfers colors throughout the whole chromaticity plane.

for recognizing classes whose surface reflectance varies significantly, for example many man-made objects such as cars and chairs. However, such invariance is expected to hurt performance on downstream tasks for which color is an important feature, like natural classes such as birds or food. One of the justifications is that without large color changes, mapping images to the same latent representation can be purely done based on color and no complex shape or texture features are learned. However, as a result the quality of the color representation learned with such algorithms is inferior and important information on surface reflectance might be absent. Additionally, some traditional supervised learning methods propose domain-specific variations of color augmentation [46, 152].

In this chapter we propose an alternative color augmentation (Figure 5.1, right) and we assess its impact on self-supervised learning. We draw on the existing color imaging literature on designing features invariant to illuminant changes commonly encountered in the real world [44]. Our augmentation, called *Planckian Jitter*, applies physically-realistic illuminant variations. We consider the illuminants described by Planck's Law for black-body radiation, that are known to be similar to illuminants encountered in real-life [141]. The aim of our color augmentation is to allow the representation to contain valuable information about the surface reflectance of objects – a feature that is expected to be important for a wide range of downstream tasks. Combining such a representation with the already high-quality shape and texture representation learned with standard data augmentation leads to a more complete visual descriptor that also describes color.

Our experiments show that self-supervised representations learned with Planckian Jitter are robust to illuminant changes. In addition, depending on the importance of color in the dataset, the proposed Planckian jitter outperforms the default color jitter. Moreover, for all evaluated datasets the combination of features of our new data augmentation with standard color jitter leads to significant performance gains of over 5% on several downstream classification tasks. Finally, we show that Planckian Jitter can be applied to several state-of-the-art self-supervised learning methods.

5.2 Background and related work

Self-supervised learning and contrastive learning. Recent improvements in self-supervision learn semantically rich feature representations without the need for labelled data. In SimCLR [22] similar samples are created by augmenting an input image, while dissimilar are chosen randomly [22]. To improve efficiency, MoCo [61] and its enhanced version [23] use a memory bank for learned embeddings which makes sampling efficient. This memory is kept in sync with the rest of the network during training via a momentum encoder. Several methods do not rely on explicit contrastive pairs. BYOL uses an asymmetric network incorporating an additional MLP predictor between the outputs of the two branches [57]. One of the branches is kept "offline" and is updated by a momentum encoder. SimSiam defines a simplified solution without a momentum encoder [24]. It obtains similar high-quality results and does not require a large minibatch size, in contrast to other methods.

We use the SimSiam method to verify our proposed color augmentation (we also apply it to SimCLR [22] and Barlow Twins [157] in the experiments). The main component of the network is a CNN-based asymmetric siamese image encoder. One branch has an additional MLP predictor whose output aims to be as close as possible to the other (Figure 5.2). The second branch is not updated during backpropagation. A negative cosine loss function is used:

$$\mathscr{L} = \frac{1}{2}\left[\mathscr{D}(p_1, \text{stopgrad}(z_2)) + \mathscr{D}(p_2, \text{stopgrad}(z_1))\right] \tag{5.1}$$

$$\mathscr{D}(p_A, z_B) = -\frac{p_A}{\left\|p_A\right\|_2} \cdot \frac{z_B}{\|z_B\|_2}, \tag{5.2}$$

where z_1, z_2 are representations for two different augmented versions, x_1 and x_2, of the same image x.

The MLP is applied in alternation to either z_1 or z_2, producing respectively p_1 or p_2. Note that Figure 5.2 only shows an instance for x_1 and does not show p_2. The

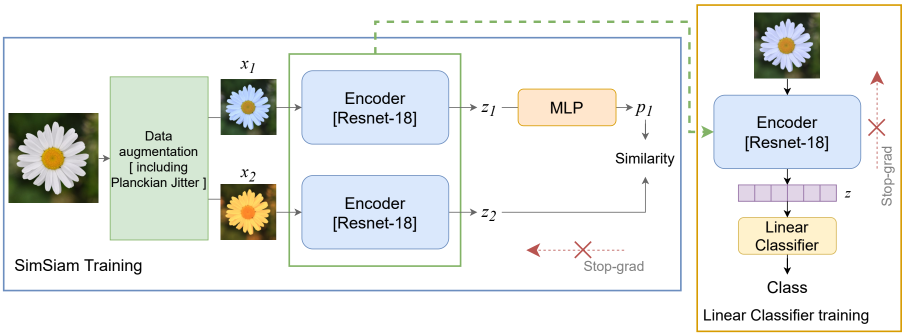


Figure 5.2: SimSiam training procedure exploiting Planckian-based data augmentation (left), and fine-tuning the linear classifier using the trained encoder (right).

stopgrad(·) operation blocks the gradient during the backpropagation. In SimSiam no contrastive term is used and only similarity is enforced during learning.

Data augmentation. Data augmentation plays a central role in the self-supervised learning process. Authors [22] and [157] discuss the importance of the different data augmentations. A set of well-defined transformations was proposed for SimCLR [22]. This set is commonly accepted and used in several later works. The augmentations include: rotation, cutout, flip, color jitter, blur and Grayscale. These operations are randomly applied to an image to generate the different views x_1, x_2 from which are extracted the features z_1 and z_2 used in the self-supervision loss in Eq. 5.2. Applied to the same image, contrastive-like self-supervision learns representations invariant to such distortions.

This multiple view creation is task-related [138], however color jittering operating on hue, saturation, brightness and contrast, is one of the most important ones in terms of overall usefulness of the learned representation for downstream tasks [22, 157]. Color jitter induces a certain level of color invariance (invariance to hue, saturation, brightnesss and contrast) which are consequently transferred to the downstream task. As a consequence, we expect these learned features to underperform on downstream tasks for which color is crucial. [151] were the first to point out that the imposed invariances might not be beneficial for downstream tasks. As a solution, they propose to learn different embedding spaces in parallel that capture each of the invariances. Differently than them, we focus on the color distortion and propose a physics-based color augmentation that allows learning invariance to physically realistic color variations.

Color imaging has a long tradition in research on color features invariant to scene-accidental events such as shading, shadows, and illuminant changes [44, 49]. Invariant features were found to be extremely beneficial for object recognition. The invariance to hue and saturation changes induced by color jitter, however, is detrimental to object recognition for classes in which color characteristics are fundamentally discriminative. Therefore, in this chapter we revisit early theory on illuminant invariance [44] to design an improved color augmentation that induces invariances common in the real world and that, when used during self-supervised learning, does not damage the color quality of the learned features.

5.3 Methodology

The image transformations introduced by default color jitter creates variability in training data that indiscriminately explores all hues at various levels of saturation. The resulting invariance is useful for downstream tasks where chromatic variations are indeed irrelevant (e.g. car color in vehicle recognition), but is detrimental to downstream tasks where color information is critical (e.g. natural classes like birds and vegetables). The main motivation for applying strong color augmentations is that this it leads to very strong shape and texture representations. Indiscriminately augmenting color information in the image requires that the representation solve the matching problem using shape [22]†.

As an alternative to color jitter, we propose a physics-based color augmentation that mimics color variations due to illuminant changes commonly encountered in the real world. The aim is to reach a representation that does not have the color crippling effects of color jitter, and that better describes classes for which surface reflectance is a determining feature. When combined with default color jitter, this representation should also provide a high-quality shape/texture and color representation.

5.3.1 Planckian Jitter

We call our color data augmentation procedure *Planckian Jitter* because it exploits the physical description of a black-body radiator to re-illuminate training images within a realistic illuminant distribution [44, 141]. The resulting augmentations are more realistic than those of the default color jitter (see Fig. 5.1). The resulting learned, self-supervised feature representation is thus expected to be robust to illumination changes commonly observed in real-world images, while simultaneously maintaining the ability to discriminate the image content based on color information.

†This is pointed out in the discussion of Figure 5 in [22]

Given an input RGB training image I, our Planckian Jitter procedure applies a chromatic adaptation transform that simulates realistic variations in the illumination conditions. The data augmentation procedure is as follows:

1. we sample a new illuminant spectrum $\sigma_T(\lambda)$ from the distribution of a black-body radiator;
2. we transform the sampled spectrum $\sigma_T(\lambda)$ into its sRGB representation $\rho_T \in \mathbb{R}^3$;
3. we create a jittered image I' by reilluminating I with the sampled illuminant ρ_T; and
4. we introduce brightness and contrast variation, producing a Planckian-jittered image I''.

A radiating black body at temperature T can be synthesized using Planck's Law [4]:

$$\sigma_T(\lambda) = \frac{2\pi hc^2}{\lambda^5(e^{\frac{hc}{kT\lambda}} - 1)} \text{ W/m}^3, \tag{5.3}$$

where $c = 2.99792458 \times 10^8$ m/s is the speed of light, $h = 6.626176 \times 10^{-34}$ Js is Planck's constant, and $k = 1.380662 \times 10^{-23}$ J/K is Boltzmann's constant. We sampled T in the interval between $3000K$ and $15000K$ which is known to result in a set of illuminants that can be encountered in real life [141]. Then, we discretized wavelength λ in 10nm steps ($\Delta\lambda$) in the interval between 400nm and 700nm; The resulting spectra are visualized in Figure 5.3 (left).

The conversion from spectrum into sRGB is obtained according to [150]:

1. we first map the spectrum into the corresponding XYZ stimuli, using the 1931 CIE standard observer color matching functions $c^{\{X,Y,Z\}}(\lambda)$, in order to bring the illuminant into a standard color space that represents a person with average eyesight;
2. we normalize this tristimulus by its Y component, convert it into the CIE 1976 L*a*b color space, and fix its L component to 50 in a 0-to-100 scale, allowing us to constrain the intensity of the represented illuminant in a controlled manner as a separate task; and
3. we then convert the resulting values to sRGB, applying a gamma correction and obtaining $\rho_T = \{R, G, B\}$; the resulting distribution of illuminants is visualized with the Angle-Retaining Chromaticity diagram [12] in Figure 5.3 (right).

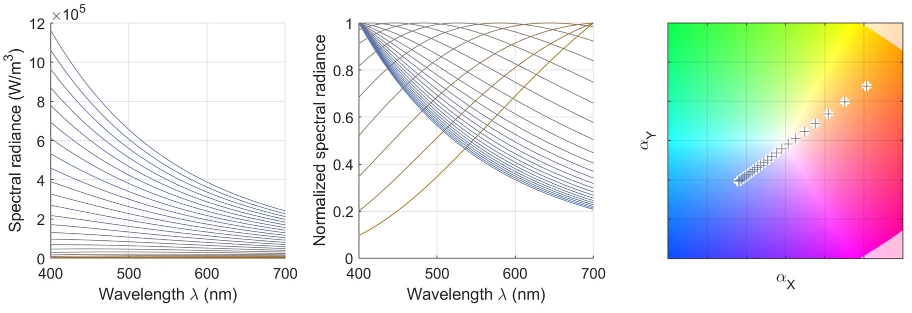


Figure 5.3: Spectral power distributions (left) and corresponding ARC chromaticities (right) of the sampled black body radiator, used to generate Planckian jittering. The illuminants are sampled from the distribution of a black body radiator, with correlated color temperature T in the interval between $3000K$ and $15000K$. The resulting spectra are visualized on the left and in the middle, while the resulting distribution of illuminants is visualized in the Angle-Retaining Chromaticity diagram on the right.

All color space conversions assume a D65 reference white, which means that a neutral surface illuminated by average daylight conditions would appear achromatic. Once the new illuminant has been converted in sRGB, it is applied to the input image I by resorting to a Von-Kries-like transform [144] given by the following channel-wise scalar multiplication:

$$I'^{\{R,G,B\}} = I^{\{R,G,B\}} \cdot \{R,G,B\}/\{1,1,1\}, \tag{5.4}$$

where we assume the original scene illuminant to be white (1,1,1). Finally, brightness and contrast perturbations are introduced to simulate variations in the intensity of the scene illumination:

$$I'' = c_B \cdot c_C \cdot I' + (1-c_C) \cdot \mu\left(c_B \cdot I'\right), \tag{5.5}$$

where $c_B = 0.8$ and $c_C = 0.8$ represent, respectively, brightness and contrast coefficients, and μ is a spatial average function.

5.3.2 Color selectivity index

Color selectivity is defined by [120] as the property of a neuron that activates strongly when a specific color appears in the input image, and does not when the color is absent. It is computed by estimating the ratio between the neuron's global activation with color input images and the global activation with corresponding grayscale images:

$$\alpha(n^{L,i}) = 1 - \frac{\sum_{j=1}^{N} w'_{j,i,L}}{\sum_{j=1}^{N} w_{j,i,L}}. \tag{5.6}$$

Here $w_{j,i,L}$ refers to the activation of an image patch j for the i-th neuron $n^{L,i}$ at layer L, normalized for the maximum activation value across all possible image patches. $w'_{j,i,L}$ is the equivalent formulation for a grayscale version of the images. The set of considered image patches is restricted to the top-N regions from a given dataset that maximally activate the neuron of interest.

We can distinguish between neurons that are colorblind or neurons that highly rely on color information by looking at the α value obtained: an α value more than 0.25 means that the neuron is high color selective, while an alpha value less than 0.1 means that the neuron is basically colorblind. These thresholds were selected based on the analysis by [120]. We collected alpha values for the neurons in the last layer of the encoders trained with different data augmentation configurations in order to compare the models sensitivity to color and how it changes in relation to the training procedure adopted.

5.3.3 Complimentarity of shape, texture and color representations

The self-supervised learning paradigm involves a pretraining phase that relies on data augmentation to produce a set of features with certain invariance properties. These features are then used as the representation for a second phase, where we learn a given supervised downstream task. The default color jitter augmentation generates features that are strongly invariant to color information, resulting in high-quality representations of shape and texture, but that is an inferior descriptor of surface reflectances (i.e. the color of objects). Our augmentation based on Planckian Jitter (see Figure 5.1) is based on transformations mimicking the physical color variations in the real world due to illuminant changes. As a result, the learned representation yields a high-quality color description of scene objects (this is also verified in section 5.4.9). However, it likely leads to a drop in the quality of the shape and texture representation (since color can be used to solve cases where previously shape/texture were required). To exploit the complimentarity of the two representations, we propose to learn both – one with color jitter and one with Planckian Jitter – and to then concatenate the results in a single representation vector (of 1024 dimensions, i.e. twice the original size of 512). We call this *Latent space combination (LSC)*.

5.4 Experimental results

In this section, we analyze the color sensitivity of the learned backbone networks, verify the superiority of the proposed color data augmentation method compared to the default color jitter on color datasets, and evaluate the impact on downstream classification tasks. We report additional results on computational time of the proposed Planckian augmentation in section 5.4.7.

5.4.1 Training and evaluation setup

We perform unsupervised training on two datasets: CIFAR-100 [76] (32×32) and ImageNet (224×224). [‡] We slightly modify the ResNet18 architecture to accommodate 32×32 images: the kernel size of the first convolutional was reduced from 7×7 to 3×3 and the first max pooling layer was removed. SimSiam training was performed using Stochastic Gradient Descent with a starting learning rate of 0.03, a cosine annealing learning rate scheduler, and mini-batch size of 512 (as in original SimSiam work by [24]). For the training on the 1000-class ImageNet training set, we follow the same procedure as [24] with ResNet50.

The linear classifier training at resolution 32×32 was performed on CIFAR-100 and FLOWERS-102 [105]. CIFAR-100 is used as a baseline for the classification task. The linear classifier training for CIFAR-100 is done with Stochastic Gradient Descent for 500 epochs with a starting learning rate 0.1, a cosine annealing learning rate scheduler, and mini-batch size of 512. The FLOWERS-102 dataset with 102 classes was selected to assess the quality of the features extracted in scenarios where color information plays an important role. Images from FLOWERS-102 are resized to 32×32 pixels to match the input dimensions of the pretrained model. Here we used the Adam optimizer with initial learning rate of 0.03.

For training linear classifiers at resolution 224×224 for downstream tasks we follow the evaluation protocol of [24]. We use six different datasets: IMAGENET, FLOWERS-102, the fine-grained VEGFRU [128], CUB-200 [147], T1K+ [26], and USED [1], all resized to 224×224 pixels. More details about the datasets are provided in section 5.4.4. In the case of CUB-200, each image was cropped using the bounding boxes given in the dataset annotations. For T1K+, we used the 266 class

[‡]We conduct the investigative part of our research in an agile manner on low-resolution images, then transfer the most significant configurations to a higher-resolution, to ether confirm or refute the initial hypotheses.

labeling to train and test the linear classifier.

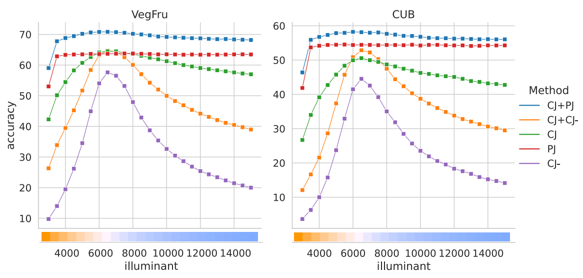
To assess the impact of color data augmentations we define six different configurations:

- *Default Color Jitter (CJ):* the default configuration, as used in SimSiam and SimCLR, uses both Random Color Jitter and Random Grayscale operations.
- *Default Color Jitter w/o Grayscale (CJ-)*: same as *Default*, without Random Grayscale.
- *Planckian Jitter (PJ)*: uses the complete proposed Planckian Jitter operating on chromaticy, brightness, and contrast aspects of the images. No Random Grayscale is applied.
- *LSC Default Color Jitter + Planckian Jitter ([CJ,PJ]*: This latent space combination (simple concatenation of representations) combines the default color jitter with our Planckian jitter. It allows evaluation of the complimentary nature of the representations.
- *LSC Default Color Jitter + Default Color Jitter w/o Grayscale ([CJ,CJ-])*: We combine the default color jitter with a version without the Grayscale augmentation, since this representation is also expected to result in a better color representation.
- *LSC of two Default Color Jitter Models ([CJ,CJ])*: We also show results of simply concatenating two independently trained models (trained from different seeds) with default color jitter (an ensemble of two models).

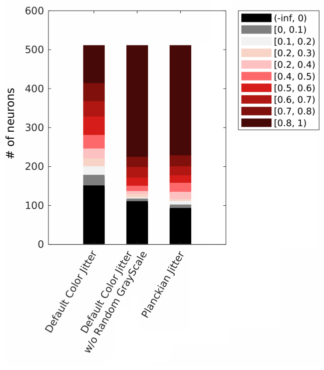
In all experiments, these are combined with the other default augmentations (crop, flip, and blur).

5.4.2 Color sensitivity analysis

We perform a robustness analysis on the VegFru and CUB-200 datasets with realistic illuminant variations, and analyzed sensitivity to color information. This experiment is driven by two motivations: to verify that we obtain invariance to the transformation applied during training, and to characterize the degradation of different non-Planckian training modalities. We assume as reference point the D65 illuminant, which for the purpose of this test is considered the default illuminant in every image. Given the different backbones pretrained on IMAGENET, we then train a linear classifier on this dataset (assumed to be under white illumination). For testing we create different versions of the dataset, each illuminated by illuminants



(a)



(b)

Figure 5.4: Color sensitivity analysis. (a) Robustness to illuminant change: we report the accuracies by differently-trained backbones as a function of illuminant. (b) The color sensitivity indexes computed for the different configurations used for training the backbone.

of differing color temperature. This allows us to evaluate the robustness of the learned representations with respect to these illuminant changes.

Results are given in Figure 5.4(a) (more results are provided in Figure 5.5 from section 5.4.6). *Planckian Jitter* obtains a remarkably stable performance between 4000K and 14000K, while *Default Color Jitter* is more sensitive to the illumination color and the classification accuracy decreases when the scene illuminant moves away from white. We also see that the combination of default and Planckian Jitter obtains the best results for all illuminants and manages to maintain a high-level of invariance with respect to the illuminant color. Among the non-Planckian curves, default color jitter (CJ) is the most invariant, followed by CJ+CJ- (although showing better performance at D65), and finally CJ-.

In order to understand the impact of the color information on each neuron in trained models, we conducted an analysis using the color selectivity index described by [120]. This index measures neuron activation when color is present or absent in input images. We computed the index for the last layer of different backbones, and high values indicate color-sensitive neurons. See section 5.3.2 for more details on color selectivity. The results are shown in Figure 5.4(b) and indicate the number of color-sensitive neurons for each of the considered models. It is clear that the default color jitter has far fewer neurons dedicated to color description. This result confirms the hypothesis that models trained in this way are color invariant, a property that negatively affects the model in scenarios where color information has an important role as seen in our experiments. We have also analyzed the results for the default color jitter without Grayscale augmentation (CJ-). These results show that removing the Grayscale augmentation improves color sensitivity significantly. We therefore also consider this augmentation in future experiments.

5.4.3 Ablation study

Table 5.1: Accuracy results with ablation on color augmentations. Self-supervised training is performed on CIFAR-100 and the learned features are evaluated at (32 × 32) on CIFAR-100 and FLOWERS-102. Augmentation techniques include variations in hue and saturation (H&S), brightness and contrast (B&C), Planckian-based chromaticity (P), and random Grayscale conversions (G). Accuracy refers to the linear classifiers trained with features extracted from the different backbones.

AUGMENTATION	**H&S**	**B&C**	**G**	**P**	**CIFAR-100**	**FLOWERS-102**
None					41.93%	36.47%
Default Color Jitter	✓	✓	✓		59.93%	30.00%
	✓	✓			41.96%	36.96%
	✓				32.46%	39.11%
				✓	36.10%	39.51%
		✓			31.78%	41.96%
Planckian Jitter		✓		✓	47.31%	42.75%

Six different models were trained and evaluated with a linear classification for image classification. For resolution 32 × 32 the model is evaluated on CIFAR-100 and FLOWERS-102. The results in terms of accuracy are reported in Table 5.1 and Table 5.2. We identify two different trends when interpreting these results. On CIFAR-100, removing color augmentations makes the model less powerful, due to the loss of color invariance in the features extracted by the encoder. This behaviour is consistent with what was reported by [22]. We see in Table 5.1 that if color augmentations (i.e. brightness/contrast and Random Grayscale) are removed completely (the *None* configuration), the accuracy drops by 18%. On FLOWERS-102 the behavior is the opposite however: removing color augmentations helps the model to better classify images, obtaining an improvement of 12.75% of accuracy with respect to the default color jitter. This behavior confirms that color invariance negatively impacts downstream tasks where color information plays an important

Table 5.2: Accuracy results for self-supervised training on CIFAR-100 and evaluated at 32×32 on CIFAR-100 and FLOWERS-102. The reported accuracy refers to the results of the linear classifiers trained with features extracted from the different trained backbones.

AUGMENTATION	**CIFAR-100**	**FLOWERS-102**
Default Color Jitter (CJ)	59.93%	30.00%
Default Color Jitter w/o Grayscale (CJ-)	41.96%	36.96%
Planckian Jitter (PJ)	47.31%	42.75%
LSC: [CJ, CJ-]	62.27%	47.65%
LSC: [CJ, PJ]	63.54%	51.66%

role.

Taking a closer look at the various augmentation on FLOWERS-102, we see that introducing more realistic color augmentations positively impacts contrastive training and produces models that achieve even better results with respect to the configuration without any kind of image color manipulation. Removing all color augmentations (None) improves results already by over 6%. Then, by simply reducing the jittering operation to influence brightness and contrast, leaving hue and saturation unchanged, yields another boost in accuracy of 5.49% (to 41.96). When we start modifying chromaticity using a more realistic transformation (i.e *Planckian Jitter*), the final result is a boost of 6.28% in accuracy with respect to the *None* configuration. Also, on CIFAR-100 we see an improvement of 5.38% from Planckian Jitter with respect no color augmentation. Despite this improvement, in this scenario the contrastive training with the realistic augmentation does not yield better results with respect to the *Default* configuration because color only plays a minor role on this dataset.

Given the results obtained using the data augmentations reported in Table 5.1, and given the considerations made in Section 5.3.3, we evaluate the complementarity of the learned representation by combining latent spaces from different backbones. Results for two different latent space combinations are given in Table 5.7. On both datasets the *Latent space combination* of Default and Planckian Jitter achieves the best results. On the original CIFAR-100 task, this combination achieves a total accuracy of 63.54%, a 3.61% improvement over *Default* and 16.23% more compared to *Planckian Jitter* alone. Comparing to the LSC using the Default ColorJitter w/o Grayscale, the version with Planckian Jitter achieves a small improvement of 1.27% in classification accuracy.

On the downstream FLOWERS-102 task, *Latent space combination* reaches an accuracy value of 51.66%: an improvement of 21.66% and 8.91% in accuracy respectively compared to the two original configurations. Compared to the LSC using Default ColorJitter w/o Grayscale, the combination with Planckian Jitter achieves a higher result, and a bigger gap in terms of accuracy with respect to the CIFAR-100 scenario. Here the use of Planckian Jitter brings an improvement of 4.01%, confirming the impact of using realistic augmentation on classification tasks for which color is important.

5.4.4 Evaluation on downstream tasks

Given the ablation study results, we performed the analysis of the proposed configurations on other downstream tasks using the backbone trained on higher resolution images (224 × 224 pixels). We report in Table 5.3 the results for: *Default Color Jitter*, *Planckian Jitter*, and latent space combinations. The datsets used in the finetuning step are:

- FLOWERS-102 [105]: Dataset consisting of 102 flower categories commonly occurring in the United Kingdom. Each class consists of between 40 and 258 images, for a total 8,189 images.
- VEGFRU [128]: Dataset consisting of more than 160,000 images of vegetables and fruits divided in 292 classes.
- CUB-200 [147]: Dataset made of 6,033 images of 200 bird species.
- T1K+ [26]: Dataset of textures divided into 1129 classes and organized in 5 groups of 266 super classes. We adopted the 266 class labeling to finetune and test our models.
- USED [1]: Dataset consisting of 14 categories of social events from around the world. Images depict the interaction between multiple objects and the background scene. We considered 1000 images per class for training, and 500 images per class for testing.

Looking at the results, we see that the *Planckian Jitter* augmentation outperforms default color jitter on three datasets (CUB-200, T1K+, and USED). Comparing the results on FLOWERS-102 with those reported above at (32 × 32) pixels, we see that default color jitter actually obtains good results. We hypothesize that for high-resolution images the shape/texture information is very discriminative, and the additional color information yields little gain (for further analysis, see also 5.4.8). Table 5.3 also contains results for latent space combination, which confirm that the

Table 5.3: Evaluation on downstream tasks. Self-supervised training was performed on IMAGENET at (224 × 224) and testing performed on the downstream datasets resized to (224 × 224).

AUGMENTATION	CUB-200	VEGFRU	T1K+	USED	FLOWERS-102
Default Color Jitter (CJ)	54.52%	67.63%	71.44%	59.90%	93.16%
Planckian Jitter (PJ)	56.28%	65.84%	77.42%	60.03%	90.29%
LSC [CJ,PJ]	**60.70%**	**74.73%**	**80.49%**	**64.07%**	**93.99%**
LSC [CJ,CJ]	56.16%	70.59%	73.47%	61.07%	93.13%
LSC [CJ,CJ-]	53.14%	70.54%	78.32%	63.87%	93.47%

Table 5.4: Additional analysis on downstream tasks. Self-supervised training is performed on TINY-IMAGENET at (64 × 64).

DATA AUGMENTATION	TINY-IMAGENET	FLOWERS-102	CUB200	VEGFRU	T1K+
None	27.06%	37.65%	18.76%	24.07%	35.82%
Default Color Jitter (CJ)	33.12%	46.27%	19.36%	23.92%	26.01%
Default Color Jitter w/o Grayscale (CJ-)	31.62%	40.39%	21.90%	27.39%	32.50%
Planckian Jitter (PJ)	30.95%	52.35%	25.12%	28.94%	32.51%
LSC: [CJ,CJ-]	39.02%	58.33%	26.82%	36.43%	37.20%
LSC: [CJ,PJ]	39.23%	61.57%	30.45%	39.65%	38.20%

two learned representations are complementary. Their combination yields gains of up to 9% on T1K+. As a sanity check we also include the latent space combination of two networks separately trained with Color Jitter. This provides a small gain on some datasets, but yields significantly inferior results than LSC.

5.4.5 Additional downstream results on Tiny-Imagenet

We also performed experiments for several other configurations of the downstream tasks with the representation trained on Tiny-ImageNet. In Table 5.4 we report results for the main task and downstream task (as in section 5.4.4 of the main paper ImageNet, but here all images are at 64 × 64 pixel resolution.

These additional comparisons confirm the conclusions described in section 5.4.4. For all of the considered downstream tasks the application of the proposed data augmentation procedure improves the results even in comparison with other combinations of the originally used data augmentations. Moreover, the comparison with the latent space combination with the two versions of the default color jitter shows how exploiting features extracted by the model trained using the proposed Planckian Jitter augmentation enriches the expressive power of the final model.

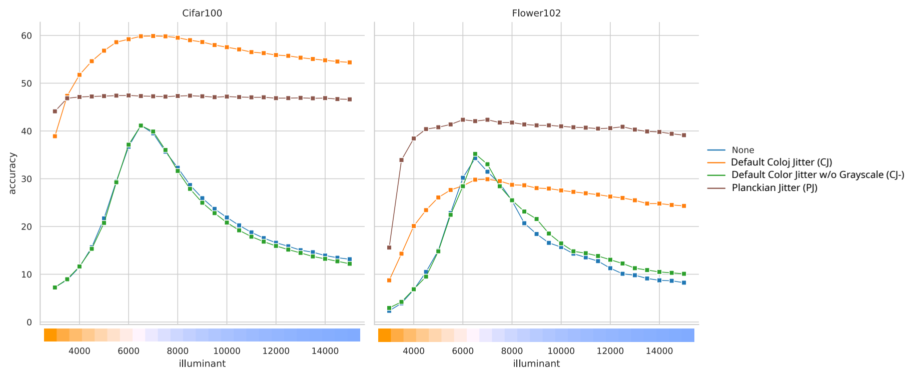


Figure 5.5: Illumimant robustness analysis. To assess feature invariance to realistic color changes in images, for each method we evaluate classification accuracy on 25 different, re-illumintated versions of the datsets. The images of the two datasets (CIFAR-100 on the left and FLOWERS-102 on the right) have been modified with the illumimants from temperature 3000 K to 15000 K using the Planckian Jitter transform.

5.4.6 Color sensitivity

To analyze feature robustness to different illuminants, we tested the models with different, re-illuminated versions of the CIFAR-100 and FLOWERS-102 datasets. We applied *Planckian Jitter* on the two datasets, generating 25 different versions of each, one for each illuminant sampled. Using these different versions of the datasets we then test the models for each illuminant and collect the classification accuracies. The results on both CIFAR-100 and FLOWERS-102 are given in Figure 5.5.

5.4.7 Execution time comparison

Here we provide an analysis of execution time to assess the usability of the Planckian Jitter compared to standard Color Jitter. We executed the two algorithms: the Color Jitter image transform from Torchvision (Torch version v1.8.1 and Torchvision version v0.9.1) and our *Planckian Jitter* at different image resolutions. For each resolution we ran the code 40 times and averaged the execution time. Results are shown in Figure 5.6. All augmentations were performed in CPU on an Intel i7-8700 processor. As can be seen, the proposed *Planckian Jitter* is faster than standard Color Jitter.

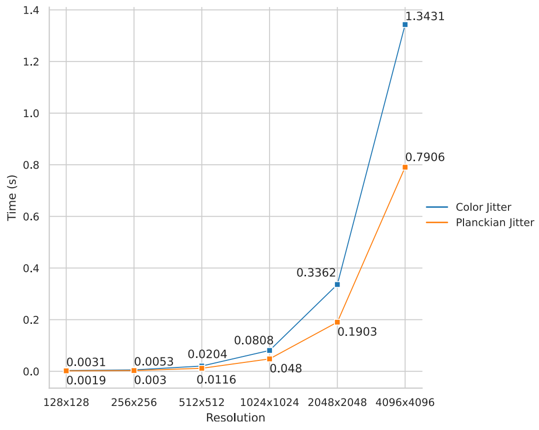


Figure 5.6: Comparison of execution time between the proposed *Plackian Jitter* transform and the Color Jitter implementation in Pytorch Torchvision v0.9.1. For each resolution we executed both the algorithms 40 times.

5.4.8 Dependence on resolution

To better understand the difference of the performances reported in Tables 5.2 and 5.3, we perform an experiment that shows that the relative importance of texture and shape increases with increased resolution.

As an additional experiment, we took the representations and classifiers learned at high-resolution (224x224) and investigated their sensitivity to high-frequency information in images. At inference time, we down-sample (down-sample resolution is given in Table 5.5) and then up-sample all images. In this way, we can compare the dependence on high-resolution information of different methods. Note, that here we do not retrain the classifier but use the one trained at 224x224. The results clearly show that CJ suffers more from down-sampling than PJ. For the Flowers

Table 5.5: Classification accuracy as a function of down-sampling size on Flowers. Results confirm that PJ is less sensitive to down-sampling than CJ.

Method	**32**	**64**	**128**	**224**
CJ	7.74	49.23	91.43	91.90
CJ+PJ	16.30	66.43	93.01	93.45
PJ	23.11	70.13	89.04	89.22

datasets, the results of CJ at resolution 224 are better than PJ. However, when we down-sample to 64, the results change and results for PJ are already significantly better than CJ. This suggest that the texture information (important for CJ) is removed and this hurts performance. For PJ, which is more dependent on color information, down-sampling hurts results less (note PJ is also using texture, shape but to a lesser degree, so results still deteriorate for smaller resolutions).

5.4.9 Color importance in Planckian Jitter based representation

Table 5.6: Methods evaluated with color and grayscale images on the Flowers dataset.

Method	**COLOR**	**ACCURACY**
CJ	COLOR	92.73
CJ	GS	89.51
PJ	COLOR	88.97
PJ	GS	21.38

To verify that CJ uses less color information than PJ, we did a simple experiment where at inference time we changed the input images from sRGB to gray-scale images. These results are provided in Table 5.6. These results clearly show that PJ is much more dependent on color than CJ. PJ has a drop of over 67.6% whereas CJ only drops 3.2 percentage points.

5.4.10 Generality and Limitations of Planckian Jitter

To show that our approach is generally applicable to self-supervised methods exploiting color augmentations, we report in Table 5.7 experiments using SimCLR, Barlow Twins, and the more recent VicReg [9]. Independently of the model, *Latent Space Combination* consistently achieves the best results on both datasets.

A drawback of Planckian Jitter is the quality reduction of shape and texture representations, because the extreme color transformation of the standard Color Jitter force the network to solve the contrastive learning problem mainly using shape/texture information. As we have shown, this problem can be addressed by exploiting their complimentary nature. Secondly, our current latent space combination requires the training of two separate backbones, which will also learn partially-overlapping features. A training scenario with both augmentations simultaneously in a single network while reserving part of the latent space for each augmentation could be pursued to address this limitation. Finally, object-specific augmentations

Table 5.7: Effect of Plackian Jitter on different contrastive learning models. Self-supervised training was performed on CIFAR-100 and the learned features are evaluated at (32 × 32) on CIFAR-100 and FLOWERS-102. We report the best configurations obtained on SimSiam model and retrained SimCLR and Barlow Twins with those selected configurations.

FRAMEWORK	AUGMENTATION	CIFAR-100	FLOWERS-102
SimSiam	Default Color Jitter (CJ)	59.93%	30.00%
	Planckian Jitter (PJ)	47.31%	42.75%
	LSC [CJ,PJ]	63.54%	51.66%
SimCLR	Default Color Jitter (CJ)	56.99%	35.29%
	Planckian Jitter (PJ)	47.75%	45.00%
	LSC [CJ,PJ]	61.07%	55.78%
Barlow Twins	Default Color Jitter (CJ)	56.60%	40.78%
	Planckian Jitter (PJ)	52.71%	54.50%
	LSC [CJ,PJ]	62.85%	62.55%
VicReg	Default Color Jitter (CJ)	65.23%	49.50%
	Planckian Jitter (PJ)	59.19%	50.90%
	LSC [CJ,PJ]	68.95%	60.80%

that also take into account shadows, the type of reflectance, secondary light sources, inter-reflections, shadows, etc, could lead to further improvements.

5.5 Conclusion

Existing research on self-supervised learning mainly focuses on tasks where color is not a decisive feature, and consequently exploits data augmentation procedures that negatively affect color-sensitive tasks. We propose an alternative color data augmentation, called Planckian Jitter, that is based on the physical properties of light. Our experiments demonstrate its positive effects on a wide variety of tasks where the intrinsic color of the objects (related to their reflectance) is crucial for discrimination, while the illumination source is not. We also proposed exploiting both color and shape information by concatenating features learned with different modalities of self-supervision, leading to significant overall improvements in learned representations. Planckian Jitter can be easily incorporated into any self-supervised learning pipeline based on data augmentations, as shown by our results demonstrating improved performance for three self-supervised learning models.

6 Conclusions and Future Work

6.1 Conclusions

In this thesis, we study the intersection between self-supervised and continual learning. For continual learning, we aimed to research and develop new exemplar-free methods that exploit the properties of self-supervised representations to mitigate catastrophic forgetting. In self-supervised learning, we focused on the study and analysis of color image augmentations and their implications for color-relevant applications.

- **Chapter 2: Continually Learning Self-Supervised Representations with Projected Functional Regularization.** This chapter introduced a novel exemplar-free continual unsupervised learning method. Traditional continual learning methods often apply regularization to class predictions or logits, which can severely limit plasticity when used in self-supervised representation learning. We presented Projected Functional Regularization, which employs a temporal projection network to overcome this limitation. This technique preserves information from the previously learned feature space while allowing to acquire new features, thereby enhancing plasticity. Our method's effectiveness was demonstrated through comprehensive experiments on CIFAR100 and Tiny ImageNet datasets, where it significantly outperformed conventional feature distillation techniques.

- **Chapter 3: Plasticity-Optimized Complementary Networks for Unsupervised Continual Learning.**

  Existing unsupervised continual learning methods can suffer from suboptimal stability and plasticity on longer sequences, since they have difficulty adapting to the new knowledge required to address the latest task while maintaining the vast knowledge already learned on earlier tasks. We introduced 'Plasticity-Optimized COmplementary Networks' (POCON); POCON employs an expert network that maximizes performance on current data without constraints, thus optimizing plasticity. Knowledge is then transferred to a main network through distillation while maintaining similarity with the previous-task main network to prevent forgetting. Our method outperformed existing techniques for exemplar-free continual self-supervised learning, such as PFR and CaSSLe, as demonstrated by experiments on CIFAR100, TinyImagenet, and ImageNet100 datasets. POCON showed particular strength in scenarios involving numerous tasks.

- **Chapter 4: Exemplar-free Continual Representation Learning via Learnable Drift Compensation.** In this chapter, we found that feature drift is a primary cause of performance decline in exemplar-free continual learning methods that use prototype accumulation. To address this, we introduced Learnable Drift Compensation (LDC), a straightforward yet powerful prototype correction technique. LDC develops a projector that maps between consecutive tasks' feature spaces, utilizing only the data available at each task along with current and previous models. This projector is applied after each task to adjust the positions of stored old prototypes. Comprehensive experiments on continual learning benchmarks demonstrated that LDC surpasses previous drift correction methods in supervised and semi-supervised contexts.

- **Chapter 5: Planckian Jitter: countering the color-crippling effects of color jitter on self-supervised training.** Current self-supervised learning research focuses on tasks where color is not critical, often using data augmentation techniques that can hinder performance on color-sensitive tasks. To address this, we introduced 'Planckian Jitter,' a novel color data augmentation method based on light's physical properties. Our experiments showed its effectiveness across various tasks where objects' intrinsic color (linked to their reflectance) is crucial for differentiation, independent of the illumination source. Furthermore, we proposed combining color and shape information by merging features learned through different self-supervision modalities, resulting in substantial improvements in overall representation learning.

6.2 Future work

For future work, we are interested in studying the accuracy boost that smaller networks can give to self-supervised continual learning (as Chapter 3 showed). We think these lower capacity networks can offer interesting properties in many-task scenarios or even in extreme settings such as online continual learning.

Second, we studied and developed methods for architectures based on convolutional neural networks. However, vision transformers (ViT) have become a standard for computer vision applications, and several methods for continual learning using ViTs have been proposed. In particular, we would like to study how to extend current ViT-based continual learning methods to the unsupervised exemplar-free regime.

Finally, this thesis focused on computer vision, but vision and language merged into the same topic as time passed. We are interested in exploring continual learning methods that can be applied to vision-language models. In this multi-modal realm, language could contribute to retaining and distilling knowledge between tasks as it has a higher level of representation than images.

Publications

1. **Gomez-Villa, A.**, Twardowski, B., Yu, L., Bagdanov, A. D., 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, CLVision Workshop.

2. **Gomez-Villa, A.**, Twardowski, B., Wang, K., 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.

3. **Gomez-Villa, A.**, Goswami, D., Wang, K., Bagdanov, A. D., Twardowski, B., van de Weijer, J. (2024). Exemplar-free Continual Representation Learning via Learnable Drift Compensation. In Proceedings of the IEEE/CVF European Conference on Computer Vision.

4. Zini, S., **Gomez-Villa, A.**, Buzzelli, M., Twardowski, B., Bagdanov, A. D., Van de Weijer, J. (2023). Planckian jitter: countering the color-crippling effects of color jitter on self-supervised training. In Proceedings of the International Conference on Learning Representations.

5. Porres, D., **Gomez-Villa, A**. (2024). At the edge of a generative cultural precipice. Fourth Workshop on Ethical Considerations in Creative Applications of Computer Vision, CVPR.

Awards

1. Best paper award at the CVPR 2022 Workshop on Continual Learning (CLVision, 3rd Edition) for the article: *Continually learning self-supervised representations with projected functional regularization.*

Scientific communication

During my PhD, I actively participated in science communication events.

1. Alexandra Gomez-Villa & Laura Martín. (2022). *La nit de la recerca*, Cosmocaixa.

2. Alexandra Gomez-Villa. (2022). *¿Sueñan las máquinas con ilusiones visuales?*, Prisma conference.

3. Alexandra Gomez-Villa. (2023). *Inteligencia artificial generativa*, 100científiques.

4. Alexandra Gomez-Villa. (2023). *Somien les màquines amb il·lusions visuals?*, ExperimentAI.

5. Arturo Fuentes & Alexandra Gomez-Villa. (2023). *Intel·ligència artificial generativa i art: les màquines són creatives?*, Intel·ligència artificial: revelant impactes socials.

6. Alexandra Gomez-Villa. (2024). *La importancia de recordar: minimizando el olvido en la inteligencia artificial para combatir el cambio climático*, concurs Tesi en 4 minuts UAB.

7. Alexandra Gomez-Villa. (2024). *Inteligencia artificial generativa*, 100científiques.

8. Alexandra Gomez-Villa. (2024). Pints of Science festival, Barcelona edition.

9. Alexandra Gomez-Villa & Diego Porres. (2024). *Experimenta amb les possibilitats de la intel·ligència artificial i l'art*, la festa de la ciència.

Bibliography

[1] Kashif Ahmad, Nicola Conci, Giulia Boato, and Francesco GB De Natale. Used: a large-scale social event detection dataset. In *Proceedings of the 7th International Conference on Multimedia Systems*, pages 1–6, 2016.

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

[3] Rahaf Aljundi, Klaas Kelchtermans, and Tinne Tuytelaars. Task-free continual learning. In *Conference on Computer Vision and Pattern Recognition*, 2019.

[4] David G Andrews. *An introduction to atmospheric physics*. Cambridge University Press, 2010.

[5] Elahe Arani, Fahad Sarfraz, and Bahram Zonooz. Learning fast, learning slow: A general continual learning method based on complementary learning system. In *International Conference on Learning Representations*, 2022.

[6] Jordan Ash and Ryan P Adams. On warm-starting neural network training. *Advances in Neural Information Processing Systems*, 33:3884–3894, 2020.

[7] Mahmoud Assran, Mathilde Caron, Ishan Misra, Piotr Bojanowski, Armand Joulin, Nicolas Ballas, and Michael Rabbat. Semi-supervised learning of visual features by non-parametrically predicting view assignments with support samples. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8443–8452, 2021.

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

[9] Adrien Bardes, Jean Ponce, and Yann LeCun. Vicreg: Variance-invariance-covariance regularization for self-supervised learning. *arXiv preprint arXiv:2105.04906*, 2021.

[10] Yuemin Bian and Xiang-Qun Xie. Generative chemistry: drug discovery with deep learning generative models. *Journal of Molecular Modeling*, 27:1–18, 2021.

[11] Matteo Boschini, Pietro Buzzega, Lorenzo Bonicelli, Angelo Porrello, and Simone Calderara. Continual semi-supervised learning through contrastive interpolation consistency. *Pattern Recognition Letters*, 162:9–14, 2022.

[12] Marco Buzzelli, Simone Bianco, and Raimondo Schettini. Arc: Angle-retaining chromaticity diagram for color constancy error analysis. *JOSA A*, 37(11):1721–1730, 2020.

[13] Vivien Cabannes, Bobak Kiani, Randall Balestriero, Yann LeCun, and Alberto Bietti. The ssl interplay: Augmentations, inductive bias, and generalization. In *International Conference on Machine Learning*, pages 3252–3298. PMLR, 2023.

[14] Lucas Caccia and Joelle Pineau. Special: Self-supervised pretraining for continual learning. In *International Workshop on Continual Semi-Supervised Learning*, pages 91–103. Springer, 2022.

[15] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[16] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. *CoRR*, abs/2104.14294, 2021.

[17] Francisco M Castro, Manuel J Marín-Jiménez, Nicolás Guil, Cordelia Schmid, and Karteek Alahari. End-to-end incremental learning. In *European Conference on Computer Vision*, 2018.

[18] Hyuntak Cha, Jaeho Lee, and Jinwoo Shin. Co2l: Contrastive continual learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9516–9525, 2021.

[19] Arslan Chaudhry, Puneet K Dokania, Thalaiyasingam Ajanthan, and Philip HS Torr. Riemannian walk for incremental learning: Understanding forgetting and intransigence. In *Proc. of ECCV, (Springer)*, pages 532–547, 2018.

[20] Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with a-GEM. In *Proc. of ICLR*, 2019.

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

[22] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.

[23] Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. Improved baselines with momentum contrastive learning. *arXiv preprint arXiv:2003.04297*, 2020.

[24] Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15750–15758, 2021.

[25] Ali Cheraghian, Shafin Rahman, Pengfei Fang, Soumava Kumar Roy, Lars Petersson, and Mehrtash Harandi. Semantic-aware knowledge distillation for few-shot class-incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2534–2543, 2021.

[26] Claudio Cusano, Paolo Napoletano, and Raimondo Schettini. T1k+: A database for benchmarking color texture classification and retrieval methods. *Sensors*, 21(3):1010, 2021.

[27] Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Ales Leonardis, Gregory Slabaugh, and Tinne Tuytelaars. A continual learning survey: Defying forgetting in classification tasks. *IEEE transactions on pattern analysis and machine intelligence*, 2021.

[28] Mostafa Dehghani, Josip Djolonga, Basil Mustafa, Piotr Padlewski, Jonathan Heek, Justin Gilmer, Andreas Peter Steiner, Mathilde Caron, Robert Geirhos, Ibrahim Alabdulmohsin, et al. Scaling vision transformers to 22 billion parameters. In *International Conference on Machine Learning*, pages 7480–7512. PMLR, 2023.

[29] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Conference on Computer Vision and Pattern Recognition*, 2009.

[30] Jacob Devlin. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[31] Prithviraj Dhar, Rajat Vikram Singh, Kuan-Chuan Peng, Ziyan Wu, and Rama Chellappa. Learning without memorizing. In *Conference on Computer Vision and Pattern Recognition*, 2019.

[32] Carl Doersch, Abhinav Gupta, and Alexei A Efros. Unsupervised visual representation learning by context prediction. In *Proceedings of the IEEE international conference on computer vision*, pages 1422–1430, 2015.

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

[34] Alexey Dosovitskiy, Jost Tobias Springenberg, Martin Riedmiller, and Thomas Brox. Discriminative unsupervised feature learning with convolutional neural networks. *Advances in neural information processing systems*, 27:766–774, 2014.

[35] Arthur Douillard, Matthieu Cord, Charles Ollion, Thomas Robert, and Eduardo Valle. Podnet: Pooled outputs distillation for small-tasks incremental learning. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XX 16*, pages 86–102. Springer, 2020.

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

[37] Debidatta Dwibedi, Yusuf Aytar, Jonathan Tompson, Pierre Sermanet, and Andrew Zisserman. With a little help from my friends: Nearest-neighbor contrastive learning of visual representations. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9588–9597, 2021.

[38] Aleksandr Ermolov, Aliaksandr Siarohin, Enver Sangineto, and Nicu Sebe. Whitening for self-supervised representation learning. In *International conference on machine learning*, pages 3015–3024. PMLR, 2021.

[39] Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, et al. Scaling rectified flow transformers for high-resolution image synthesis. In *Forty-first International Conference on Machine Learning*, 2024.

[40] European Parliament and Council of the European Union. Regulation (EU) 2016/679 of the European Parliament and of the Council.

[41] Zach Evans, CJ Carr, Josiah Taylor, Scott H Hawley, and Jordi Pons. Fast timing-conditioned latent audio diffusion. *arXiv preprint arXiv:2402.04825*, 2024.

[42] Chrisantha Fernando, Dylan Banarse, Charles Blundell, Yori Zwols, David Ha, Andrei A Rusu, Alexander Pritzel, and Daan Wierstra. Pathnet: Evolution channels gradient descent in super neural networks. *arXiv*, 2017.

[43] Enrico Fini, Victor G Turrisi Da Costa, Xavier Alameda-Pineda, Elisa Ricci, Karteek Alahari, and Julien Mairal. Self-supervised models are continual learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9621–9630, 2022.

[44] Graham D Finlayson and Gerald Schaefer. Solving for colour constancy using a constrained dichromatic reflection model. *International Journal of Computer Vision*, 42(3):127–144, 2001.

[45] Valentin Gabeff, Marc Rußwurm, Devis Tuia, and Alexander Mathis. Wildclip: Scene and animal attribute retrieval from camera trap data with domain-adapted vision-language models. *International Journal of Computer Vision*, pages 1–17, 2024.

[46] Adrian Galdran, Aitor Alvarez-Gila, Maria Ines Meyer, Cristina L Saratxaga, Teresa Araújo, Estibaliz Garrote, Guilherme Aresta, Pedro Costa, Ana Maria Mendonça, and Aurélio Campilho. Data-driven color augmentation techniques for deep skin image analysis. *arXiv preprint arXiv:1703.03702*, 2017.

[47] Jhair Gallardo, Tyler L Hayes, and Christopher Kanan. Self-supervised training enhances online continual learning. *arXiv preprint arXiv:2103.14010*, 2021.

[48] Jonas Geiping, Quentin Garrido, Pierre Fernandez, Amir Bar, Hamed Pirsiavash, Yann LeCun, and Micah Goldblum. A cookbook of self-supervised learning. *arXiv preprint arXiv:2304.12210*, 2023.

[49] J-M Geusebroek, Rein Van den Boomgaard, Arnold W. M. Smeulders, and Hugo Geerts. Color invariance. *IEEE Transactions on Pattern analysis and machine intelligence*, 23(12):1338–1350, 2001.

[50] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations. *arXiv preprint arXiv:1803.07728*, 2018.

[51] Alex Gomez-Villa, Dipam Goswami, Kai Wang, Andrew D Bagdanov, Bartlomiej Twardowski, and Joost van de Weijer. Exemplar-free continual representation learning via learnable drift compensation. *Proceedings of the IEEE/CVF European Conference on Computer Vision*, 2024.

[52] Alex Gomez-Villa, Bartlomiej Twardowski, Kai Wang, and Joost van de Weijer. Plasticity-optimized complementary networks for unsupervised continual learning. In *IEEE/CVF Winter Conference on Applications of Computer Vision*, 2024.

[53] Alex Gomez-Villa, Bartlomiej Twardowski, Lu Yu, Andrew D Bagdanov, and Joost Van de Weijer. Continually learning self-supervised representations with projected functional regularization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3867–3877, 2022.

[54] Dipam Goswami, Yuyang Liu, Bartłomiej Twardowski, and Joost van de Weijer. Fecam: Exploiting the heterogeneity of class distributions in exemplar-free continual learning. *Advances in Neural Information Processing Systems*, 36, 2024.

[55] Dipam Goswami, Albin Soutif-Cormerais, Yuyang Liu, Sandesh Kamath, Bartłomiej Twardowski, and Joost van de Weijer. Resurrecting old classes with new data for exemplar-free continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[56] Dipam Goswami, Bartłomiej Twardowski, and Joost Van De Weijer. Calibrating higher-order statistics for few-shot class-incremental learning with pre-trained vision transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024.

[57] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, Bilal Piot, koray kavukcuoglu, Remi

Munos, and Michal Valko. Bootstrap your own latent - a new approach to self-supervised learning. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 21271–21284. Curran Associates, Inc., 2020.

[58] Linting Guan and Yan Wu. Reduce the difficulty of incremental learning with self-supervised learning. *IEEE Access*, 2021.

[59] Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 1735–1742. IEEE, 2006.

[60] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16000–16009, 2022.

[61] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9729–9738, 2020.

[62] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Conference on Computer Vision and Pattern Recognition*, 2016.

[63] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. In *NIPS Deep Learning Workshop*, 2014.

[64] Harold Hotelling. Relations between two sets of variates. In *Breakthroughs in statistics: methodology and distribution*, pages 162–190. Springer, 1992.

[65] Saihui Hou, Xinyu Pan, Chen Change Loy, Zilei Wang, and Dahua Lin. Lifelong learning via progressive distillation and retrospection. In *European Conference on Computer Vision*, 2018.

[66] Saihui Hou, Xinyu Pan, Chen Change Loy, Zilei Wang, and Dahua Lin. Learning a unified classifier incrementally via rebalancing. In *Proc. of CVPR, (IEEE)*, pages 831–839, 2019.

[67] Xinting Hu, Kaihua Tang, Chunyan Miao, Xian-Sheng Hua, and Hanwang Zhang. Distilling causal effect of data in class-incremental learning. In

Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 3957–3966, 2021.

[68] Po-Yao Huang, Hu Xu, Juncheng Li, Alexei Baevski, Michael Auli, Wojciech Galuba, Florian Metze, and Christoph Feichtenhofer. Masked autoencoders that listen. *Advances in Neural Information Processing Systems*, 35:28708–28720, 2022.

[69] Ahmet Iscen, Jeffrey Zhang, Svetlana Lazebnik, and Cordelia Schmid. Memory-efficient incremental learning through feature adaptation. In *European Conference on Computer Vision*, 2020.

[70] Paul Janson, Wenxuan Zhang, Rahaf Aljundi, and Mohamed Elhoseiny. A simple baseline that questions the use of pretrained-models in continual learning. In *NeurIPS 2022 Workshop on Distribution Shifts: Connecting Methods and Applications*, 2022.

[71] Zhiqi Kang, Enrico Fini, Moin Nabi, Elisa Ricci, and Karteek Alahari. A soft nearest-neighbor framework for continual semi-supervised learning. *arXiv preprint arXiv:2212.05102*, 2022.

[72] Ronald Kemker and Christopher Kanan. Fearnet: Brain-inspired model for incremental learning. In *International Conference on Learning Representations*, 2018.

[73] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *National Academy of Sciences*, 2017.

[74] Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited. In *International Conference on Machine Learning*, pages 3519–3529. PMLR, 2019.

[75] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *IEEE International Conference on Computer Vision Workshops*, pages 554–561, 2013.

[76] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[77] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

[78] Ananya Kumar, Aditi Raghunathan, Robbie Jones, Tengyu Ma, and Percy Liang. Fine-tuning can distort pretrained features and underperform out-of-distribution. *arXiv preprint arXiv:2202.10054*, 2022.

[79] Dharshan Kumaran, Demis Hassabis, and James L McClelland. What learning systems do intelligent agents need? complementary learning systems theory updated. *Trends in cognitive sciences*, 20(7):512–534, 2016.

[80] Ya Le and Xuan Yang. Tiny imagenet visual recognition challenge. *CS 231N*, 2015.

[81] Shuang Li, Yilun Du, Gido van de Ven, and Igor Mordatch. Energy-based models for continual learning. In *Conference on Lifelong Learning Agents*, pages 1–22. PMLR, 2022.

[82] Xilai Li, Yingbo Zhou, Tianfu Wu, Richard Socher, and Caiming Xiong. Learn to grow: A continual structure learning framework for overcoming catastrophic forgetting. In *International Conference on Machine Learning*, 2019.

[83] Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

[84] X. Liu, M. Masana, L. Herranz, J. Van de Weijer, A. M. López, and A. D. Bagdanov. Rotate your networks: Better weight consolidation and less catastrophic forgetting. In *Proc. of ICPR*, pages 2262–2268, 2018.

[85] Xialei Liu, Chenshen Wu, Mikel Menta, Luis Herranz, Bogdan Raducanu, Andrew D Bagdanov, Shangling Jui, and Joost van de Weijer. Generative feature replay for class-incremental learning. In *Conference on Computer Vision and Pattern Recognition Workshops*, 2020.

[86] Yixin Liu, Kai Zhang, Yuan Li, Zhiling Yan, Chujie Gao, Ruoxi Chen, Zhengqing Yuan, Yue Huang, Hanchi Sun, Jianfeng Gao, et al. Sora: A review on background, technology, limitations, and opportunities of large vision models. *arXiv preprint arXiv:2402.17177*, 2024.

[87] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[88] Divyam Madaan, Jaehong Yoon, Yuanchun Li, Yunxin Liu, and Sung Ju Hwang. Rethinking the representational continuity: Towards unsupervised continual learning. *arXiv preprint arXiv:2110.06976*, 2021.

[89] Divyam Madaan, Jaehong Yoon, Yuanchun Li, Yunxin Liu, and Sung Ju Hwang. Representational continuity for unsupervised continual learning. In *ICLR*, 2022.

[90] Simone Magistri, Tomaso Trinci, Albin Soutif-Cormerais, Joost van de Weijer, and Andrew D Bagdanov. Elastic feature consolidation for cold start exemplar-free incremental learning. *arXiv preprint arXiv:2402.03917*, 2024.

[91] S. Maji, J. Kannala, E. Rahtu, M. Blaschko, and A. Vedaldi. Fine-grained visual classification of aircraft. *arXiv*, 2013.

[92] Tamasha Malepathirana, Damith Senanayake, and Saman Halgamuge. Napa-vq: Neighborhood-aware prototype augmentation with vector quantization for continual learning. In *International Conference on Computer Vision (ICCV)*, 2023.

[93] Arun Mallya, Dillon Davis, and Svetlana Lazebnik. Piggyback: Adapting a single network to multiple tasks by learning to mask weights. In *Proc. of ECCV, (Springer)*, pages 67–82, 2018.

[94] Arun Mallya and Svetlana Lazebnik. Packnet: Adding multiple tasks to a single network by iterative pruning. In *Conference on Computer Vision and Pattern Recognition*, 2018.

[95] Bangalore S Manjunath and Wei-Ying Ma. Texture features for browsing and retrieval of image data. *IEEE Transactions on pattern analysis and machine intelligence*, 18(8):837–842, 1996.

[96] Marc Masana, Xialei Liu, Bartłomiej Twardowski, Mikel Menta, Andrew D Bagdanov, and Joost Van De Weijer. Class-incremental learning: survey and performance evaluation on image classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(5):5513–5533, 2022.

[97] Marc Masana, Tinne Tuytelaars, and Joost Van de Weijer. Ternary feature masks: zero-forgetting for task-incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3570–3579, 2021.

[98] James L McClelland, Bruce L McNaughton, and Randall C O'Reilly. Why there are complementary learning systems in the hippocampus and neocortex: insights from the successes and failures of connectionist models of learning and memory. *Psychological review*, 102(3):419, 1995.

[99] Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pages 109–165. Elsevier, 1989.

[100] Mark D McDonnell, Dong Gong, Amin Parvaneh, Ehsan Abbasnejad, and Anton van den Hengel. Ranpac: Random projections and pre-trained models for continual learning. *Advances in Neural Information Processing Systems*, 36, 2024.

[101] Lia Medeiros, Dimitrios Psaltis, Tod R Lauer, and Feryal Özel. The image of the m87 black hole reconstructed with primo. *The Astrophysical Journal Letters*, 947(1):L7, 2023.

[102] Amil Merchant, Simon Batzner, Samuel S Schoenholz, Muratahan Aykol, Gowoon Cheon, and Ekin Dogus Cubuk. Scaling deep learning for materials discovery. *Nature*, 624(7990):80–85, 2023.

[103] Martial Mermillod, Aurélia Bugaiska, and Patrick Bonin. The stability-plasticity dilemma: Investigating the continuum from catastrophic forgetting to age-limited learning effects. *Frontiers in psychology*, 2013.

[104] K L Navaneet, Soroush Abbasi Koohpayegani, Ajinkya Tejankar, and Hamed Pirsiavash. Simreg: Regression as a simple yet effective tool for self-supervised knowledge distillation. In *British Machine Vision Conference (BMVC)*, 2021.

[105] Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *2008 Sixth Indian Conference on Computer Vision, Graphics & Image Processing*, pages 722–729. IEEE, 2008.

[106] Mehdi Noroozi and Paolo Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *European conference on computer vision*, pages 69–84. Springer, 2016.

[107] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

[108] Randall C O'Reilly and Kenneth A Norman. Hippocampal and neocortical contributions to memory: Advances in the complementary learning systems framework. *Trends in cognitive sciences*, 6(12):505–510, 2002.

[109] Oleksiy Ostapenko, Timothee Lesort, Pau Rodríguez, Md Rifat Arefin, Arthur Douillard, Irina Rish, and Laurent Charlin. Foundational models for continual learning: An empirical study of latent replay. *arXiv preprint arXiv:2205.00329*, 2022.

[110] Pingbo Pan, Siddharth Swaroop, Alexander Immer, Runa Eschenhagen, Richard E Turner, and Mohammad Emtiyaz Khan. Continual deep learning by functional regularisation of memorable past. In *Proc. Adv. Neural Inf. Process. Syst.*, 2020.

[111] Aristeidis Panos, Yuriko Kobe, Daniel Olmeda Reino, Rahaf Aljundi, and Richard E. Turner. First session adaptation: A strong replay-free baseline for class-incremental learning. In *International Conference on Computer Vision (ICCV)*, 2023.

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

[113] German I Parisi, Jun Tani, Cornelius Weber, and Stefan Wermter. Lifelong learning of spatiotemporal representations with dual-memory recurrent self-organization. *Frontiers in neurorobotics*, page 78, 2018.

[114] Francesco Pelosin, Saurav Jha, Andrea Torsello, Bogdan Raducanu, and Joost van de Weijer. Towards exemplar-free continual learning in vision transformers: an account of attention, functional and weight regularization. *arXiv preprint arXiv:2203.13167*, 2022.

[115] Grégoire Petit, Adrian Popescu, Hugo Schindler, David Picard, and Bertrand Delezoide. Fetril: Feature translation for exemplar-free class-incremental learning. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 3911–3920, 2023.

[116] Benedikt Pfülb and Alexander Gepperth. A comprehensive, application-oriented study of catastrophic forgetting in dnns. In *International Conference on Learning Representations*, 2019.

[117] Quang Pham, Chenghao Liu, and Steven Hoi. Dualnet: Continual learning, fast and slow. *Advances in Neural Information Processing Systems*, 34:16131–16144, 2021.

[118] Helen A. S. Popkin. Ai 50 2022: North america's top ai companies shaping the future. *Forbes*, 2022. Accessed: 05.08.2024.

[119] Ameya Prabhu, Philip HS Torr, and Puneet K Dokania. Gdumb: A simple approach that questions our progress in continual learning. In *Proc. of ECCV, (Springer)*, pages 524–540. Springer, 2020.

[120] Ivet Rafegas and Maria Vanrell. Color encoding in biologically-inspired convolutional neural networks. *Vision research*, 151:7–17, 2018.

[121] Dushyant Rao, Francesco Visin, Andrei Rusu, Razvan Pascanu, Yee Whye Teh, and Raia Hadsell. Continual unsupervised representation learning. In *Advances in Neural Information Processing Systems*, 2019.

[122] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *Proc. of CVPR, (IEEE)*, pages 2001–2010, 2017.

[123] Tal Ridnik, Emanuel Ben-Baruch, Asaf Noy, and Lihi Zelnik-Manor. Imagenet-21k pretraining for the masses. In *Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2021.

[124] David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy Lillicrap, and Gregory Wayne. Experience replay for continual learning. In *Advances in Neural Information Processing Systems*, pages 350–360, 2019.

[125] Amir Rosenfeld and John K Tsotsos. Incremental learning through deep adaptation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2018.

[126] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

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

[128] Yushan Feng Saihui Hou and Zilei Wang. Vegfru: A domain-specific dataset for fine-grained visual categorization. In *IEEE International Conference on Computer Vision*, 2017.

[129] Mert Bulent Sariyildiz, Yannis Kalantidis, Karteek Alahari, and Diane Larlus. No reason for no supervision: Improved generalization in supervised models. In *International Conference on Learning Representations*, 2023.

[130] Jonathan Schwarz, Jelena Luketina, Wojciech M Czarnecki, Agnieszka Grabska-Barwinska, Yee Whye Teh, Razvan Pascanu, and Raia Hadsell. Progress & compress: A scalable framework for continual learning. In *International Conference on Machine Learning*, 2018.

[131] Joan Serra, Didac Suris, Marius Miron, and Alexandros Karatzoglou. Overcoming catastrophic forgetting with hard attention to the task. In *International Conference on Machine Learning*, 2018.

[132] Wuxuan Shi and Mang Ye. Prototype reminiscence and augmented asymmetric knowledge aggregation for non-exemplar class-incremental learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1772–1781, 2023.

[133] Yujun Shi, Li Yuan, Yunpeng Chen, and Jiashi Feng. Continual learning via bit-level information preserving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16674–16683, 2021.

[134] Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual learning with deep generative replay. In *Advances in Neural Information Processing Systems*, 2017.

[135] Amelia Sorrenti, Giovanni Bellitto, Federica Proietto Salanitri, Matteo Pennisi, Concetto Spampinato, and Simone Palazzo. Selective freezing for efficient continual learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3550–3559, 2023.

[136] Stanford. Tiny imagenet challenge, cs231n course., CS231N.

[137] Shixiang Tang, Dapeng Chen, Jinguo Zhu, Shijie Yu, and Wanli Ouyang. Layerwise optimization by gradient decomposition for continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9634–9643, 2021.

[138] Yonglong Tian, Chen Sun, Ben Poole, Dilip Krishnan, Cordelia Schmid, and Phillip Isola. What makes for good views for contrastive learning? In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 6827–6839. Curran Associates, Inc., 2020.

[139] Michalis K Titsias, Jonathan Schwarz, Alexander G de G Matthews, Razvan Pascanu, and Yee Whye Teh. Functional regularisation for continual learning with gaussian processes. In *Proc. Int. Conf. Learn. Repres.*, 2020.

[140] Marco Toldo and Mete Ozay. Bring evanescent representations to life in lifelong class incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16732–16741, 2022.

[141] Shoji Tominaga, Satoru Ebisui, and Brian A Wandell. Color temperature estimation of scene illumination. In *Color and Imaging Conference*, volume 1999, pages 42–47. Society for Imaging Science and Technology, 1999.

[142] Zhan Tong, Yibing Song, Jue Wang, and Limin Wang. Videomae: Masked autoencoders are data-efficient learners for self-supervised video pre-training. *Advances in neural information processing systems*, 35:10078–10093, 2022.

[143] Gido M van de Ven and Andreas S Tolias. Three scenarios for continual learning. In *NeurIPS Continual Learning Workshop*, 2018.

[144] J von Kries. Theoretische studien über die umstimmung des sehorgans. *Festschrift der Albrecht-Ludwigs-Universität*, pages 145–158, 1902.

[145] Liyuan Wang, Kuo Yang, Chongxuan Li, Lanqing Hong, Zhenguo Li, and Jun Zhu. Ordisco: Effective and efficient usage of incremental unlabeled data for semi-supervised continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5383–5392, 2021.

[146] Kilian Q Weinberger, John Blitzer, and Lawrence K Saul. Distance metric learning for large margin nearest neighbor classification. In *Advances in neural information processing systems*, pages 1473–1480, 2006.

[147] P. Welinder, S. Branson, T. Mita, C. Wah, F. Schroff, S. Belongie, and P. Perona. Caltech-UCSD Birds 200. Technical Report CNS-TR-2010-001, California Institute of Technology, 2010.

[148] Chenshen Wu, Luis Herranz, Xialei Liu, Joost van de Weijer, Bogdan Raducanu, et al. Memory replay gans: Learning to generate new categories without forgetting. *Proc. of NIPS, (Curran Associates)*, 31:5962–5972, 2018.

[149] Yue Wu, Yinpeng Chen, Lijuan Wang, Yuancheng Ye, Zicheng Liu, Yandong Guo, and Yun Fu. Large scale incremental learning. In *Proc. of CVPR, (IEEE)*, pages 374–382, 2019.

[150] Gunter Wyszecki and Walter Stanley Stiles. *Color science*, volume 8. Wiley New York, 1982.

[151] Tete Xiao, Xiaolong Wang, Alexei A Efros, and Trevor Darrell. What should not be contrastive in contrastive learning. In *International Conference on Learning Representations*, 2021.

[152] Yang Xiao, Etienne Decencière, Santiago Velasco-Forero, Hélène Burdin, Thomas Bornschlögl, Françoise Bernerd, Emilie Warrick, and Thérèse Baldeweck. A new color augmentation method for deep learning segmentation of histological images. In *2019 IEEE 16th International Symposium on Biomedical Imaging (ISBI 2019)*, pages 886–890. IEEE, 2019.

[153] Zhenda Xie, Zheng Zhang, Yue Cao, Yutong Lin, Jianmin Bao, Zhuliang Yao, Qi Dai, and Han Hu. Simmim: A simple framework for masked image modeling. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9653–9663, 2022.

[154] Shipeng Yan, Jiangwei Xie, and Xuming He. Der: Dynamically expandable representation for class incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3014–3023, 2021.

[155] Jaehong Yoon, Eunho Yang, Jeongtae Lee, and Sung Ju Hwang. Lifelong learning with dynamically expandable networks. *arXiv preprint arXiv:1708.01547*, 2017.

[156] Lu Yu, Bartlomiej Twardowski, Xialei Liu, Luis Herranz, Kai Wang, Yongmei Cheng, Shangling Jui, and Joost van de Weijer. Semantic drift compensation for class-incremental learning. In *Proc. of CVPR, (IEEE)*, pages 6982–6991, 2020.

[157] Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, and Stéphane Deny. Barlow twins: Self-supervised learning via redundancy reduction. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 12310–12320. PMLR, 2021.

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

[159] Chen Zeno, Itay Golan, Elad Hoffer, and Daniel Soudry. Task-Agnostic Continual Learning Using Online Variational Bayes With Fixed-Point Updates. *Neural Computation*, 33(11):3139–3177, 10 2021.

[160] Mengyao Zhai, Lei Chen, and Greg Mori. Hyper-lifelonggan: Scalable lifelong learning for image conditioned generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2246–2255, 2021.

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

[162] Junting Zhang, Jie Zhang, Shalini Ghosh, Dawei Li, Serafettin Tasci, Larry Heck, Heming Zhang, and C-C Jay Kuo. Class-incremental learning via deep model consolidation. In *Winter Conf. on App. of Computer Vision*, 2020.

[163] Richard Zhang, Phillip Isola, and Alexei A Efros. Colorful image colorization. In *European conference on computer vision*, pages 649–666. Springer, 2016.

[164] Da-Wei Zhou, Hai-Long Sun, Jingyi Ning, Han-Jia Ye, and De-Chuan Zhan. Continual learning with pre-trained models: A survey. *arXiv preprint arXiv:2401.16386*, 2024.

[165] Da-Wei Zhou, Fu-Yun Wang, Han-Jia Ye, and De-Chuan Zhan. Pycil: a python toolbox for class-incremental learning. *SCIENCE CHINA Information Sciences*, 2023.

[166] Jinghao Zhou, Chen Wei, Huiyu Wang, Wei Shen, Cihang Xie, Alan Yuille, and Tao Kong. ibot: Image bert pre-training with online tokenizer. *arXiv preprint arXiv:2111.07832*, 2021.

[167] Fei Zhu, Zhen Cheng, Xu-yao Zhang, and Cheng-lin Liu. Class-incremental learning via dual augmentation. *Advances in Neural Information Processing Systems*, 34:14306–14318, 2021.

[168] Fei Zhu, Xu-Yao Zhang, Chuang Wang, Fei Yin, and Cheng-Lin Liu. Prototype augmentation and self-supervision for incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5871–5880, 2021.

[169] Kai Zhu, Wei Zhai, Yang Cao, Jiebo Luo, and Zheng-Jun Zha. Self-sustaining representation expansion for non-exemplar class-incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9296–9305, 2022.

[170] Kai Zhu, Kecheng Zheng, Ruili Feng, Deli Zhao, Yang Cao, and Zheng-Jun Zha. Self-organizing pathway expansion for non-exemplar class-incremental learning. In *International Conference on Computer Vision (ICCV)*, 2023.

[171] Simone Zini, Alex Gomez-Villa, Marco Buzzelli, Bartłomiej Twardowski, Andrew D Bagdanov, and Joost van de Weijer. Planckian jitter: countering the color-crippling effects of color jitter on self-supervised training. *Proceedings of the International Conference on Learning Representations*, 2023.