NEW INSIGHTS ON CATASTROPHIC FORGETTING IN NEURAL NETWORKS

Nader Asadi

A thesis
in
The Department
of
Computer Science and Software Engineering

Presented in Partial Fulfillment of the Requirements
For the Degree of Master of Science (Computer Science) at
Concordia University
Montréal, Québec, Canada

April 2023

Concordia University
School of Graduate Studies

This is to certify that the thesis prepared

By: Nader Asadi

Entitled: **New Insights on Catastrophic Forgetting in Neural Networks**

and submitted in partial fulfillment of the requirements for the degree of

Master of Science (Computer Science)

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee:

Dr. Thomas Fevens — Chair

Dr. Thomas Fevens — Examiner

Dr. Yang Wang — Examiner

Dr. Eugene Belilovsky — Co-supervisor

Dr. Sudhir P. Mudur — Supervisor

Approved by

Dr. Leila Kosseim, Graduate Program Director
Department of Computer Science and Software Engineering

_____ 20 _____

Dr. Mourad Debbabi, Dean
Faculty of Engineering and Computer Science

Abstract

New Insights on Catastrophic Forgetting in Neural Networks

Nader Asadi

Continual learning, the ability of agents to learn from a changing distribution of data while respecting memory and compute constraints, is a challenging and important problem in machine learning. The central challenge of Continual Learning (CL) is to balance effective adaptation of new information while combating catastrophic forgetting, *i.e.* stability-plasticity dilemma. This thesis is comprised of four major chapters that explore different aspects of continual learning and propose novel solutions to address some of its challenges.

First, we investigate the impact of experience replay (ER) on the change in representations of observed data that arises when previously unobserved classes appear in the incoming data stream. We show that applying ER causes the representations of newly added classes to overlap significantly with the previous classes, leading to highly disruptive parameter updates. To mitigate this issue, we propose a new method that shields the learned representations from drastic adaptation to accommodate new classes. Specifically, we use an asymmetric update rule that pushes new classes to adapt to the older ones, which is more effective, especially at task boundaries where much of the forgetting typically occurs. Empirical results on standard continual learning benchmarks show significant gains over strong baselines.

Then, we focus on the concept of representation forgetting, which refers to the change in a model's representation without losing knowledge about prior tasks. We observe that models trained without any explicit control for forgetting often experience small representation forgetting, which can sometimes be comparable to methods that explicitly control for forgetting, especially in longer task sequences. We propose a simple yet competitive approach to learning representations continually with standard supervised contrastive learning while constructing prototypes of class samples when queried on old samples. We show that this approach can lead to new insights on the effect of model capacity and loss function used in continual learning.

Finally, we address the challenge of balancing effective adaptation while combating catastrophic forgetting, *i.e.* stability-plasticity dilemma, without relying on prior

task data. We propose a holistic approach to jointly learn the representation and class prototypes while maintaining the relevance of old class prototypes and their embedded similarities. We use a supervised contrastive loss to learn representations in an embedding space and evolve class prototypes continually in the same latent space, enabling learning and prediction at any point. To continually adapt the prototypes without keeping any prior task data, we propose a novel distillation loss that constrains class prototypes to maintain relative similarities as compared to new task data. Empirical results show that this method yields state-of-the-art performance in the task-incremental setting and provides strong performance in the class-incremental setting without using any stored data points.

Overall, in this thesis, we provide new insights and methods for effective adaptation in CL without catastrophic forgetting. The proposed methods achieve state-of-the-art performance on standard continual learning benchmarks and provide new insights into the role of model capacity, loss functions, and forgetting in CL.

Acknowledgments

During my two-year Master's program, I had an incredible learning experience and met many amazing people who contributed to my journey. My supervisors, Sudhir Mudur and Eugene Belilovsky, deserve special thanks for guiding me to become a better researcher and giving me the freedom to explore my research interests. They were always willing to discuss my work and help me improve my papers. I am grateful for their guidance in showing me the guidelines of doing great research and shaping my thoughts.

I would like to take this opportunity to express my heartfelt gratitude to Rahaf Aljundi for her invaluable guidance, support, and mentorship during my research projects. Her extensive knowledge, expertise, and passion for research have been a constant source of inspiration for me. I am also grateful to my collaborators, MohammadReza Davari and Lucas Caccia. They taught me how to approach research problems with creativity and a critical mindset, and their encouragement and belief in my abilities boosted my confidence as a researcher.

I would also like to thank Soroush Saryazdi, Amir Sarfi, and Farzad Salajegheh for their thoughtful interactions that contributed to my thinking and research philosophy. Your feedback, whether it was positive or constructive, helped me to refine my research methodology and approach complex problems from different perspectives.

I am thankful to Mahdi Eftekhari, Hadis Mohseni, and Farid Saberi-Movahed, my supervisor and advisors during my bachelor's studies, for putting their trust in me submitting recommendation letters on my behalf.

I would like to take this opportunity to express my sincere gratitude and appreciation to my friends, family, and girlfriend, who have supported me throughout my Master's degree journey. Your unwavering encouragement, love, and support have been invaluable to me, and I could not have made it this far without your help. I am grateful to have you all in my life, thank you again for being my support system, and I appreciate everything you have done for me.

Contents

List of Figures

List of Tables

List of Abbreviations

CIL Class Incremental Learning

CL Continual Leanring

CNN Convolutional Neural Network

EG Expert gate

ER Experience Replay

EWC Elastic Weight Consolidation

GEM Gradient Episodic Memory

i.i.d Independent and identically distributed

iCaRL Incremental Classifier and Representation Learning

IMM Incremental Moment Matching

KD Knowledge Distillation

KL Kullback-Leiber divergence

LP Linear Probe

LwF Learning without Forgetting

MAS Memory Aware Synapses

MLE Maximum Likelihood Estimation

MLP Multi-Layer Perceptron

NME Nearest Mean Examplar

NN Neural Network

ReLU Rectified Linear Unit

SGD Stochastic Gradient Descent

SI Synaptic Intelligence

SSL Self-supervised Learning

Chapter 1

Introduction

1.1 Overview

The world we live in is constantly changing and our brains are continuously evolving to adapt to these changes. The non-stationary nature of evolutionary systems in living creatures contrasts with the stationary world described by classical physics. To create effective machine intelligence, our artificial agents must be able to adapt and exploit new information in the same dynamic, non-stationary world. Without mechanisms to constantly adapt, effective machine intelligence cannot be achieved.

Neural networks, represented by current machine learning models, can learn and even surpass human-level performance in individual tasks. However, the learning process creates static neural models that cannot adapt or expand their functions. When faced with new data, the neural network training process must start from scratch. This practice becomes intractable in real-world scenarios where data streams, disappears, or cannot be stored due to storage constraints or privacy issues. Neural networks must be able to adapt and update over time to keep up with the constantly evolving world.

The main obstacle to developing continually adapting systems is catastrophic forgetting, where the learning of new data completely erases previously acquired knowledge. Although catastrophic forgetting has been observed in neural networks, the ability of these networks to implicitly store acquired knowledge highlights the need to understand this phenomenon.

Humans tend to learn concepts sequentially and revisit some concepts, but the complete loss of previous knowledge is rare. Artificial neural networks cannot learn in this manner due to catastrophic forgetting. Revisiting training data improves performance, but this is not the case for humans. Research shows that learning

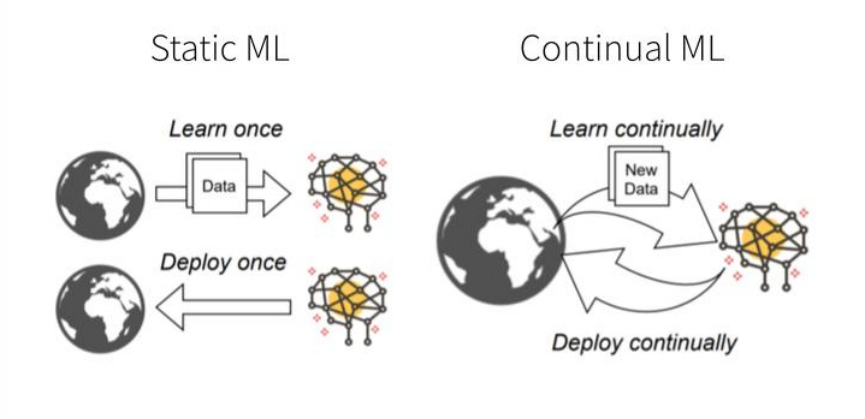


Figure 1: An illustration of the continual machine learning versus static machine learning.

two events sequentially in rats results in complete forgetting of the first event once the second is learned. However, concurrent learning of the same two events shows forgetting, but not catastrophic forgetting. Overcoming catastrophic forgetting in higher mammals may be due to the development of a hippocampal-neocortical separation.

Catastrophic interference is caused by the stability-plasticity dilemma in neural networks. While plasticity allows for the integration of new knowledge, stability preserves previous knowledge while new data is encoded. To build intelligent systems that are dynamic and sensitive to new information while being stable and immune to catastrophic interference, the challenge is to balance stability and plasticity. This challenge has been the driving goal of the work conducted during this master's studies.

1.2 Continual Learning

Lifelong learning, also known as continual learning, incremental learning, or sequential learning, is the study of learning from an infinite stream of data that comes from changing input domains and different tasks. The ultimate goal is to use this acquired knowledge in problem-solving and future learning [6]. The key feature of continual learning is the sequential nature of the learning process, where only a small portion of input data from one or a few tasks is available at once. It is impossible to label all training examples from all tasks before initiating the learning process, and with a constantly evolving world, adaptation and continual learning are essential. However, to be efficient, previously seen data should not be stored in their raw format, and full

re-training at each point is not feasible at such a large scale.

Researchers have studied the catastrophic forgetting problem since the early development of neural networks and proposed that parameter sharing, which allows neural networks to generalize from seen data, is the reason behind catastrophic forgetting [7, 8]. After learning one task, the network parameters correspond to one point in the parameter space. When learning a new task, the parameters will change their values to a new point that might not correspond to a solution to the first task. It has been shown that the parameter space of shallow networks contains cliffs in which small moves lead to a severe change in the function output [9].

Early research developed several strategies to mitigate forgetting without storing the training data, mostly on a small scale of a few examples and considering shallow networks [10–12]. Recently, with the revival of neural networks, the catastrophic forgetting problem and the continual learning paradigm have received increased attention [13–16]. We first define the general continual learning setting and describe the main desired criteria of a continual learning system.

General Continual Learning Setting The general continual learning setting considers an infinite stream of data where at each time step t, the system receives a new sample(s) $\{x_t, y_t\}$ drawn *non i.i.d.* from a current distribution Q that could itself experience sudden or gradual changes. The main goal is to learn a function f parameterized by θ that minimizes a predefined loss ℓ on the new sample(s) without interfering with and possibly improving on those that were learned previously.

1.3 Main Contributions

This master's thesis is structured around mitigating catastrophic forgetting in continual learning settings. We aim to propose solutions to provide artificial agents with the ability to learn continually and accumulate the acquired knowledge over their lifetime. We considered various specific situations and approaches presented in several chapters:

Chapter 3: Mitigating Representation Drift in Online Continual Learning The focus of our work is on how representations of observed data change when new classes appear in the incoming data stream, and the challenge of distinguishing them from previous classes. We discovered that using ER causes the newly added classes to overlap significantly with previous classes, resulting in disruptive parameter

updates. To address this issue, we propose a new method that shields the learned representations from drastic adaptation to accommodate new classes. Our method uses an asymmetric update rule that pushes new classes to adapt to the older ones, which is more effective at task boundaries where forgetting often occurs. We tested our method on standard continual learning benchmarks and achieved significant gains over strong baselines. The work in this chapter has led to a conference publication:

- Caccia, Lucas, Rahaf Aljundi, Nader Asadi, Tinne Tuytelaars, Joelle Pineau, and Eugene Belilovsky. "New Insights on Reducing Abrupt Representation Change in Online Continual Learning." In *International Conference on Learning Representations (ICLR) 2022*.

Chapter 4: One-Class Incremental Learning with Regularized Class Prototypes Unlike other continual learning scenarios, where a learner is presented with new samples multiple times, in this setting, the learner is presented with new samples only once and must distinguish between all seen classes. To overcome this challenge, some successful methods store and replay a subset of samples alongside incoming data in an efficient manner. One recent method called ER-AML achieved strong performance by using an asymmetric loss based on contrastive learning to distinguish incoming data from replayed data. However, the proposed method avoids contrasts between incoming and stored data, which makes it impractical when only one new class is introduced in each phase of the stream. In this work, we adapt a recently proposed approach (BYOL [17]) from self-supervised learning to the supervised learning setting, unlocking the constraint on contrasts. We then show that by supplementing this with additional regularization on class prototypes, we achieve a new method that performs strongly in the one-class incremental learning setting and is competitive with the top-performing methods in the multi-class incremental setting. The work in this chapter has led to a workshop publication:

- Asadi, Nader, Sudhir Mudur, and Eugene Belilovsky. "Tackling Online One-Class Incremental Learning by Removing Negative Contrasts." In *NeurIPS 2022 Workshop on Distribution Shifts (DistShift)*.

Chapter 5: Exploring Representation Forgetting and its Implications We explore the idea of representation forgetting, which is identified by the difference in performance of an optimal linear classifier before and after introducing a new task. By using this tool, we revisit various standard continual learning benchmarks

and find that model representations trained without explicit control for forgetting often experience small representation forgetting and can sometimes be as effective as methods that explicitly control for forgetting, particularly in longer task sequences. We also discover that representation forgetting provides new insights into the impact of model capacity and loss function used in continual learning. Based on our findings, we suggest a simple yet effective approach of continually learning representations using standard supervised contrastive learning while generating prototypes of class samples when queried on old samples. The work in this chapter has led to a conference publication:

- Davari, MohammadReza*, Nader Asadi*, Sudhir Mudur, Rahaf Aljundi, and Eugene Belilovsky. "Probing representation forgetting in supervised and unsupervised continual learning." In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16712-16721. 2022.
  * denotes equal contribution

Chapter 6: Replay-Free Continual Learning with Evolving Class Prototypes In this research, we aim to overcome the need to use previous task data by starting with robust representation learning methods that have been demonstrated to be less susceptible to forgetting. We propose a comprehensive approach to simultaneously learn the representation and class prototypes while maintaining the relevance of old class prototypes and their embedded similarities. Specifically, we map samples to an embedding space where we learn representations using a supervised contrastive loss, and continually evolve class prototypes in the same latent space, enabling learning and prediction at any point. To continually adapt the prototypes without retaining any prior task data, we propose a novel distillation loss that constrains class prototypes to maintain relative similarities compared to new task data. The work in this chapter is in submission as a conference paper, at the time of writing this thesis. The preprint can be found as:

- Asadi, Nader, MohammadReza Davari, Sudhir Mudur, Rahaf Aljundi, and Eugene Belilovsky. "Prototype-Sample Relation Distillation: Towards Replay-Free Continual Learning." *arXiv preprint arXiv:2303.14771* (2023).

Chapter 2

Background

In this chapter, we explain briefly the background materials related to the works presented in this manuscript.

2.1 Neural Networks

Given a training dataset $D = \{x, y\}_{n=1}^{N}$ sampled i.i.d. from a target distribution $Q_{x,y}$, a neural network is employed to learn a function f that maps a given input x_n to a target output y_n. For example in handwritten digit recognition, x_n represents an image while y_n corresponds to the drawn digit. They are called networks because the learned function is composed of multiple functions, *e.g.* $f(x) = f_3(f_2(f_1(x)))$ where f_1 is the first layer, f_2 is the second layer and f_3 is the output layer. While the target y_n represents the desired output for the output layer, there is no target output for the other layers and they are called hidden layers. Each layer, represented by f_l, is parametrized by a weight vector $\theta_l = \{\theta_{ij}\}$ with a vector input and a vector output, $f_l(x) = \phi(\theta_l x)$ where ϕ is the activation function. The layer can be viewed as a group of neurons, also called units or perceptrons, where each neuron maps a vector input to a scalar output, hence the name multi-layer perceptron, see Fig. 2 for an example. ϕ is usually a non-linear function except maybe from the output layer. By stacking multiple layers of neurons, one can approximate various non-linear functions. In fact, neural networks can be seen as universal approximators if provided with enough neurons in a hidden layer. The learning occurs through minimizing a loss function that represents the divergence between the output of the neural network and the target output. Due to the non-linearity of the neural networks, most cost functions become non-convex; and they are usually optimized by iterative gradient descent steps. The gradients of the loss function are computed through the backpropagation

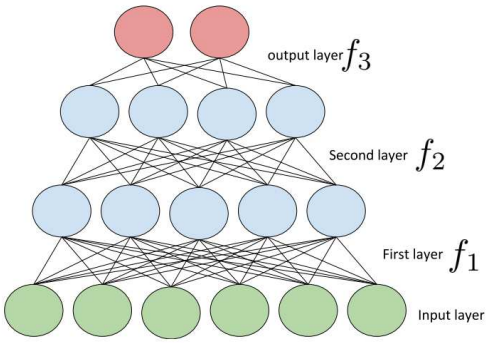


Figure 2: An example of a 3 layers neural network.

[131] of the estimated loss to the preceding layers of the network. Typically, the optimizer performs gradient steps on batches randomly sampled from the training set. This process is repeated and samples are re-visited until convergence. After a local minimum is reached, the neural network is deployed and used for performing predictions on new samples. You can see a mini-batch stochastic gradient descent algorithm in the following:

Algorithm 1: Optimizing a neural network with gradient descent.

input: Dataset D with pairs of (x_i, y_i)
input: Loss function $\mathcal{L}$
input: Model function f_θ with θ as the learnable parameters
input: Learning rate η //
while *stopping criterion not satisfied* **do**
  $(x, y) \leftarrow$ sample mini-batch of size b from D
  Forward pass: $\hat{y} \leftarrow f_\theta(x)$
  Compute loss: $\mathcal{L} \leftarrow \mathcal{L}(\hat{y}, y)$
  Compute the gradients: $\delta \leftarrow \nabla_\theta \mathcal{L}$
  Update all parameters: $\theta \leftarrow \theta - \eta\delta$
end

The learning rate η controls the step size when updating the parameters in the direction of the gradient. The batch size b defines the number of images seen during one update. The backpropagation algorithm [18] computes the gradients and the update of the parameters. In the case of image classification and segmentation, the main topics explored in this thesis, we discern the feature extractor from the classifier in the neural network. The former transforms the input space so that the latter

linearly discriminates classes. We refer to [19] for detailed information on deep neural networks.

2.2 Parameter Estimation and Regularization

Maximum Likelihood Estimation Given a dataset $D = \{x, y\}_{n=1}^{N}$ sampled *i.i.d.* from a target distribution $Q_{x,y}$ and a model with parameters $\theta = \{\theta_{ij}\}$, the likelihood of the data given the model parameters is defined as $\mathcal{L}(\theta) = p(D|\theta)$. A maximum likelihood estimator of the model parameters θ is:

$$\theta^{MLE*} = \max_{\theta} \; p(D|\theta) \tag{1}$$

where θ^{MLE*} is the model parameters that maximize the likelihood of the data D. Usually, in optimization, we minimize an objective function, *e.g.* loss function. As such instead of maximizing the likelihood, we minimize the negative log-likelihood.

$$\theta^{MLE*} = \min_{\theta} \; -\log\left(p(D|\theta)\right) \tag{2}$$

Regularization The dataset used for training can be small and noisy while the model is typically overparameterized. Minimizing the log-likelihood of the training data could result in overfitting, i.e. small training errors but higher test errors. To reduce overfitting, regularization is imposed to control the model variance without a significant increase in the bias. Many regularization techniques aim at limiting the model capacity by adding a penalty that corresponds to the norm of the model parameters. As examples we mention:

L2 Regularization An ℓ_2 norm penalty which is usually referred to as weight decay. It pulls the parameters towards the origin, keeping them with small magnitude. L2 regularization is also known as ridge regression or Tikhonov regularization. Minimizing the penalized objective would lead to:

$$\theta^{MLE*} = \min_{\theta} \; -\log\left(p(D|\theta)\right) + \frac{1}{2}\|\theta\|_2^2 \tag{3}$$

L1 Regularization Instead of ℓ_2 penalty, ℓ_1 norm is also a popular norm for regularization. It is defined as $\|\theta\|_1 = \sum_{i,j}|\theta_{i,j}|_2^2$. ℓ_1 norm results in more sparse parameters than ‘2 norm and it has been used for feature selection as in the LASSO [20] (least absolute shrinkage and selection operator).

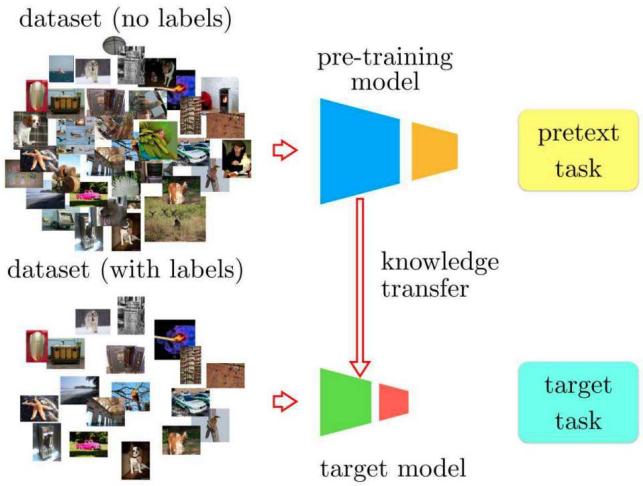


Figure 3: A block diagram of self-supervised learning algorithms.

2.3 Self-supervised and Deep Metric Learning

The ability to learn rich patterns from the abundance of data available today has made the use of deep neural networks a compelling approach in the majority of computer vision and natural language processing tasks. Even though there is a plethora of data available, the lack of annotations has pushed researchers to find alternative approaches that can leverage them. This is where self-supervised methods play a vital role in fueling the progress of deep learning without the need for expensive annotations and learn feature representations where data provide supervision.

Self-supervised learning can be considered as a bridge between supervised and unsupervised learning. The goal of self-supervised learning is to utilize a supervisory signal which naturally exists in the data, to learn generalizable representations to downstream tasks. This is also called predictive or pretext learning. The self-supervised algorithm has the ability to auto-generate the labels for unlabelled data, which converts the unsupervised model to a supervised model. Fig. 3 shows a high-level overview of self-supervised learning pipeline.

Pretext Task Early works in self-supervised learning methods rely on all sorts of pretext to learn visual representations. For example, colorizing gray-scale images [21], image jigsaw puzzle [22], image super-resolution [23], image inpainting [24], predicting a relative offset for a pair of patches [25], predicting the rotation angle [26], and image reconstruction [27, 28]. Although these methods have shown their effectiveness, they lack the generality of the learned representations.

Instance Discrimination The recent contrastive learning methods [29–32] have made a lot of progress in the field of self-supervised learning. Most of the previous contrastive learning methods are based on the instance discrimination [33] task in which positive pairs are defined as different views of the same image, while negative pairs are formed by sampling views from different images. SimCLR [29, 34] shows that image augmentation (e.g.Grayscale, Random Resized Cropping, Color Jittering, and Gaussian Blur), nonlinear projection head and large batch size plays a critical role in contrastive learning. SimCLR loss is given by:

$$\mathcal{L}_{SimCLR} = -\sum_{\mathbf{x}_i \in \mathbf{X}} \log \frac{\texttt{sim}(f_\theta(\mathbf{x}_p(i)), f_\theta(\mathbf{x}_i))}{\sum_{\mathbf{x}_n \in \mathbf{A}(i)} \texttt{sim}(f_\theta(\mathbf{x}_n), f_\theta(\mathbf{x}_i))} \tag{4}$$

where $\texttt{sim}(a, b) = \exp(\frac{a^T b}{\tau \|a\| \|b\|})$, $A(i)$ corresponds to all minibatch examples and their data augmentations except x_i, and $x_p(i)$ represents an augmented version of x_i.

Contrastive loss functions are also becoming a popular alternative to cross-entropy (CE) in the supervised setting [35,36], referred to as SupCon. Given a representation f_θ, often consisting of a primary network and a projection, the SupCon loss for a minibatch X is given by:

$$\mathcal{L}_{SupCon} = \sum_{\mathbf{x}_i \in \mathbf{X}} \frac{-1}{|P(i)|} \sum_{\mathbf{x}_p \in P(i)} \log \frac{\texttt{sim}(f_\theta(\mathbf{x}_p), f_\theta(\mathbf{x}_i))}{\sum_{\mathbf{x}_a \in \mathbf{X}/x_i} \texttt{sim}(f_\theta(\mathbf{x}_a), f_\theta(\mathbf{x}_i))} \tag{5}$$

where $P(i)$ represents the same class samples as x_i from the minibatch, and the denominator is taken over all other samples.

Contrastive Learning without Negatives Most previous contrastive learning methods prevent the model collapse in an explicit manner (*e.g.* push different instances away from each other or force different instances to be clustered into different groups). BYOL [17] can learn a high-quality representation without negatives. Specifically, it trains an online network to predict the target network representation of the same image under a different augmented view and using an additional predictor network on top of the online encoder to avoiding the model collapse. SimSiam [37] shows that simple Siamese networks can learn meaningful representations even without the use of negative pairs, large batch size, and momentum encoders.

2.4 Knowledge Distillation

Knowledge distillation [38] is a technique that aims at transferring the knowledge embedded in one model to another. Usually, a big model or an ensemble of models is used to learn a function from a big training dataset, which is quite computationally demanding. This technique aims at distilling the knowledge from the big/ensemble to a smaller model that is much lighter to deploy. The probabilities of each class in the output layer of the big model convey richer information than the label of the image. It also captures the similarities/dissimilarities between the different classes that are learned by the large model. Suppose that $\hat{y}$ is the output of the light model and y^* is the output of the large model. Then the knowledge distillation loss is defined as:

$$\ell_{dist}(\hat{y}, y^*) = -\langle z^*, \log \hat{z} \rangle \tag{6}$$

where log is operated entry-wise and

$$z_i^* = \frac{y_i^{*1/\tau}}{\sum_j y_j^{*1/\tau}} \quad and \quad \hat{z}_i = \frac{\hat{y}_i^{1/\tau}}{\sum_j \hat{y}_j^{1/\tau}} \tag{7}$$

where $i, j \in \{1, ..., J\}$ running over the output layer units, τ is the temperature. The application of a high temperature τ softens the probability distribution over the classes. Different forms of knowledge distillation can be used to reduce forgetting by distilling the knowledge from a previous model to a model being trained on a new task, as we show in Chapter 6.

2.5 Continual Learning Terminology

In this manuscript, we use terms that have become common in the continual learning community. You can see a short description of frequently used terms in the following:

- **Catastrophic Forgetting**: The complete erase of the previously acquired knowledge once new knowledge is learned.
- **Task**: A more specific term than "problem" with a predefined input and output. For example, scene classification is a task of the image classification problem.
- **Joint Training**: Indicates the training of a shared model on multiple tasks using all their data.

- **Fine-tuning**: Indicates training a given task using as initialization a model pre-trained on another task, typically with large amount of labelled data. It is the most typical form of transfer learning.
- **Replay**: When learning a new task or new samples, samples from previous history are re-intorduced to the learner.
- **Rehearsal**: Performing learning steps on the replayed samples from previous history when learning a new task or new samples.

2.6 Continual Learning Scenarios

Continual Learning involves training a model on a continually changing dataset without forgetting previous knowledge. In traditional training, it is assumed that the dataset is fixed and independent and identically distributed, but transfer learning can be used to transfer knowledge from one dataset to another. In Continual Learning, new classes or samples from new domains are continuously added to the training dataset, and the test dataset is assumed to evolve similarly. Various types of distribution drifts exist in Continual Learning and that they are referred to by different names in literature. The major drifts involve changes in the input sample and its ground-truth label, such as in image classification or semantic segmentation. The major drifts are:

- **Covariate drift:** when $p(x)$ changes, it happens with the introduction of new domains [39].
- **Prior drift:** when $p(y)$ changes. *Class-Incremental Learning* happens with this kind of drift.
- **Conceptual drift:** when $p(y|x)$ changes. Seldom covered in the literature, it can be found in Continual Semantic Segmentation.

The idea of training a new model on an ever-growing dataset can be achieved by starting from scratch and using the combination of past and new data. However, there are constraints such as privacy concerns, limited time, or limited storage capacity that limit the amount of previous data that can be kept. In some cases, the model may only have access to new data and not the old data. If the parameters are initialized randomly, the model will not be able to predict the past data distribution. To overcome this, the new model parameters can be initialized using the previous

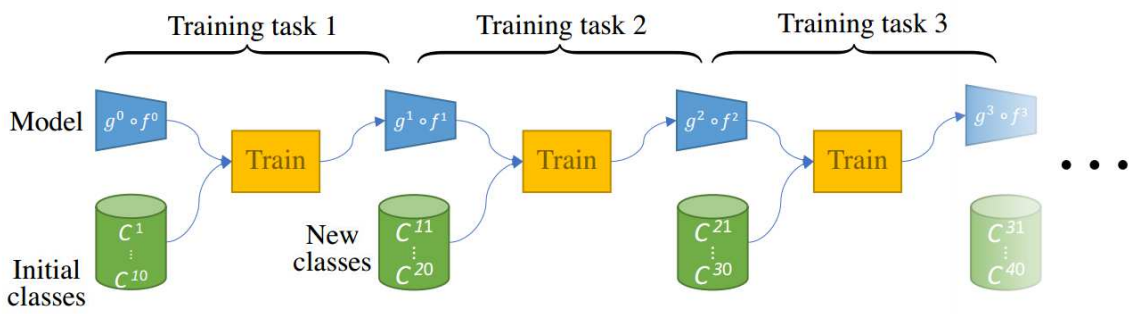


Figure 4: Continual learning overview. At each training task we learn a new set of classes, and the model must retain knowledge about all classes.

parameters ($\theta^t := \theta_*^{t-1}$). It's important to note that the optimization procedure for the old $(t-1)$ model and the new (t) model will be different.

Evidently, the optimal parameters θ_* are different for the task t and $t-1$. This difference results in what we call a forgetting: θ_*^t is optimal for the new task t but is suboptimal for the task $t1$, therefore performance on the previous task may be degraded ($\mathcal{L}(f_{\theta^t}, D^{t1}) \gg \mathcal{L}(f_{\theta^{t-1}}, D^{t1})$). This performance drop is actually so important that the literature names it *catastrophic forgetting* [11]. It is particularly critical in the context of image classification where each new task brings new classes to be classified as described below.

Class-Incremental Learning A well-known benchmark in Class Incremental Learning (CIL) involves learning the classification dataset, consisting of N classes, in multiple stages. Each stage involves learning several new classes. During the first stage ($t = 1$), the model f_1 learns the first k classes ($\mathcal{C}^{t=1}$). The second stage ($t = 2$) begins by initializing the model f_2 with the parameters learned in the previous stage f_1 and then adding 10 more classes ($\mathcal{C}^{t=2}$) to its knowledge. This process continues until all 100 classes have been learned in the final stage ($t = T$). After each stage, the model is evaluated on the testing data for all the classes that it has seen so far ($\mathcal{C}^{1:t}$). Fig. 4 shows the continual protocol and Fig. 4 compares the performance of a model that was trained from scratch for each task and a model that was continually fine-tuned with new data. The latter tends to overpredict new classes and has a lower overall accuracy due to the phenomenon of *catastrophic forgetting*, which highlights the importance of addressing this issue.

Single-Head vs Multi-Head The two primary ways of evaluating in Continual Learning are Single-Head and Multi-Heads, according to [40]. In the Single-Head setting, the model must identify samples among all classes it has seen ($\mathcal{C}^{1:t}$), regardless of the step in which they were learned. Conversely, in the Multi-Heads evaluation, the model knows the step/task identifier (i) of the samples during testing, allowing it to classify samples based solely on the limited number of classes from that step ($\mathcal{C}^i$). This form of evaluation is closely connected to multi-task learning. This thesis focuses on Single-Head evaluation, as it is more practical, as it is rarely possible to determine the origin of a sample in real-life situations. Additionally, Single-Head evaluation is more challenging as it requires a single classifier to discriminate among all classes from all tasks, instead of having a separate classifier for each task [41].

Evaluation Metrics Multiple metrics exist in Continual Learning: the most common are the final accuracy and average incremental accuracy. The former measures the performance of the model on all tasks at the last step, the latter measures the average of performance on all seen tasks after each new task learned [42]. Practically, given $A_{i,t}$ the accuracy of the i^{th} task after learning the t^{th} task, the final accuracy is (assuming balanced tasks):

$$\text{Acc}_F = \frac{1}{T}\sum_{i=1}^{T} A_{i,T} \tag{8}$$

and the average incremental accuracy:

$$\text{Acc}_a = \frac{1}{T}\sum_{t=1}^{T}\frac{1}{t}\sum_{i=1}^{t} A_{i,T} \tag{9}$$

Average incremental accuracy is somewhat more important than simply the final accuracy: a continual model should be good after every step because in a true continual setting, there is not a "final task".

There are other metrics besides performance, such as forgetting (as recorded by [43]), which measures the decline in a model's performance on a task compared to when it was initially learned. This metric is valuable because it is independent of the model's overall performance. Additionally, metrics like processing speed (i.e. the number of images processed per second) and capacity utilization (i.e. the number of parameters learned) are crucial to consider, as demonstrated by [1], who found that larger models have lower forgetting rates.

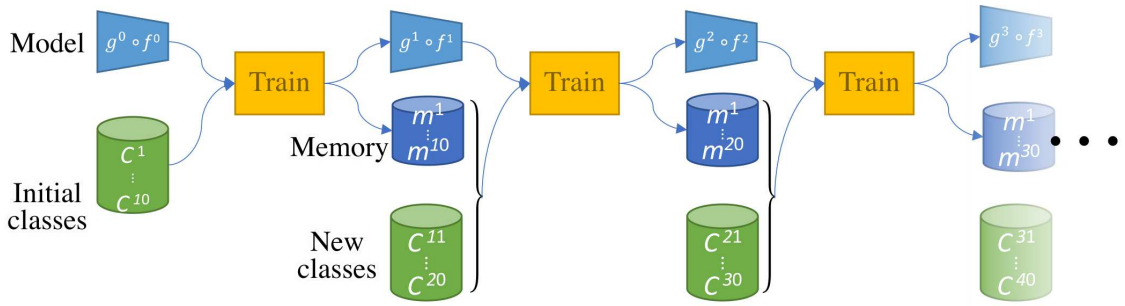


Figure 5: Overview of continual learning with rehearsal buffer. Following each task, a portion of the data is saved in a memory for use in the next task. While rehearsal learning is the most effective way to minimize forgetting, the capacity of the memory is frequently inadequate.

2.7 Continual Learning Methods

There are several methods to combat forgetting in Continual Learning, including replay of old data, regularization to control the model's behavior, and adjusting the model's structure.

2.7.1 Rehearsal-based Methods

The most effective way to prevent forgetting in continual learning is to use rehearsal learning, where old samples are shown alongside new samples. However, the number of old samples that can be stored is limited, as storing too many would go against the purpose of continual learning. Fig. 5 demonstrates how rehearsal learning happens in Continual Learning. In the first step, the model is trained on all available samples and then a limited number of those samples are stored in a memory. In the second step, the model has access to both new samples and all previously stored samples in the memory. In Class-Incremental learning, an equal number of samples per class is stored in the memory. Two approaches have been proposed to determine the number of samples: [42] suggest fully utilizing a memory of size $\mathcal{M}$ among all $\mathcal{C}$ classes, while [44] keep the number of samples stored per class fixed to M divided by the number of classes $\mathcal{M}/|\mathcal{C}^{1:T}|$.

Herding refers to selecting which samples per class should be stored in the rehearsal memory. One simple method is to randomly sample images, which surprisingly yields comparable results to more complex methods [45]. Similarly, in active learning,

random sampling has been found to be effective [46]. Other herding methods include selecting samples close to the class mean in feature space [45] or close to an incremental barycenter [42].

Sampling is a crucial but relatively unexplored topic in continual learning. Many models combine all memory samples with new samples without any under- or over-sampling. [45] suggest fine-tuning on a balanced set of old and new class samples for a few epochs after training on a new step. [47] oversample tiny memory with just one sample per class and demonstrate that continual models do not overfit in the context of online learning where models learn in only one epoch. In the same context, [48] propose oversampling the memory examples with the highest losses. In imbalanced situations for continual learning, over- and under-sampling can be applied depending on the number of samples per class.

Efficient storing is essential for rehearsal learning because a larger rehearsal memory leads to less forgetting [44]. Consequently, multiple works explore how to store more rehearsal samples with the same memory size. For instance, [49] compress intermediate features of memory samples with a lossless compression algorithm. [50] also store features but modify them during training to handle inherent internal covariate drift.

Pseudo-rehearsal generates pseudo-samples for rehearsal instead of storing actual samples [51]. The generation can be achieved with auto-encoders from intermediate features [52,53] or by using a generative adversarial network (GAN) [54]. Unfortunately, these methods have several drawbacks, such as difficulty in scaling to large images, requiring a generator size greater than a classic rehearsal memory size, and the generator itself may experience catastrophic forgetting [55]. Alternatively, [56] (2020) propose a method halfway between rehearsal and pseudo-rehearsal, where real images are randomly sampled and slightly modified via bi-level optimization [57] to minimize forgetting during continual training.

2.7.2 Regularization-based Methods

Fig. 6 shows that one effective and frequently used approach to alleviate forgetting is to limit the variance in performance between the old and new models. Multiple methods can be used to impose these restrictions, which are detailed below.

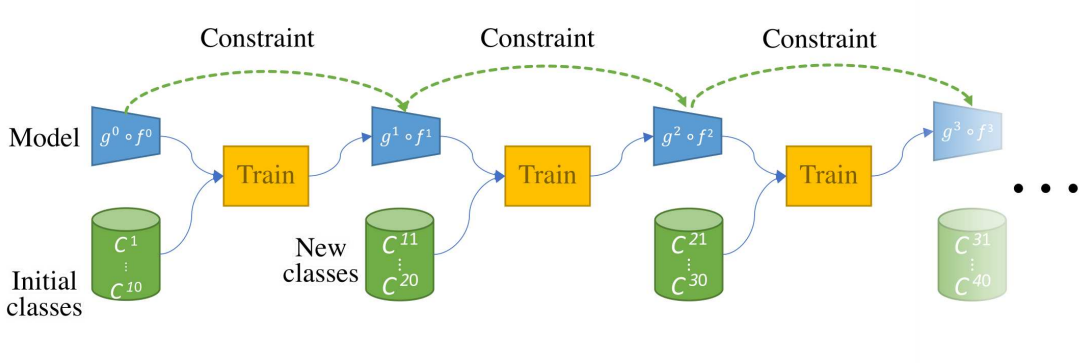


Figure 6: Constraining the new model based on the old model. The new model is restrained to resemble the old model during each task after the initial one. This entails constraining the new model to maintain similarity with the old one.

Weight-based Regularization The easiest way to prevent complete forgetting is to keep the old and new models the same. However, this approach makes the model rigid, unable to change and learn new tasks. Thus, a line of research proposed to constrain only a portion of the neurons:

$$\mathcal{L}\left(\theta^t, \theta^{t-1}\right) = \mathcal{L}_t\left(\theta^t\right) + \lambda \sum_i \Omega_i^{t-1}\left(\theta_i^t - \theta_i^{t-1}\right)^2 \tag{10}$$

where $L_t(\theta)$ is the loss at the current task t (*e.g.* the cross-entropy), θ_i^t and θ_i^{t-1} respectively the i^{th} neuron of the current and previous model, and Ω_i^{t-1} a neuron-wise importance factor.

The idea is to keep the important neurons from the previous task (t-1) unchanged, while adapting other neurons to fit the new task (t). [14], followed by [58] and [43], propose using the diagonal Fisher information matrix as importance factors. The motivation behind this is that the posterior $p(\theta^{t-1}|D^{t-1})$ should contain information about which parameters are important to the previous dataset, D^{t-1}. This posterior can be approximated by a Gaussian distribution whose diagonal precision is given by the diagonal Fisher information matrix. A higher value indicates a more important neuron for the previous task, and the constraint should be increased accordingly. Conversely, a lower value for a less important neuron means that it can change drastically, which facilitates learning new data. This approach strikes a balance between rigidity (no change and no forgetting) and plasticity (changing and learning new concepts). [59] use the sensitivity of the model when small perturbations are added to the neurons to measure their importance.

However, it's worth noting that weight-based constraints are typically only useful

in multi-head settings where a task identifier is available at test time. [41] show that in the single-head setting, they struggle to reduce forgetting and are outperformed by simple rehearsal learning, which is memory-intensive.

Gradient-based Regularization [60] introduced the GEM model that combines gradient constraints and rehearsal learning. The algorithm enforces that the loss on a given stored sample does not increase despite the model learning new classes. This can be achieved by requiring the gradient of a new sample (g) to be in the same direction as the gradient of all stored old samples, given a locality assumption:

$$\langle g, g_i \rangle \geq 0, \text{ for all } i \in \mathcal{M} \tag{11}$$

with $\mathcal{M}$ as the rehearsal buffer. If this constraint is violated, the new gradient g is projected to the closest gradient that satisfies the angle constraint by minimizing a quadratic program. However, the computational cost of this method can become prohibitive when the memory is too large. To address this issue, [40] proposed Averaged-GEM, which only constraints the average of all memory samples. [61] also improved GEM's speed by selecting a subset of the memory samples that maximize the feasible region.

In contrast, [62] introduced OGD, which constrains the gradients of task t to be orthogonal to gradients of task $t-1$. They use the Gram-Schmidt procedure to orthogonalize the new gradients, which allows updates for the new task that minimally interfere with the performance of old tasks.

Distillation-based Regularization The majority of Continual models evaluated on large datasets, such as ImageNet [5], use a combination of rehearsal learning and constraints on the model's outputs. Knowledge Distillation (KD) [38] is applied to the model's probabilities by LWF [13] and iCaRL [42]. This involves minimizing the Kullback-Leiber divergence (KL) between the probabilities of the old and new models.

$$\mathcal{L}_{\text{KD}} = \text{KL}\left(\text{softmax}\left(\frac{\tilde{y}^{t-1}}{\tau}\right) \middle\| \text{softmax}\left(\frac{\tilde{y}^{t}}{\tau}\right)\right). \tag{12}$$

where $\hat{y}^{t-1} = g^{t-1}(f_{t-1}(x))$ .The probabilities, also known as dark knowledge, contain additional information about the model that is useful for distilling. In Class-Incremental scenarios, the new model predicts more classes than the old model, so the KL is only applied to the logits that are common to both models. The KD is sometimes defined as the binary cross-entropy between the sigmoid-activated logits.

Probabilities are commonly constrained in models by most base losses. However, a few models have also considered constraining intermediate outputs. For instance, MK2D [63] uses the KD from both the final and auxiliary classifiers, similar to the Inception network [64]. [44] maximizes the cosine similarity between the embeddings produced by the global average pooling (GAP). [65] minimize the L1 distance between the attention maps produced by GradCam.

2.7.3 Parameter Isolation Methods

Parameter isolation methods allocate different subsets of model parameters to each task to prevent forgetting previous tasks. If there are no constraints on the architecture size, this can be achieved by freezing the learned parameters after each task and adding new branches for new tasks [16, 66]. The Expert Gate method [67] is an example that ensures no forgetting and aims for optimal performance in future tasks through knowledge transfer.

Alternatively, methods with a fixed architecture identify the parts used for previous tasks and mask them out during the training of the new task. This can be done at the parameter or neuron level [68, 69]. However, these methods usually require a task oracle to activate the corresponding masks or task branch during prediction [70]. The Expert Gate method [67] avoids this issue by learning an auto-encoder gate. Generally, this family of methods is limited to the task incremental setting and is better suited for learning long sequences of tasks when model capacity is not constrained, and optimal performance is a priority.

Chapter 3

Mitigating Representation Drift in Online Continual Learning

This work was the fruit of collaboration with Lucas Caccia, Rahaf Aljundi, Tinne Tuytelaars, Joelle Pineau, and Eugene Belilovsky. It was published as an article in ICLR 2022 [71].

3.1 Introduction

Continual learning is concerned with building models that can learn and accumulate knowledge and skills over time. A continual learner receives training data sequentially, from a potentially changing distribution, over the course of its learning process. The distribution change might be either a shift in the input domain or new categories being learned. The main challenge is to design models that can learn how to use the new data and acquire new knowledge while preserving or improving the performance of previously learned data. While different settings have been investigated of how new data are being received and learned, we focus on the challenging scenario of learning from an *online* stream of data with *new classes being introduced at unknown points in time* and where *memory and compute constraints* are applied on the learner. Additionally, we assume a shared output layer among all the learned classes [61]. This setting is different and harder than the conventional multi-head setting [72] where each new group of classes is considered as a new task with a dedicated head (classification layer), requiring a task oracle at test time to activate the correct head. The axes of our setting (online learning, no task boundary, no test time oracle, constant memory, and bounded compute) align with the main desiderata of continual learning as described in [73].

Catastrophic forgetting [7], where previous knowledge is overwritten as new concepts are learned, remains a key challenge in the online continual learning setting. To prevent forgetting, methods usually rely on storing a small buffer of previous training data and replaying samples from it as new data is learned. This can partially counteract catastrophic forgetting but still tends to lead to large disruptions in accuracy, particularly at the initial task boundary or shift in distribution. Various works focus on studying which samples to store [61, 74] or which samples to replay when receiving new data [48]. In this work, we direct our attention to the representations being learned and investigate how the features of previously learned classes change and drift over time.

Consider the time point in a stream when a new class is introduced after previous classes have been well learned. If we consider the representation being learned, incoming samples from new classes are likely to be dispersed, potentially near and between representations of previous classes, while the representations of previous classes will typically cluster according to their class. Indeed, one might expect minimal changes to the learned representation of the previous classes, while the new classes samples are pushed away from the clusters of old class data. However, with a standard Experience Replay (ER) algorithm [75], we observe that it is the representations of older classes that is heavily perturbed after just a few update steps when training on the new class samples. We hypothesize that the fundamental issue arises from the combination of: new class samples representations lying close to older classes and the loss structure of the standard cross entropy applied on a mix of seen and unseen classes. We illustrate the observed effect in Fig. 7 (left).

This behavior is exacerbated especially in the regime of low buffer size. With larger replay buffers, the learner can recover knowledge about the prior classes over time, while with smaller buffers the initial disruptive changes in representations are challenging to correct. Indeed we illustrate this effect in Fig. 7 (right), we see that ER only recovers from the initial displacement given a much larger buffer size.

In standard continual learning with replay [48,75] the same loss function is usually employed on both the newly received samples and the replayed samples. In contrast, we propose a simple and efficient solution to mitigate this representation drift by using **separate losses on the incoming stream and buffered data**. The key idea is to allow the representations of samples from new classes to be learned in isolation of the older ones first, by excluding the previously learned classes from the incoming data loss. The discrimination between the new classes and the older ones is learned through the replayed batches, but only after incoming data has been learned,

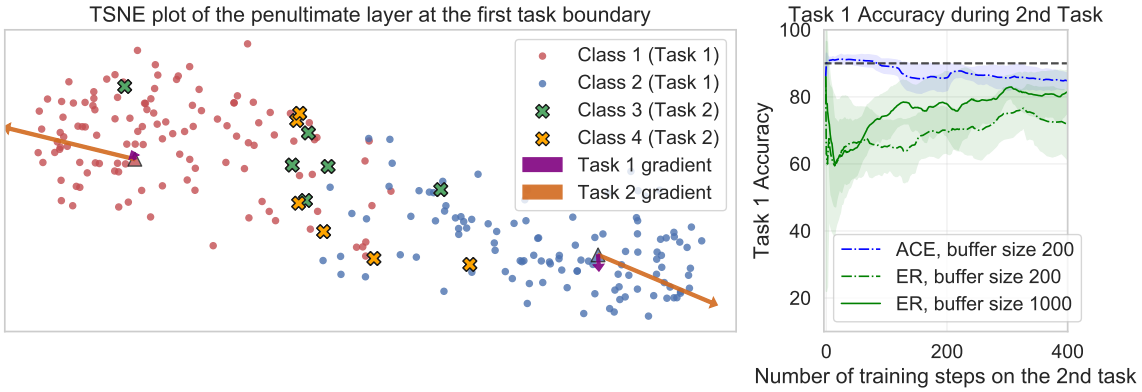


Figure 7: (left) Analysis of representations with the first task's class prototypes *at a task boundary*. Under ER when Task 2 begins, class 1 & 2 prototypes experience a large gradient and subsequent displacement caused by the close location of the unobserved sample representations, this leads to a significant drop in performance (right). Our proposed method (ACE) mitigates the representation drift issue and observes no performance decrease on a task switch.

added to the buffer, and made available for replay. To allow more direct control of the structure in representations we first consider a metric learning based loss for the incoming data, proposed in [35], where we propose to exclude samples of previously learned classes from the negative samples. We show that this type of negative selection is critical, and in contrast issues arise when negative examples are sampled uniformly from the buffer. These issues mimic those seen with standard losses in experience replay (ER) [48]. On the other hand we use a different loss on replay buffer data that is allowed to consider new and old classes, thereby consolidating knowledge across current and previous tasks. We call this overall approach ER with *asymmetric metric learning* (ER-AML).

Since cross entropy losses can be more efficient in training for classification than metric learning and contrastive losses (avoiding positive and negative selection) and it is widely used in incremental and continual learning, we also propose an alternative cross entropy solution that similarly applies an asymmetric loss between incoming and replay data. Notably, the cross entropy applied to the incoming data *only considers logits of classes of the incoming data*. This variant, named ER with *asymmetric cross-entropy* (ER-ACE), along with ER-AML show strong performance, with little disruption at task boundaries Fig. 7 (right). We achieve state of the art results in existing benchmarks while beating different existing methods including the traditional ER solution with an average relative gain of 36% in accuracy. Our improvements are especially high in the small buffer regime. We also show that the mitigation

of the old representation drift does not hinder the ability to learn and discriminate the new classes from the old ones. This property emerges from *only learning the incoming data in isolation*; as we will see, also isolating the rehearsal step (as in [76]) leads to poor knowledge acquisition on the current task. Furthermore we show our ER-ACE objective can be combined with existing methods, leading to additional gains. Finally, we take a closer look at the computation cost of various existing methods. We show that some methods, while obtaining good performance under standard evaluation protocols, fail to meet the computational constraints required in online CL. We provide an extensive evaluation of computational and memory costs across several baselines and metrics.

To summarize, our contributions are as follows. We first highlight the problem of representation drift in the online continual learning setting. We identify a root cause of this issue through an extensive empirical analysis (Sec. 3.4.2). Second, we propose a new family of methods addressing this issue by treating incoming and past data asymmetrically (Sec. 3.4.1, 3.4.3) . Finally, we show strong gains over replay baselines in a new evaluation framework designed to monitor real world constraints (Sec. 6.4). To the best or our knowledge, we are the first to report the computation costs of different methods in our setting, revealing new insights.

3.2 Related Work

Research on continual learning can be divided based on the sequential setting being targeted (see [77–80] for categorizations of the settings and [73] for a broad survey on continual learning). Earlier works consider the relaxed setting of task incremental learning [13, 70, 81] where the data stream is divided into chunks of tasks and each task is learned offline with multiple iterations over the data of this task. While this setting is easier to handle as one task can be learned entirely, it limits the applicability of the solution.

In this work, we consider the challenging setting of an online stream of non-i.i.d. data where changes can anytime occur in the input domain or in the output space. This more realistic setting has attracted increasing interest lately [48,60]. Specifically, we study the *single-head* (or shared head) setting, where when queried, the learner is not told which task the sample belongs to (as opposed to the *multi-head* setting). The single-head assumption is further studied in task-agnostic continual learning settings [41, 82–85] in which the task-boundary assumption, amongst others, is also relaxed. Many of the solutions to the online continual learning problem rely on the use of a

buffer formed of previous memories which are replayed alongside new data during the learning process. Several works [61, 74, 75] propose solutions to select which samples should be stored, or retrieved for replay [48], or both [86]. [40,60] use replay to perform constrained optimization, limiting interference with previous tasks as new ones are learned. Our work, on the other hand, focuses on the appropriate loss function in this context. [87] propose a graph-based approach that capture pairwise similarities between samples. Dark Experience Replay (DER) [88] suggests an alternative replay loss. Samples are stored along with their predicted logits and once replayed the current model is asked to keep its output close to the previously recorded logits. While the method is simple and effective it is worth noting that it relies heavily on data augmentation. Our work is orthogonal and can be combined with DER as we show in Sec. 3.5.6. Finally, concurrent work [89] also use a contrastive loss for online continual learning, but not in an asymmetric fashion.

In our work we also investigate the underlying causes for performance degradation in replay-based methods. Related to this study are works in the class incremental setting, where similar to our case a shared output layer is used, *but classes are learned offline*. Works in this area address the implicit class imbalance issue occurring when new classes are learned alongside replayed data. [90] proposes to correct last layer weights after a group of classes is learned via adjusting the weights norm. [91] suggests to deploy extra additional parameters in order to linearly correct the "bias" in the shared output layer. Those parameters are learned at the end of each training phase. [44] considers addressing this imbalance through applying cosine similarity based loss as opposed to the typical cross entropy loss along with a distillation loss and a margin based loss with negatives mining to preserve the feature of previous classes. Recently, [76] propose to learn the incoming tasks and the previous tasks separately. They use a masked softmax loss for the incoming and rehearsal data, to counter the class imbalance. All the methods highlighted above operate in the *offline* setting, where data from the current task can be revisited as needed making the disruptive issues emphasized at the task boundary less critical. In this paper, we focus on the online setting, with potentially overlapping tasks. As we will see, work by [76] developed to counter class imbalance, can inhibit learning of the current task in the online setting (see Appendix 3.5.4). Lastly, [77] uses a logit masking related to our method but their context is based on the multi-head setting, and does not consider replay based methods, where learning across tasks occurs. Their goal is to activate only the head of which the samples within the new batch belong to. However, our approach is more general and it applies to the single head setting (where we have a

single output layer for all classes, and no task oracle.)

3.3 Learning Setting and Notation

We consider the setting where a learner is faced with a possibly never-ending stream of data. At every time step, a labelled set of examples $(\mathbf{X}^{in}, \mathbf{Y}^{in})$ drawn from a distribution D_t is received. However, the distribution D_t itself is sampled at each timestep and can suddenly change to D_{t+1}, when a task switch occurs. The learner is not explicitly told when a task switch happens, nor can it leverage a task identifier during training or evaluation. We note that this definition generalizes task-incremental learning, where each task is seen one after the other. In this scenario, given T tasks to learn, D_t changes $T-1$ times over the full steam, yielding T locally i.i.d learning phases. We also explore in this paper a more general setting without the notion of clearly delineated tasks [92, 93], where the data distribution gradually changes over time.

Given a model $f_\theta(x)$ representing a neural network architecture with parameters θ, we want to minimize the classification loss $\mathcal{L}$ on the newly arriving data batch while not negatively interfering with the previously learned classes (i.e. increasing the classification loss). A simple and efficient approach to achieve this is to replay stored samples from a fixed size memory, $\mathcal{M}$, in conjunction with the incoming data [75,94]. The core of our approach is that instead of treating the replayed batch and the incoming one similarly and naively minimizing the same loss, *we opt for a specific loss structure on the incoming batch that would limit the interference with the previously well learned classes.* We approach this by allowing the features of the newly received classes in the incoming data to be initially learned in isolation of the older classes. We first present our idea based on a metric learning loss and then generalize to the widely deployed cross-entropy loss.

3.4 Methods

3.4.1 A Distance Metric Learning Approach for Reducing Drift

In order to allow fine-grained control of which samples will be pushed away from other samples given an incoming batch, we propose to apply, on the incoming data, a metric learning based loss from [35]. Related loss functions have recently been popularized

in the self-supervised learning literature [29]. We combine this in a holistic way with a cross-entropy type loss on the replay data. This allows us to control the representation drift of old classes while maintaining strong classification performance. Note that if a metric learning loss is used alone we need to perform predictions using a Nearest Class Means [42] approach, which we show is computationally expensive in the online setting.

Given an input data point x, we consider the function $f_\theta(x)$ mapping x to its hidden representation before the final linear projection. We denote the incoming N datapoints by $\mathbf{X}^{in}$ and data replayed from the buffer by $\mathbf{X}^{bf}$. We use the following loss, denoted SupCon [35], on the incoming data $\mathbf{X}^{in}$.

$$\mathcal{L}_1(\mathbf{X}^{in}) = -\sum_{\mathbf{x}_i \in \mathbf{X}_{in}} \frac{1}{|P(\mathbf{x}_i)|} \sum_{\mathbf{x}_p \in P(\mathbf{x}_i)} \log \frac{\texttt{sim}(f_\theta(\mathbf{x}_p), f_\theta(\mathbf{x}_i))}{\sum_{\mathbf{x}_n \in N \cup P(\mathbf{x}_i)} \texttt{sim}(f_\theta(\mathbf{x}_n), f_\theta(\mathbf{x}_i))} \tag{13}$$

where $\texttt{sim}(a, b) = \exp(\frac{a^T b}{\tau \|a\| \|b\|})$ computes the exponential cosine similarity between two vectors, with scaling factor τ [31,95]. Here we denote the incoming data $\mathbf{x}_i \in \mathbf{X}^{in}$. We use the P and N to denote the set of positives and negatives with respect to $\mathbf{x}_i$ and the positive examples x_p are selected from the examples in $\mathbf{X}^{in} \cup \mathcal{M}$, which are from the same classes as $\mathbf{x}_i$. In the sequel, we will consider $\mathbf{x}_n$ selected from $\mathbf{X}^{in} \cup \mathcal{M}$ in two distinct ways: (a) from a mix of current and previous classes and (b) only from classes of the $\mathbf{X}^{in}$. Note that this implicitly learns a distance metric where samples of the same class lie close by. For the rehearsal step, we apply a modified cross-entropy objective as per [95] which allows us to link the similarity metric from above to the logits.

$$\mathcal{L}_2(\mathbf{X}^{bf}) = -\sum_{x \in \mathbf{X}_{bf}} \log \frac{\texttt{sim}(\mathbf{w}_{c(x)}, f_\theta(\mathbf{x}))}{\sum_{c \in C_{all}} \texttt{sim}(\mathbf{w}_c, f_\theta(\mathbf{x}))} \tag{14}$$

where C_{all} the set of all classes observed, and $c(x)$ denotes the label of x. The above formulation allows us to interpret the rows of the final projection $\{\mathbf{w}_c\}_{c \in C_{all}}$ as class prototypes and inference to be performed without the need for the nearest neighbor search. We combine the loss functions on the incoming and replay data

$$\mathcal{L}(\mathbf{X}^{in} \cup \mathbf{X}^{bf}) = \gamma \mathcal{L}_1(\mathbf{X}^{in}) + \mathcal{L}_2(\mathbf{X}^{bf}) \tag{15}$$

We refer to this approach as Experience Replay with Asymmetric Metric Learning (ER-AML). Note the buffer may contain samples with the same classes as the incoming data stream. The subroutine `FetchPosNeg` is used to find one positive

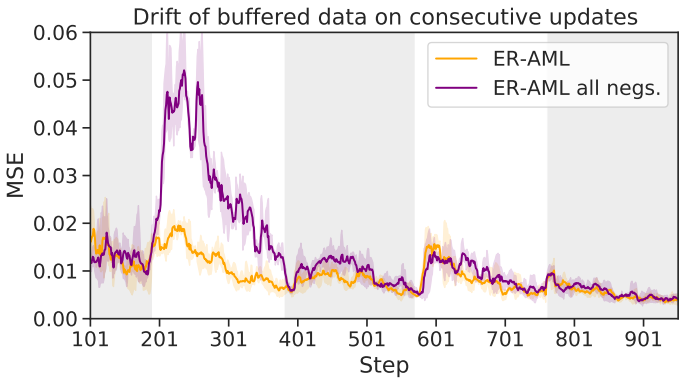


Figure 8: Buffer displacement in a 5 task stream. Background shading denotes different tasks.

and negative sample per incoming datapoint in $\mathbf{X}^{in}$, which can reside in either the buffer memory $\mathcal{M}$ or in $\mathbf{X}^{in}$.

3.4.2 Negative Selection Affects Representation Drift

The selection of negatives for the proposed loss $\mathcal{L}$ can heavily influence the representation of previously learned classes and is analogous to the key issues faced in the regular replay methods where cross-entropy loss is applied to both incoming and replay data. A typical approach in this loss for classification may be to select the negatives from any other class [96]. However, this becomes problematic in the continual learning setting as the old samples will be too heavily influenced by the poorly embedded new samples that lie close to the old sample representations. To illustrate what is going on in the feature space, consider the case of a ER-AML's $\mathcal{L}_1$ term, which explicitly controls distances between sample representations. $\mathcal{L}_1$ considers the incoming batch samples (containing new classes) as anchors. As the representations from these classes haven't been learned, anchors may end up placed near or in-between points from previous classes (analogous to the illustration in Figure 7). Since the previous classes samples will be clustered together, if we use them as negatives for the incoming sample anchors, the gradients magnitude of the positive term will be out-weighted by the negative terms coming from the new class samples, similar to what is observed in Figure 7. In this case, there is a sharp change in gradients norms of the loss w.r.t. the features of previous classes, as we illustrate in Appendix 3.5.7, which leads to a large change in the representation at the task boundary (and subsequently poor performance). On the other hand, if we use only incoming batch examples as negatives we can avoid this excessive representation

drift. We illustrate this in Figure 8 by showing the representations drift at the task boundaries for ER-AML when using negative samples from all classes and when using only classes in the incoming batch. In the context of the model under consideration we measure the one iteration representation drift of a sample x as $\|f_{\theta^t}(x) - f_{\theta^{t+1}}(x)\|$, the output of the network being normalized. We observe that naively applying the proposed loss results in large changes of the learned representation. On the other hand, when allowing only negatives from classes in the incoming batch, we see a reduction in this representation drift. In the Appendix 7 we further demonstrate that the accuracy of models trained using ER-AML with only incoming batch negatives can improve the continual learning system performance by a large margin. We emphasize that ER-AML with all negatives and the regular ER method used for online continual learning suffer from a similar issue and thus lead to similar poor performances, with appropriate negative selection resolving the problem. This is further emphasized in Appendix 3.5.10 where we observe similar poor drift behavior for ER.

3.4.3 Cross-entropy Based Alternative

Having demonstrated the effect of controlling the incoming batch loss in avoiding a drastic representation drift, we now extend it to be applicable to the standard cross-entropy loss typically studied in ER [48, 75]. Given an incoming data batch, consider C_{old} the set of previously learned classes and C_{curr} the set of classes observed in the current incoming mini-batch. Denoting C the set of classes included in the cross-entropy loss, we define the $\mathcal{L}_{ce}(\mathbf{X}, C)$ cross-entropy loss as: $\mathcal{L}_{ce}(\mathbf{X}, C) = -\sum_{x \in \mathbf{X}} \log \frac{\texttt{sim}(w_{c(x)}, f_\theta(x))}{\sum_{c \in C} \texttt{sim}(w_c, f_\theta(x))}$ where $C \subset C_{all}$ denotes the classes used to compute the denominator. We note that restricting the classes used in the denominator has an analogous effect to restricting the negatives in the contrastive loss. Consider the gradient for a single datapoint x, $\frac{\partial \mathcal{L}_{ce}(x,C)}{\partial f_\theta^n} = \mathbf{W}((\vec{p} - \vec{y}) \odot 1_{\vec{y} \in C})$. Here $\vec{p}$ denotes the softmax output of the network, $\vec{y}$ a one-hot target, $1_{\vec{y} \in C}$ a binary vector masking out classes not in C, and $\mathbf{W}$ the matrix with all class prototypes $\{\mathbf{w}_c\}_{c \in C_{all}}$. When the loss is applied in the batch setting, it follows that only prototypes whose labels are in C will serve roles analogous to positives and negatives in the contrastive loss. We can then achieve a similar control as the metric learning approach on the learned representations.

Now, our loss applied at each step would be:

$$\mathcal{L}_{ace}(\mathbf{X}^{bf} \cup \mathbf{X}^{in}) = \mathcal{L}_{ce}(\mathbf{X}^{bf},\ C_{old} \cup C_{curr}) + \mathcal{L}_{ce}(\mathbf{X}^{in}, C_{curr})$$

where C_{curr} denotes the set of the classes represented in the incoming batch and C_{old} denotes previously seen classes that are not presented in the incoming batch, those that we want to preserve their representation. Note this is a straightforward procedure and induces no additional computational overhead. We refer to it as Experience Replay with Asymmetric Cross-Entropy (ER-ACE).

3.5 Experiments

We have highlighted the issue of abrupt representation change when new classes are introduced, and propose two methods that address this issue. We now demonstrate that mitigating drift directly leads to better performance on standard online continual learning benchmarks. As in [48,60,75] we use a reduced Resnet-18 for our experiments, and leave the *batch size* and the *rehearsal batch size* fixed at *10*. This allows us to fairly compare different approaches, as these parameters have a direct impact on the computational cost of processing a given stream.

Datasets All benchmarks are evaluated in the single-head setting, i.e. task descriptors are not provided to the model at test time, hence the model performs N-way classification where N is the total amount of classes seen.

Split CIFAR-10 partitions the dataset into 5 disjoint tasks containing two classes each (as in [48, 97])

Split CIFAR-100 comprises 20 tasks, each containing a disjoint set of 5 labels. We follow the split in [75]. All CIFAR experiments process 32×32 images.

Split MiniImagenet splits the MiniImagenet dataset into 20 disjoint tasks of 5 labels each. Images are 84×84.

Baselines We focus our evaluation on replay-based methods, as they have been shown to outperform other approaches in the online continual learning setting [48, 75, 98]. We keep buffer management constant across methods : all samples are kept or discarded according to Reservoir Sampling [99]. We consider the following state-of-the-art baselines:

ER: Experience Replay with a buffer of a fixed size. Unlike [48], we do not leverage the task identifier during training to ensure that rehearsal samples belong to previous classes.

iCaRL [42] A distillation loss alongside binary cross-entropy is used during training. Samples are classified based on closest class prototypes, obtained from

recomputing and averaging buffered data representations.

MIR [48] selects for replay samples interfering the most with the incoming data batch.

DER++ [88] uses a distillation loss on the logits to ensure consistency over time.

SS-IL [76] learns both the current task loss and the replay loss in isolation of each other. An additional task-specific distillation is used on the rehearsal data.

GDUMB [100] performs offline training on the buffer with unlimited computation and unrestricted use of data augmentation at the end of the task sequence.

iid: The learner is trained with a single pass on the data, in a single task containing all the classes. We also consider a version of this baseline using a similar compute budget as replay methods (**iid++**)

We note additional baselines such as [40, 60] were shown to perform poorly in this setting by prior work [88] and are thus left out for clarity.

3.5.1 Evaluation Metrics and Considerations

Our evaluation includes the metrics and experimental settings used in previous works on online continual learning with a single-head [48,97,98]. We provide extra emphasis on anytime evaluation and comparisons of the computation time per incoming batch. We also consider several additional settings in terms of computation and use of image priors.

Anytime evaluation A critical component of online learning is the *ability to use the learner at any point* [73] . Although most works in the online (one-pass through the data) setting report results throughout the stream [40, 60, 61], several prior works have reported the final accuracy as a proxy [48, 97]. However a lack of anytime evaluation opens the possibility to exploit the metrics by proposing offline learning baselines that are inherently incompatible with anytime evaluation [100].
In order to make sure that learners are indeed online learners, we evaluate them throughout the stream. We define the Anytime Accuracy at time k (AA_k) as the average accuracy on the test sets of all distributions seen up to time k. If the learning experience lasts T steps, then AA_T is equivalent to the final accuracy. Finally, we report the *Averaged Anytime Accuracy (AAA)* [83], which measures how well the model performed over the learning experience

$$\text{AAA} = \frac{1}{T}\sum_{t=1}^{T}(AA)_t. \tag{16}$$

Computation and Memory Constraints While memory constraints are well documented in previous work, careful monitoring of computation is often overlooked; some methods can indeed hide considerable overhead which can make the comparison across methods unfair. On the other hand this is critical to the use cases of online continual learning. To remedy this, we report for each method the total number of FLOPs used for training. While we cannot fix this quantity as we can for memory (since different methods require different computations), this will shed some light on how different methods compare. Note that we also include in this total any inference overhead required by the models; Nearest Class Mean (NCM) classifiers must compute class prototypes before inference for example. We add this cost every time the model is queried to measure its Anytime Accuracy. Let

$$\text{Mem} = \frac{1}{T}\sum_{t=1}^{T}|\theta_t|+|\mathcal{M}_t|, \quad \text{Comp} = \sum_{t=1}^{T}\mathcal{O}(m(\cdot;\theta_t)), \tag{17}$$

where $\mathcal{O}(m(\cdot;\theta_t))$ denotes the number of FLOPs used at time t. Since the same backbone and buffer is used for all methods in this paper, we will focus our constraint analysis on computation

Data Augmentation In the settings of [48, 60, 75, 97, 98] data augmentation is not used. However, this is a standard practice for improving the performance on small datasets and can thus naturally complement most methods utilizing replay buffers. Notably, [100], the offline learning method, utilizes data augmentation when comparing to the above online learners. To avoid unfair comparisons, in our experiments we indicate when a method uses augmentation. When not specified, we treat it as a hyperparameter and report the best performance.

Hyperparameter selection For all datasets considered, we withhold 5 % of the training data for validation. For each method, optimal hyperparameters were selected via a grid search performed on a validation set. The selection process was done on a per dataset basis, that is we picked the configuration which maximized the accuracy averaged over different memory settings. We found that for both ER-AML and ER-ACE, the same hyperparameter configuration worked across all settings and datasets. All necessary details to reproduce our experiments can be found in the Appendix.

Method	Data Aug.	$M = 5$ AAA	$M = 5$ Acc	$M = 20$ AAA	$M = 20$ Acc	$M = 100$ AAA	$M = 100$ Acc	Train TFLOPs	Mem. (Mb)
iid	✗	-	62.7±0.7	-	62.7±0.7	-	62.7±0.7	8	4
iid++	✗	-	72.9±0.7	-	72.9±0.7	-	72.9±0.7	16	4
DER++	✓	50.7±1.1	31.8±0.9	55.6±1.2	39.3±1.0	60.1±1.3	52.3±1.1	24	**(4, 7)**
ER	✗	40.0±0.8	19.7±0.3	45.2±1.3	26.7±1.0	55.4±1.4	38.7±0.8	**17**	**(4, 7)**
	✓	45.6±1.1	28.4±1.0	55.9±1.2	40.3±0.6	60.3±1.3	49.4±1.3		
iCaRL†	✗	47.0±0.8	30.6±0.8	55.1±0.7	41.7±0.6	59.3±0.6	45.1±0.6	(21, 47)	(8, 11)
	✓	49.1±1.0	33.4±1.0	54.4±0.7	39.2±0.8	56.9±0.7	42.3±0.8		
MIR†	✗	39.3±1.0	19.7±0.5	44.7±1.1	29.7±0.6	53.8±1.7	43.3±1.0	41	**(4, 7)**
	✓	44.9±0.9	29.8±0.8	49.7±1.0	41.8±0.6	54.6±1.4	49.3±0.6		
SS-IL†	✗	42.6±1.7	29.6±0.4	44.8±1.8	35.1±0.9	48.1±2.2	41.1±0.4	19	(8, 11)
	✓	41.1±1.6	31.6±0.5	47.0±1.2	38.3±0.4	48.1±1.7	47.5±0.7		
ER-ACE (ours)	✗	**53.1**±1.0	**35.6**±1.0	**58.0**±0.7	42.6±0.7	**61.9**±0.9	52.2±0.7	**17**	**(4, 7)**
	✓	**52.6**±0.9	**35.1**±0.8	**56.4**±1.0	43.4±1.6	**61.7**±0.9	**53.7**±1.1		
ER-AML (ours)	✗	49.4±1.0	30.9±0.8	**57.0**±1.0	39.2±1.0	**63.3**±1.0	52.2±1.1	**17**	**(4, 7)**
	✓	50.4±1.3	**36.4**±1.4	**56.8**±1.0	**47.7**±0.7	**62.0**±0.9	**55.7**±1.3		
GDUMB	✓	0±0.0	35.0±0.6	0±0.0	45.8±0.9	0±0.0	61.3±1.7	(43, 853)	(11, 14)

Table 1: split CIFAR-10 results. † indicates the method is leveraging a task identifier at training time. For methods whose compute depend on the buffer size, we report min and max values. We evaluate the models every 10 updates. Results within error margin of the best result are bolded.

Method	AAA	Acc.	Train TFLOPs	Mem. (Mb.)	AAA	Acc.	Train TFLOPs	Mem. (Mb.)
iid	-	19.8±0.3	9	4	-	16.7±0.5	59	4
iid++	-	28.3±0.3	17	4	-	25.0±0.8	118	4
DER++	23.3±0.5	15.1±0.4	25	36	21.7±0.6	12.9±0.3	176	217
ER	24.2±0.6	19.8±0.4	**17**	**35**	26.2±0.8	18.2±0.5	**118**	**216**
iCaRL†	26.3±0.3	17.3±0.2	294	39	24.4±0.4	17.1±0.1	2097	220
MIR†	23.6±0.8	20.6±0.5	41	**35**	27.2±0.7	20.2±0.8	294	**216**
SS-IL†	31.5±0.5	25.0±0.3	19	39	**29.7**±0.6	**23.5**±0.5	137	220
ER-ACE (ours)	**32.7**±0.5	**25.8**±0.4	**17**	**35**	**30.2**±0.6	**22.7**±0.6	**118**	**216**
ER-AML (ours)	30.2±0.6	24.3±0.4	28	**35**	27.0±0.7	19.3±0.6	200	**216**

Table 2: Split CIFAR-100 (left) and Mini-Imagenet (right) results with $M = 100$. For each method, we report the best result between using (or not) data augmentations.

3.5.2 Standard Online Continual Learning Settings

We evaluate on Split CIFAR-10, Split CIFAR-100 and Split MiniImagenet using the protocol and constraints from [48, 97, 98] . We note in all results each method is run 10 times, and we report the mean and standard error. We first discuss dataset specific results, before analysing the computation cost of each method.

CIFAR-10 results are found in Table 1 using a variety of buffer sizes. In this setting, we see that both the methods we propose, ER-AML and ER-ACE *consistently outperform* other methods by a significant margin. This result holds in both settings

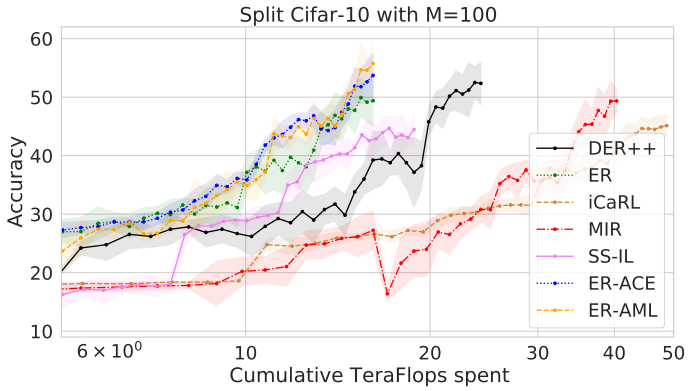


Figure 9: Total Accuracy as a function of TeraFLOPs spent. Here the models are evaluated on all 10 classes, to ensure consistency across timesteps.

where data augmentation is (or not) used, outperforming previous state-of-the-art methods MIR and DER++. Shifting our attention to SS-IL, its underperformance w.r.t to ER-ACE highlights the importance of having a rehearsal objective that considers the new classes. In Appendix 3.5.4, we observe that when applying SS-IL in the online setting: (1) the method performs poorly on the current task, as is it unable to consolidate old and new knowledge, (2) yet mitigates representation drift even on a perfectly balanced stream. The latter is surprising, as the method was designed specifically to address stream imbalance. Finally, we note the offline training baseline G-DUMB cannot satisfy the anytime evaluation criteria.

textbfLonger Task Sequence results are shown in Table 2 with CIFAR-100 on the left and MiniImagenet on the right. On both datasets similar findings are observed, our proposed methods match or outperform strong existing baselines. SS-IL performs similarly to our method on mini-imagenet hile having a higher computational and memory cost. As mentioned above, the method struggles to learn the current task, however here the “weight" of the current task is small in the final acc of the 20-task regime. Finally, ER-ACE shows relative gains of **35%** in accuracy over ER, without any additional computation cost. For Mini-Imagenet, ER-ACE outperforms the single-pass iid baseline, and nearly reaches the performance of the equal-compute iid baseline.

Computation Budget To provide another view of the computational advantages of our proposal we report the accuracy given compute budget over the length of the sequence in Fig 9. When monitoring the computation performed by each baseline, we notice that several methods do not compete on equal footing. First, the use of Nearest Class Mean (NCM) classifiers leads to a significant compute cost, as shown

Method	$M = 20$	$M = 100$
ER	$32.1_{\pm 1.5}$	$42.7_{\pm 2.2}$
DER++	$31.0_{\pm 1.4}$	$41.7_{\pm 1.4}$
ER-AML	$\mathbf{45.6}_{\pm 1.2}$	$\mathbf{55.2}_{\pm 1.1}$
ER-ACE	$\mathbf{44.5}_{\pm 0.5}$	$50.2_{\pm 1.1}$

Table 3: CIFAR-10 Blurry Task Boundary Experiments

for iCaRL. For our experiments, we evaluate the model after 10 mini-batches (100 total samples), where NCM classifier **must forward the whole buffer** to get class prototypes. We argue that such an approach has disadvantages in the online setting due to poor computational trade-offs. Second, MIR [48] has an expensive sample retrieval cost. It remains to show if this step can be approximated more efficiently. Finally, we note that our method, ER-AML has varying compute: for streams with a small number of classes per task (CIFAR10), it can compute the incoming loss leveraging only the incoming data. In other datasets, where an incoming batch may not have at least two samples of each class, an additional cost to forward a buffered point is incurred.

Evaluation with augmentation The use of augmentations also permits extra benefits of replay methods particularly in settings where buffer overfitting is more present, e.g. in the small buffer regime. From the results in Table 1, we see that *augmentations provides significant gains* for a large set of methods. It is therefore crucial to compare methods on equal footing, where they can all leverage (or not) data augmentation. For example, *gains reported in* [100] *over ER completely vanish* when ER is given the same access to augmented data. We note that for Mini-Imagenet, augmentations did not help. We hypothesize that since this is the hardest task the risk of overfitting on the buffer is less severe.

3.5.3 Blurry Task Boundaries

Next, we explore a setting where the distribution is continuously evolving, rather than clearly delineated by task boundaries (similar to settings considered in [61]). To do this, we linearly interpolate between tasks over time, resulting in new classes being slowly mixed into the data stream. This experiment is done on Split-CIFAR10, and the interpolation is such that at every timestep, the incoming data batch has

on average 2 unique labels (as in the original experiment). We only evaluate task-free methods in this setting: methods like *MIR and SS-IL cannot be used in such setting*. Results in table 3 report the final accuracy, averaged over 5 runs, we report the standard error. We observe our ER-AML and ER-ACE methods perform well in this setting.

3.5.4 An in-depth analysis of SS-IL in the online setting

SS-IL is a related method. In this section, we highlight several key observations when deploying SS-IL in the online setting which are on the other hand not issues for ER-AML and ER-ACE. We then provide several additional experiments, shedding some light on the inner workings of the method.

SS-IL fails to learn the current task As stated earlier, the key difference between SS-IL without distillation and ER-ACE is that in the latter, *the rehearsal loss in unmasked.* In this section, we highlight the problems that occur when using a masked rehearsal loss alongside a masked incoming loss as in SS-IL. We show that since both losses are masked, *the model never learns to classify classses across tasks.* Specifically, there is no objective in which the model learns to distinguish classes in the current task from classes in the previous tasks. As we show in Figure 10, SS-IL is unable to classify samples from the current task in a single-head setting. The method actually performs worse than random chance on samples form the current task. On the other extreme we see that ER does very well on the current task (shifting abrupty the previous representations to accomodate the new task). Finally, we see that ER-ACE strikes a good tradeoff between the two, reaching a reasonable accuracy on the current task without disrupting the learned representations of previous tasks. We note that the same conclusion is reached when using the original SS-IL method with the distillation loss.

SS-IL does more than correcting for class imbalance SS-IL is motivated as a method which addresses the class imbalance issue arising in replay methods. Specifically, when drawing a fixed number of rehearsal points at every epoch, it follows that as more and more tasks are seen, previous classes are underrepresented in the training stream when compared to points from the current tasks.

In this section, we test whether or not the behavior of SS-IL differs from standard Experience Replay *when no class imbalance is present.* In this experiment, we increase the number of rehearsal points sampled at every task such that when combining incoming and rehearsal data, we obtain perfectly balanced training data on average.

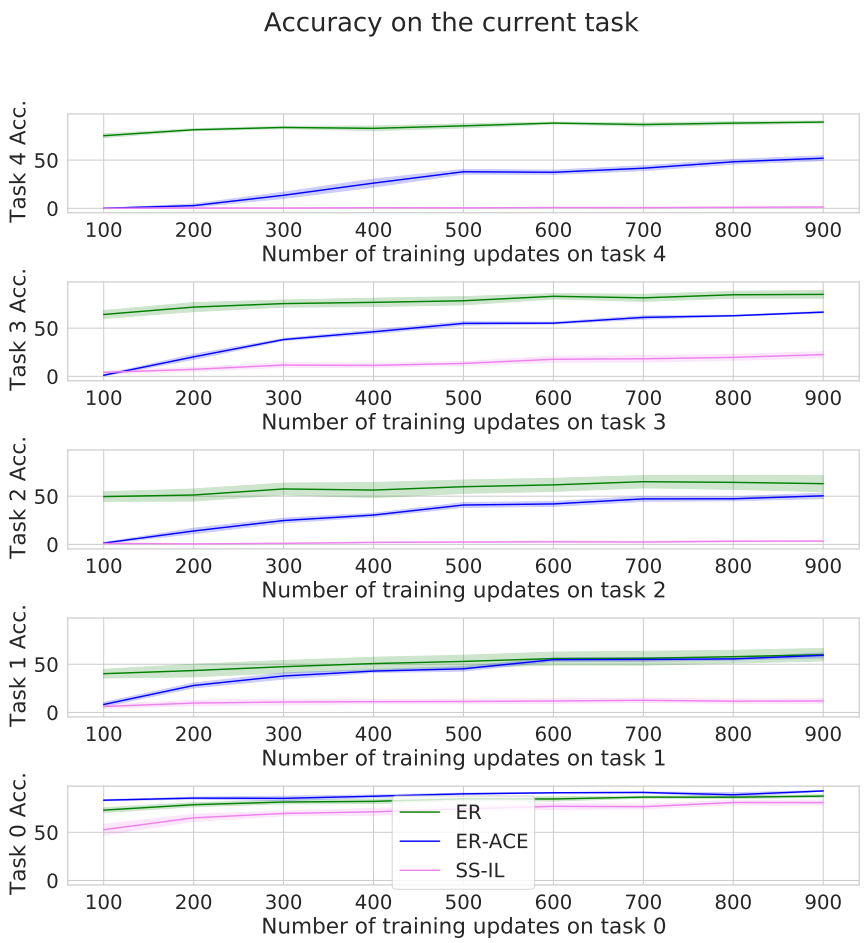


Figure 10: For Split-CIFAR-10, we monitor the performance on the current task observed in the stream for SS-IL, ER, and ER-ACE. ER fits too abruptly current task; ER-ACE incorporates this knowledge slowly; SS-IL barely on the other hand is unable to learn new tasks when they are first observed in the stream

This is experiment is done on the Split-CIFAR10 benchmark with 2 classes per task, with a minibatch of 10 incoming datapoints. Therefore, we sample $0, 10, 20, 30, 40$ rehearsal points per incoming databatch for the first, second, third, fourth and fifth task.

What we observe is that SS-IL still outperforms regular Experience Replay, suggesting that the method does more than simply addressing class imbalance in the data stream. We report final accuracy in Table 4. SS-IL's performance gap with ER is bigger with small buffer. This is consistent with what we observe for representation drift : methods with larger buffer can better correct for abrupt representation change, making the gap between ER vs ER-ACE and ER-AML smaller. From this we give new insights on the inner workings of SS-IL, namely that it works well because it

Method	$M = 20$	$M = 50$	$M = 100$
ER	21.0 ± 1.2	25.7 ± 1.1	37.8 ± 0.7
SS-IL	30.3 ± 1.0	34.6 ± 0.8	39.1 ± 0.6

Table 4: Final Accuracy on split CIFAR-10 with class balanced stream.

addresses representation drift rather than class imbalance.

3.5.5 Overfitting on buffered samples

We study the extent to which our proposed method reduces over-fitting to samples stored in the buffer.

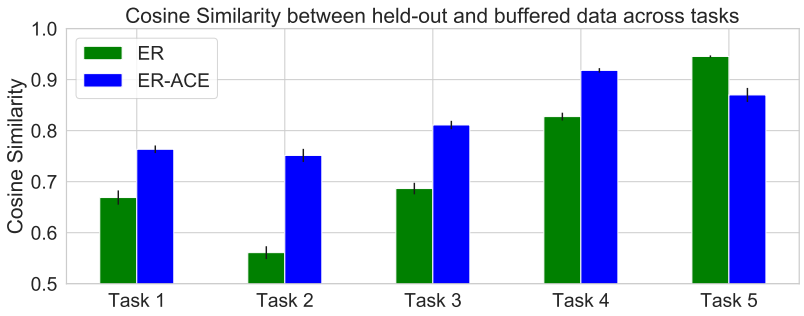


Figure 11: Alignment between buffer and holdout representations. ER-ACE has constantly larger alignment between seen and unseen samples compared to ER especially for older tasks.

A good model fit should yield a learned representation where same class datapoints are aligned, *whether or not they were seen during training.* To evaluate this potential mismatch, we first train a model and compare the representations of a) samples in the buffer M after training and b) held-out samples from the validation set V. That is, for each datapoint $x_m \in M$ we find the point $x_v \in V$ with $c(x_m) = c(x_v)$ which maximizes the cosine similarity between $f_\theta(x_m)$ and $f_\theta(x_v)$. This allows to compare alignment across models, irrespective of their internal scaling. We report the results in Figure 11, where similarity values are averaged over points from the same task. We find that our proposed method, ER-ACE, designed to reduce representation drift also reduces the extent to which the model overfits on the buffer. We observe that for earlier tasks, ER-ACE still retains a strong alignment between rehearsal and held-out data, which is not the case for ER.

3.5.6 Combining ER-ACE with DER++

In this section, we apply our method on top of the strong DER++ [88] baseline. For this experiment, we use the same setting as in the DER paper. Specifically, we port our implementation to their public codebase `https://github.com/aimagelab/mammoth`. We keep the default settings for CIFAR-10, using a single pass through the data. We find that combining ER-ACE with DER++ yields additional advantages. Not only do we observe small gains in accuracy, we notice significant gains in forgetting. Results are shown in Figure 12. Forgetting is defined as in [40].

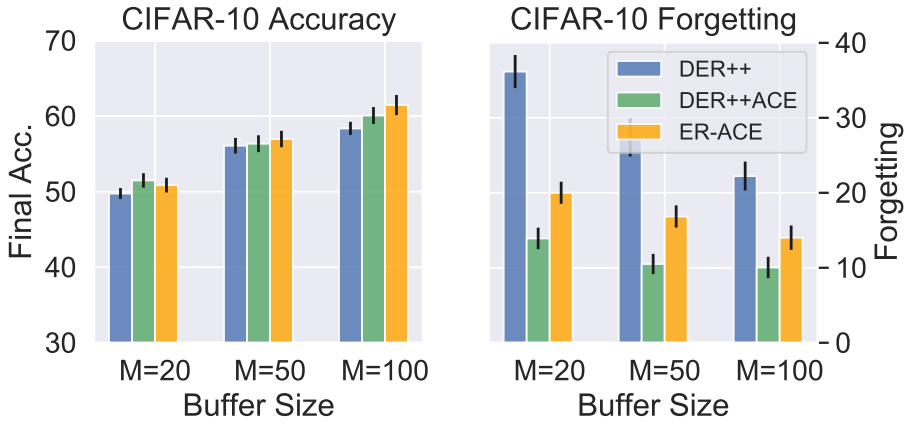


Figure 12: Comparison to Dark Experience Replay (DER). We obtain improved performance and we can enhance the DER method using the ER-ACE approach

3.5.7 Gradient Norm

Figure 13 shows the gradients norms of the features of previous classes in a stream of two tasks. Note how for normal ER, at the task switch the gradients of the previous classes features are suddenly very high leading potentially to large drift on these features.

ER	$(3.2 \pm 1.8) \times 10^{-2}$
ER-AML-Triplet w. All Negs	$(3.0 \pm 0.6) \times 10^{-2}$
ER-AML-Triplet w. Incoming Negs	$(2.5 \pm 0.6) \times 10^{-2}$

Table 5: Average Drift (avg distance in feature space) of buffered representations for CIFAR-10 during learning of the second task. We observe similar behavior to ER-AML with SupCon

	Accuracy ↑			
	$M = 5$	$M = 20$	$M = 50$	$M = 100$
iid online	60.8 ± 1.0	60.8 ± 1.0	60.8 ± 1.0	60.8 ± 1.0
iid++ online	72.0 ± 0.1	72.0 ± 0.1	72.0 ± 0.1	72.0 ± 0.1
iid offline	79.2 ± 0.4	79.2 ± 0.4	79.2 ± 0.4	79.2 ± 0.4
fine-tuning	18.4 ± 0.3	18.4 ± 0.3	18.4 ± 0.3	18.4 ± 0.3
ER	19.0 ± 0.1	26.7 ± 0.3	36.1 ± 0.6	41.5 ± 0.6
ER-AML Triplet	33.0 ± 0.3	40.1 ± 0.4	46.0 ± 0.5	49.8 ± 0.5
ER-AML SupCon	33.0 ± 0.2	$\mathbf{41.9 \pm 0.1}$	$\mathbf{48.3 \pm 0.2}$	$\mathbf{51.9 \pm 0.3}$

Table 6: Ablation comparing ER-AML with triplet loss to ER-AML with SupCon. We observe both improve over ER but SupCon has better performance in larger buffer sizes

Figure 13: Gradient's norm for first task features in a two task learning scenario. We observe a sharp increase when all negatives are used and decrease using only incoming negatives.

3.5.8 ER-AML with Triplet loss

We observe similar behavior for ER-AML implemented with the triplet loss in terms of the importance of negative selection on drift as illustrated in Table 5. We also ablate ER-AML based on SupCon and Triplet in Table 6 finding the former outperforms in settings with higher buffer sizes, but that both outperform ER.

3.5.9 Ablations Negative Selection

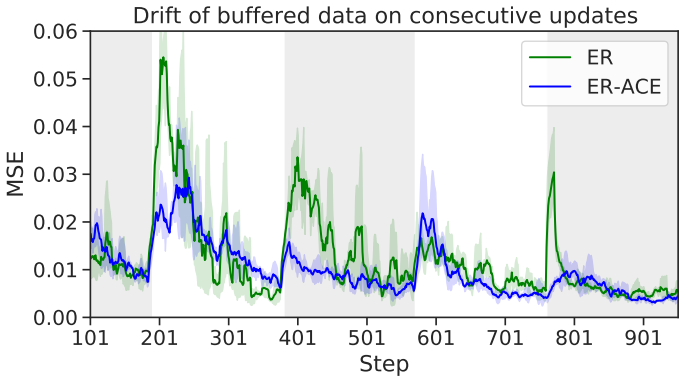
As discussed in the main paper, the selection of negatives is a critical aspect of ER-AML and motivates ER-ACE. To further illustrate this we ablate the performance of ER-AML when all possible negatives are used versus the prescribed negative selection strategy (using only classes in the incoming batch). The results are shown in Table 7. We observe that performance of ER-AML with all negatives is similar to but slightly better than ER, while use of well-selected negatives greatly improves performance.

	Accuracy (↑ is better)		Forgetting (↓ is better)	
	$M = 20$	$M = 50$	$M = 20$	$M = 50$
ER	26.7 ± 0.3	36.1 ± 0.6	47.1 ± 0.8	37.6 ± 0.9
ER-AML(all negatives)	28.5 ± 0.3	41.4 ± 0.4	56.7 ± 0.6	35.0 ± 0.4
ER-AML(incoming negatives)	$\mathbf{41.9 \pm 0.1}$	$\mathbf{48.3 \pm 0.2}$	$\mathbf{33.6 \pm 0.2}$	$\mathbf{25.8 \pm 0.3}$

Table 7: Ablation of ER-AML with all negative selection versus negatives selected from incoming classes. We use the CIFAR-10 dataset. We observe that performance of ER-AML with all negatives is similar to but slightly better than ER, while use of well-selected negatives greatly improves performance.

3.5.10 Additional Drift Results

We showed in Figure 8 that the selection of negatives has a significant impact on the amount of representation change. Here we show that a similar behavior is observed with ER vs ER-ACE.



3.5.11 Analysis of the Representations During the Second Task

In this section we take a closer look at the model's internal representation during the learning of the second task for different methods. This experiment replicates the setup illustrated in Figure 7 (split-CIFAR-10 with $M = 20$). For each method, the figures for all iterations were projected together to ensure that the figures are comparable across timesteps. All methods were initialized starting from the same base model trained on the first task. The dotted representations shown for each class come from held-out samples.

We start by looking at the representations obtained at the begining of the second task. We see that for all three methods, (i) the prototypes of the classes from the first task (Class 0 and Class 1) are well placed, while the other prototypes are placed at random since they are not trained.

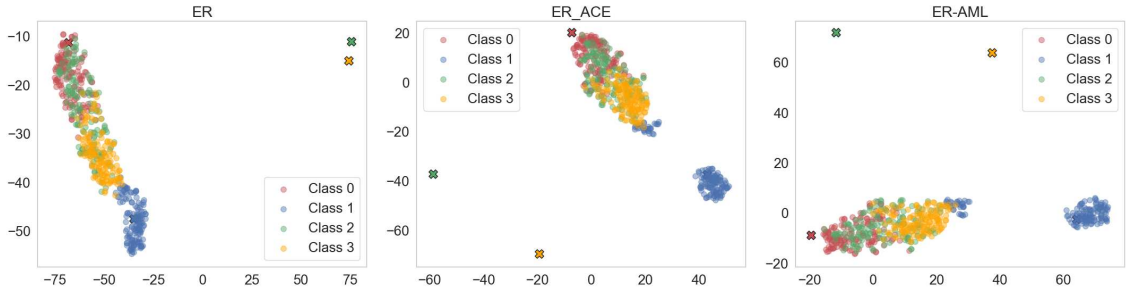


Figure 14: 1 Training Iteration on the Second Task

After 100 training iterations, we see that for ER, the prototypes of the old classes have been significantly displaced and are far from the points of similar class. This is not the case for the latter two methods; for ER-ACE and ER-AML, the model is beginning to separate de classes from one another, and the class prototypes are near their respective classes.

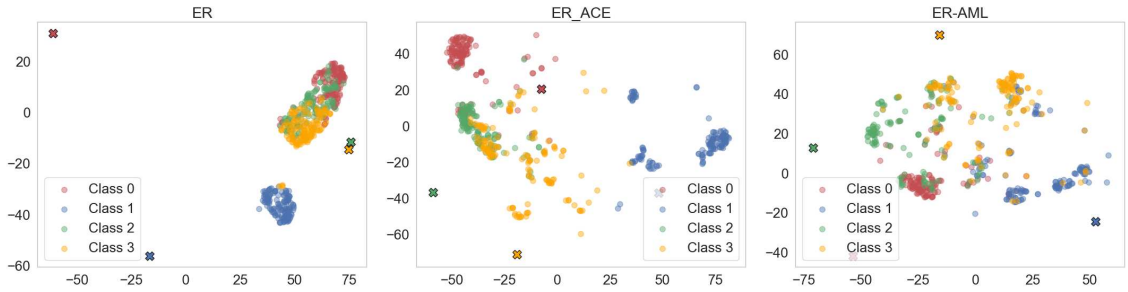


Figure 15: 100 Training Iterations on the Second Task

After 400 training iterations, ER still struggles to align the class prototypes with the respective classes. ER-ACE has already well clustered the respective classes. ER-AML, continues to cluster the classes together, however does not do it as fast as ER-ACE.

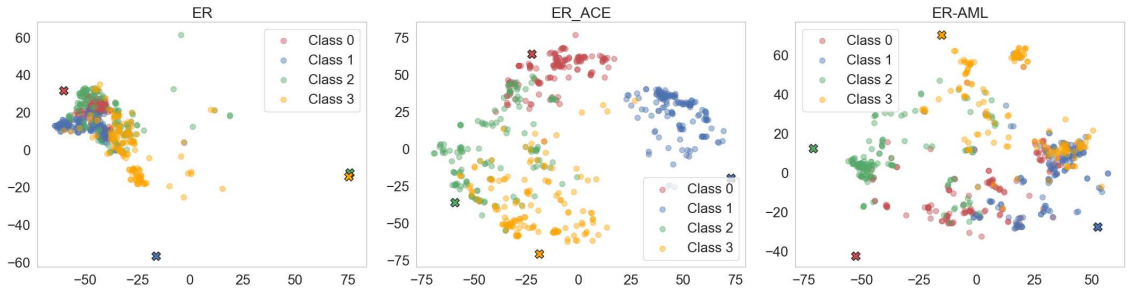


Figure 16: 400 Training Iterations on the Second Task

At the end of the second task, ER-ACE and ER-AML have successfully clustered the classes **and** aligned their respective prototypes with the clusters. As for ER, while the data is clustered, the prototypes are not properly aligned with class clusters. Moreover, we still see a strong overlap between prototypes of Class 2 and 3.

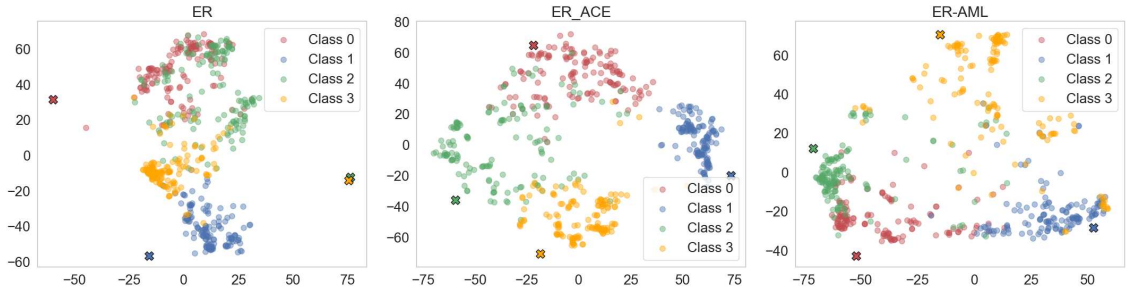


Figure 17: End of the Second Task

3.5.12 Additional Blurry Task Boundaries Experiments

Here we provide blurry task results for varying levels of task overlap. To give an idea of how much the tasks overlap, we report the average number of unique classes per incoming minibatch (MB): a small number means that the tasks are well separated. A high number means that there is a strong overlap. In the fully i.i.d setting, this number would be maximized. On the other hand, when this equals 1, each data class is streamed one after the other.

Experiments are performed again on CIFAR-10 with $M = 20$. We use augmentations to fairly compare with DER++. Results are averaged over 5 runs.

Method	Avg. unique classes per MB				
	1	2	3	4	5
ER	23.1	25.7	26.3	31.1	34.4
DER++	20.3	31.1	31.4	37.3	34.4
ER-ACE	32.8	36.2	36.8	41.7	44.5
ER-AML	**34.0**	**40.4**	**46.0**	**47.6**	**47.9**

We see that through a wide range of different blurriness levels, our methods show strong improvement over other task-free baselines

3.5.13 Experiments with limited training data available

Next, we evaluate the methods above using varying percentages of the training data from the second task onwards (we use all the data for the first task so the model has converged to a reasonable solution before the first distribution shift). Moreover, we augment the rehearsal batch size for ER, ER-ACE and ER-AML to 20, so that their compute cost equals DER++. This is again on CIFAR-10, $M = 20$. Results averaged over 5 runs.

Method	% of Data Used			
	5%	10%	25%	50%
ER	17.3	22.5	28.0	33.2
DER++	17.4	19.9	24.8	32.8
SS-IL	15.2	21.7	28.6	31.9
ER-ACE	**20.5**	**25.4**	**31.2**	36.1
ER-AML	18.1	24.3	31.0	**38.7**

Again, we see that the proposed methods outperforms the baselines suggested above.

3.6 Summary

We have illustrated how in the online continual learning setting the standard loss applies excessive pressure on old class representations. We proposed two modifications of the loss function, both based on treating the incoming and replay data in an asymmetric fashion. Our proposed method does not require knowledge of the current task and is shown to be suitable for long task sequences achieving strong performance with minimal or no additional cost. We also raise the standard for high quality evaluation in online continual learning by considering a wide number of baselines and metrics.

Chapter 4

One-Class Incremental Learning with Regularized Class Prototypes

This work was with the collaboration of Sudhir Mudur and Eugene Belilovsky. It was published as a workshop article in NeurIPS 2022 [101].

4.1 Introduction

Continual learning is a paradigm that aims to allow deep learning algorithms which will have the ability to learn online from a non-stationary and never-ending stream of data. Such systems must become capable of acquiring new knowledge, while avoiding catastrophic forgetting of previously seen data, a problem commonly known and suffered by gradient-based neural networks [102]. A number of common continual learning scenarios exist in the classification setting, each with their own set of related but also different challenges. These are often characterized along a number of axes. The first is which information is available to the learner at training and test time. In our setting we focus on the single-head setting where a learner is unaware at test time which task the data belongs to. In other words, when new classes are presented the learner must learn to distinguish them from all previously observed classes. Another common distinction is the online and offline setting. In the offline setting the learner receives the full set of data for each task and can perform unlimited training on this before moving to the next. On the other hand in the online setting the learner receives one or a small number of samples from a stream and must process and/or store these samples under a computational and memory budget. In this work we focus on the latter.

An extreme continual classification scenario involves the learner observing

individual classes at each changing point in the data stream. Until now this scenario has been studied to a limited degree in the offline setting [44, 103], but has not been considered in the online setting. This is due to the inherent challenging nature of the online setting which has only recently started obtaining results competitive with iid baselines [104, 105]. Indeed, in [103] they use an expensive regularization approach that is impractical under the constraints of online continual learning. Furthermore, the method proposed by [103] uses a pre-trained model, performing an initial stage of offline iid pretraining on half of the classes in the dataset. Where as, our aim is to tackle the problem without any assumption of pre-trained models.

In the online continual learning setting the best performing methods rely on various forms of rehearsal, where old samples are stored in a finite memory and reused at later points in the stream. A common strong baseline is experience replay [47,61]. Recently, Caccia et al. [105] showed that in online continual learning settings, after each distribution shift (task boundaries), the model observes a significant drift in the representation of previously learned classes. They hypothesize that this is fundamentally due to: (i) new class samples representations lying close to older classes and (ii) the loss structure of the standard cross entropy applied on a mix of seen and unseen classes. To mitigate this, they proposed a method to allow fine-grained control over which samples will be pushed away from other samples given an incoming batch. The main downside of methods proposed by Caccia et al. [105], is the critical need for negative samples in the incoming batch to learn the representation of incoming data while avoiding catastrophic forgetting. The result is the inability of their methods to be applied to the more challenging setting where the model observes one class at a time in an online data stream.

In this work, we apply recent ideas from [17], which break the dependence on negative samples in self-supervised contrastive learning to the supervised continual learning setting. This allows us to maintain an asymmetric loss structure between replay samples and incoming data as in [105], while providing learning on new incoming class data without the need to contrast to old classes or their representations. Augmenting this approach with a regularization term that constrains class prototypes, further yields a new method which far exceeds performance of strong online CL benchmarks in the one class incremental setting and is competitive with the top performing methods in the multi-class incremental setting.

4.2 Methods

Similar to [105], given a model $f_\theta(x)$ representing a neural network architecture with parameters θ, we want to minimize the classification loss $\mathcal{L}$ on the newly arriving data batch while not negatively impacting previous learning of other classes, and have the ability to be applied to one-class incremental settings. We opt for a specific loss structure on the incoming batch that would enable the model to learn the representation of each class independently and in isolation from all other classes, either in the incoming batch or in buffered samples. This removes the need for negative samples and enables the model to learn useful representations, even in one-class incremental settings. We first present our supervised modification of BYOL and apply it in the asymmetric setting of [105], then propose our new method.

4.2.1 Supervised BYOL (SupBYOL)

In order to remove the need for negative samples to learn the representation of the incoming batch, we apply a supervised modification of BYOL [17] on the incoming data. BYOL uses a twin architecture with online and target networks. The online network is comprised of three stages: an encoder f_θ, a projector g_θ, and a predictor q_θ. The target network has the same architecture but with different parameters ξ, and is the exponential moving average of the online network: $\xi \leftarrow \tau\xi + (1-\tau)\theta$. We use the following loss, a modification of BYOL loss [17], on the incoming data $\mathbf{X}^{in}$.

$$\mathcal{L}_1^{byol} = -\frac{1}{N} \sum_{\mathbf{x}_i \in \mathbf{X}_{in}} \frac{1}{|P(\mathbf{x}_i)|} \sum_{\mathbf{x}_p \in P(\mathbf{x}_i)} \texttt{sim}(q_\theta(z_\theta), z_\xi) \tag{18}$$

where $\texttt{sim}(a,b) = \frac{a^T b}{\tau \|a\| \|b\|}$, and $z_\theta = g_\theta(f_\theta(x_i))$ and $z_\xi = g_\xi(f_\xi(x_i))$ are the projections outputted by online and target networks respectively. We denote the incoming N data points by $\mathbf{X}^{in}$, data replayed from the buffer by $\mathbf{X}^{bf}$, and the set of positive samples with respect to $\mathbf{x}_i$ by P. Following [105], for the rehearsal step, we apply a modified cross-entropy objective as per [95].

$$\mathcal{L}_2(\mathbf{X}^{bf}) = -\sum_{x_i \in \mathbf{X}_{bf}} \log \frac{\exp\left(\texttt{sim}(\mathbf{c}_{y(x_i)}, f_\theta \mathbf{z_i})\right)}{\sum_{y \in \mathcal{Y}_{all}} \exp\left(\texttt{sim}(\mathbf{c}_y, \mathbf{z_i})\right)} \tag{19}$$

where Y_{all} is the set of all observed classes, and $z_i = g_\theta(f_\theta(x_i))$.

4.2.2 CCP: Continual Contrast of Class Prototypes

Although BYOL can learn the representation of each class independently from other classes, in case of supervised learning, specially where we have multiple classes in the incoming data, it is tricky to evade collapsed representations. In some cases, we observed representation collapse between new classes in the incoming data or old classes in the memory buffer, which results in forgetting of the corresponding classes (drop in accuracy). Also, due to the twin architecture design, BYOL has a larger compute footprint than ER-AML, which makes it less suitable for online continual learning.

Inspired by [106], we use randomly initialized prototypes as class cluster representatives to reduce intra-class variance while enforcing inter-class variance. We represent each observed class $y \in \mathcal{Y}$ by a prototype $\mathbf{c}_y$ in prototypes memory $\mathcal{C}$. The network parameters θ and class prototypes $\mathcal{C}$ are jointly optimized to project an instance $\mathbf{x}_i \in \mathbf{X}^{in}$ of class y close to its corresponding cluster prototype $\mathbf{c}_y$ as well as other samples of class y in the batch, i.e. positive samples $P(\mathbf{x}_i)$. In order to evade collapsed representations, we use a contrast term between class prototypes which enforces inter-class variance. So, as the prototypes act like a representative for the whole corresponding class samples, we can enforce inter-class variance without any need for negative samples, and also provide stable training without BYOL's optimization tricks such as twin architecture and predictor head. We formulate the objective as follows:

$$\mathcal{L}_1^{ccp} = -\frac{1}{N} \sum_{\mathbf{x}_i \in \mathbf{X}_{in}} \left(\texttt{sim}(\mathbf{z}_i, \mathbf{c}_{y(x_i)}) + \frac{1}{|P(\mathbf{x}_i)|} \sum_{\mathbf{x}_p \in \mathbf{P}(\mathbf{x_i})} \texttt{sim}(\mathbf{z}_i, \mathbf{z}_p) \right) + \frac{1}{|\mathcal{C}|} \sum_{\mathbf{c}_i \in \mathcal{C}} \sum_{\mathbf{c}_j \in \mathcal{C}/i} \texttt{sim}(\mathbf{c}_i, \mathbf{c}_j) \tag{20}$$

where $\texttt{sim}(a, b) = \frac{a^T b}{\tau \|a\| \|b\|}$, $z_i = g_\theta(f_\theta(x_i))$, and $y(x)$ denotes the class label of x. As in the case of SupBYOL, we combine this with the same $\mathcal{L}_2$ term applied to the buffered samples. In order to avoid contrast between new and old classes introduced by the last term, we do not directly update old class prototypes using gradients. In other words, for the incoming data, we perform stochastic optimization to minimize $\mathcal{L}_1 + \mathcal{L}_2$ with respect to θ and $c \in \mathcal{C}^{in}$, where $\mathcal{C}^{in}$ is the class labels of the incoming data. However, to update the old class prototypes, we follow [73] and use the replayed samples to obtain a momentum update after each training step to stabilize the model against drastic changes in the representation of learned classes: $c_y \leftarrow \alpha c_y + (1 - \alpha)\bar{c}_y$ where $y \in \mathcal{Y}_{bf}$ denotes the label of old classes stored in the buffer and $\bar{c}_y$ is the

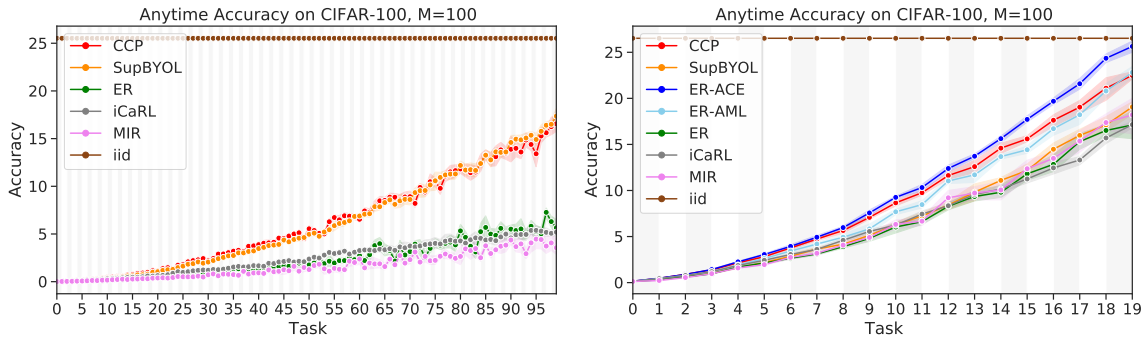


Figure 18: Split CIFAR-100 anytime evaluation results in one-class(left) and multi-class(right) incremental settings with $M = 100$. SupBOYL and CCP greatly outperform the existing methods in one-class setting while being competitive with the top performing methods in multi-class setting. Note that ER-ACE and ER-AML cannot be applied in one-class setting since they require more than one class in the incoming data.

	Accuracy (↑ is better)				Forgetting (↓ is better)		
	$M = 5$	$M = 20$	$M = 50$	$M = 100$	$M = 20$	$M = 50$	$M = 100$
iid online	63.4 ±0.6	63.4 ±0.6	63.4 ±0.6	63.4 ±0.6	N/A	N/A	N/A
fine-tuning	10.0 ±0.0	10.0 ±0.0	10.0 ±0.0	10.0 ±0.0	100.0±0.0	100.0±0.0	100.0±0.0
ER [47]	15.7 ±1.7	18.2 ±2.6	19.6 ±3.1	21.2 ±2.3	54.7±2.5	51.1±2.8	45.7±2.6
iCarl [42]	18.8 ±1.6	20.5 ±1.5	23.3 ±1.2	24.4 ±0.9	46.1±1.7	42.8±1.7	40.1±1.4
MIR [48]	17.8 ±1.8	19.3 ±2.7	20.2 ±2.2	22.0 ±2.4	52.0±2.9	47.4±2.6	41.9±2.8
DER++ [88]	20.1 ±1.4	21.2 ±2.4	23.2 ±2.4	24.6 ±1.5	58.9±2.2	53.3±2.2	49.8±1.8
SupBYOL (ours)	**24.4** ±1.2	30.6 ±1.4	33.3 ±2.1	36.0 ±2.3	28.3±1.6	25.6±2.1	23.1±2.4
CCP (ours)	**24.2** ±0.9	**34.3** ±1.1	**36.0** ±1.3	**39.1**±1.1	**27.2**±1.0	**24.2**±1.1	**21.9**±1.0

Table 8: Accuracy and Forgetting results on Split CIFAR-10 in **one-class** incremental setting(10 tasks) with augmentations and different buffer sizes. Averages and standard deviations are computed over five runs. SupBYOL and CCP outperform other methods with a considerable margin in both Accuracy and Forgetting.

updated prototype for class y.

4.3 Experiments

Our method enables the model to learn the representation of each class in isolation from other classes without the need to have more than one class in the incoming data. Following [75,105], we use a reduced Resnet-18 with *batch size* and the *rehearsal batch size* of *10* samples. All of the models are trained in single head setting, so the task id is not revealed to the model at test time. Similar to [105], we find data augmentation to be useful in most of the settings, specially in simple datasets like CIFAR-10 with a small buffer size where the model might overfit on the buffer samples. Our data

	Accuracy (↑ is better)				Forgetting (↓ is better)		
	$M = 5$	$M = 20$	$M = 50$	$M = 100$	$M = 20$	$M = 50$	$M = 100$
iid online	63.4 ± 0.6	63.4 ± 0.6	63.4 ± 0.6	63.4 ± 0.6	N/A	N/A	N/A
fine-tuning	17.9 ± 0.2	17.9 ± 0.2	17.9 ± 0.2	17.9 ± 0.2	80.9± 0.1	80.9± 0.1	80.9± 0.1
iCarl [42]	33.4± 1.0	39.2± 0.8	41.6± 0.9	42.3± 0.8	31.3± 0.8	30.8± 1.2	29.4± 1.6
ER [47]	28.4± 1.0	40.3± 0.6	42.8± 1.2	49.4± 1.3	26.8± 0.8	24.4± 1.2	22.8± 1.6
MIR [48]	29.8± 1.0	41.8± 0.6	45.6± 0.7	49.3± 0.6	38.2± 1.2	21.6± 0.9	15.8± 1.1
DER++ [88]	31.8 ± 0.9	39.3± 1.0	46.7± 1.1	52.3± 1.1	29.7± 1.1	24.5± 1.0	19.0± 1.1
ER-AML [105]	**36.4**± 1.4	**47.7**± 0.7	**52.6** ± 1.1	**55.7**± 1.3	19.8± 0.4	16.4± 0.4	15.9± 0.3
ER-ACE [105]	35.1± 0.9	43.4± 1.6	49.3 ± 1.2	53.7± 1.1	**18.3**± 0.6	**15.2**± 0.8	**14.6**± 0.8
SupBYOL (ours)	25.4 ± 1.2	36.6 ± 1.3	41.4 ± 1.2	43.6 ± 1.8	20.7± 1.1	18.9± 1.2	17.4± 1.2
CCP (ours)	34.2 ± 0.9	42.0 ± 1.1	47.6 ± 1.0	51.2 ± 0.9	19.7± 0.8	16.3± 1.0	**14.4**± 0.8

Table 9: Accuracy and Forgetting results on Split CIFAR-10 in **multi-class** incremental setting (5 tasks) with augmentations and different buffer sizes. Averages and standard deviations are computed over five runs. Our proposed method, CCP, is competitive with top performing methods in this setting.

augmentation pipline consists of a simple random crop followed by random horizontal flip. We use SGD for optimization with a learning rate of 0.1 as in [48]. We now present the experiments on 10 and 100 task settings for Split CIFAR-10 and Split CIFAR100.

Split CIFAR-10 typically partitions the dataset into 5 disjoint tasks containing two classes each ([48, 97]). In this work we also consider partitioning into 10 disjoint sets (1 class each). When applying the 10 class split we will indicate $S = 1$, while the case of 2 class tasks will be denoted $S = 2$.

We show in Table 8 results for split CIFAR10 on $S = 1$ and for $S = 2$ in Table 9 showing the overall accuracy and forgetting at the end of the task sequence for a variety of memory settings. Observe that in the one class setting ER-ACE and ER-AML cannot be applied as they require other classes in the incoming data. SupBOYL and CCP greatly outperform the existing methods in this setting, with CCP obtaining top performance in all categories. For the multi-class setting of $S = 2$, SupBYOL performs poorly, but CCP greatly improves upon SupBYOL and achieves performance close to ER-AML and ER-ACE in both accuracy and forgetting categories.

Split CIFAR-100 Consists of 100 classes typically split into 20 tasks, each containing a disjoint set of 5 labels ($S = 5$). In our one class incremental work we will also consider the case of splitting into 100 distinct task switches ($S = 1$). All CIFAR experiments process 32×32 images.

In Figure 18, we show the results on $S = 1$ and $S = 5$. In the $S = 1$ setting especially as the number of classes grows, we observe increasing margins over existing

methods for both SupBYOL and CCP. On the other hand, in the $S = 5$ setting, SupBYOL does not perform well as in the CIFAR-10, while CCP matches the performance of ER-AML and is close to the performance of ER-ACE (which nearly matches the i.i.d performance).

4.4 Summary

Our major contribution is a new method to handle online one-class incremental learning without the need for negative contrasts. We demonstrated that this method can outperform strong baselines, and is also applicable and highly competitive in traditional online continual learning settings. Furthermore we have shown that recent advances in self-supervised learning without contrasts can be adapted to supervised settings, particularly in continual classification. Future work can consider if our approach can be applied in the offline one class incremental setting.

Chapter 5

Exploring Representation Forgetting and its Implications

This work was the fruit of collaboration with MohammadReza Davari, Sudhir Mudur, Rahaf Aljundi, and Eugene Belilovsky. It was published as an article in CVPR 2022 [107].

5.1 Introduction

Continual Learning (CL) is concerned with methods for learners to manage changing training data distributions. The goal is to acquire new knowledge from new data distributions while not forgetting previous knowledge. A common scenario is CL in the classification setting, where the class labels presented to the learner change over time. In this scenario, a phenomenon known as catastrophic forgetting has been observed [7, 102]. This phenomenon is often described as a loss of knowledge about previously seen data and is observed in the classification setting as a decrease in accuracy.

Deep learning has been traditionally motivated as an approach, which can automatically learn representations [108], forgoing the need to design handcrafted features. Indeed representation learning is at the core of deep learning methods in supervised and unsupervised settings [109]. In the case of many practical scenarios we may not simply be interested in the final performance of the model, but also the usefulness of the learned features for various downstream tasks [110]. Although a model's representation may change, sometimes drastically at task boundaries [105], this does not necessarily imply a loss of useful information and may instead correspond to a simple transformation. For example, consider a standard multi-head CL setting,

where each task shares a representation and only differs through task specific “heads". A permutation of the features leading into the classification heads leads to total catastrophic forgetting as measured by standard approaches as the task heads no longer match with the representations. However, this does not correspond to a loss of knowledge about the data, nor a less useful representation. Indeed recent works have highlighted the importance of fast remembering versus catastrophic forgetting [111, 112], a looser continual learning requirement where in the task performance may decrease but the agent is able to recover rapidly upon observing a few samples from the previous task. In this light, maintaining a useful representation, which facilitates rapid recovery, is as important as maintaining high performance for the task.

CL envisions having learners operate over long time horizons while continually maintaining old knowledge and integrating new knowledge. Hence, in addition to directly measuring the performance on previous tasks using the last layer classifiers, it is sensible to consider the usefulness of their representations for previous tasks. In this paper we highlight that traditional approaches of evaluating forgetting are unable to properly disambiguate trivial changes in the features (e.g. permutation) from abrupt losses of useful representations. We instead use optimal Linear Probes (LP), commonly used to study unsupervised representations [29] and intermediate layer representations [113, 114], to evaluate CL algorithms and their effectiveness.

We revisit several CL settings and benchmarks and measure forgetting using LP. Our focus is particularly on re-evaluating finetuning approaches that do not apply explicit control for the non-iid nature of continual learning. We observe that in many commonly studied cases of catastrophic forgetting, the representations under naive finetuning approaches, undergo minimal forgetting, without losing critical task information.

Our major contributions in this work are as follows. First we bring three new significant insights obtained and demonstrated through extensive experimental analysis:

1. In a number of CL settings the observed accuracy can be a misleading metric for studying forgetting, particularly when compared to finetuning approaches
2. Naive training with SupCon [35] or SimCLR (in the unsupervised case) have advantageous properties for continual learning, particularly in longer sequences.
3. By using LP based evaluation, forgetting is clearly decreased for wider and deeper models, which is not seen that clearly from earlier observed accuracy.

Secondly, we suggest a simple approach to facilitate fast remembering, which does not require using a large memory during training; it relies only on a small memory combined with SupCon based finetuning.

5.2 Related Work

The design of CL methods is often focused on mitigating the catastrophic forgetting phenomenon, with aspects such as maximization of forward and backward transfer between tasks taken as secondary [60]. One class of methods focuses on bypassing this problem by growing architectures over time as new tasks arrive [16, 67, 115, 116]. Under the fixed architecture setting, one can identify two main categories. The first category of methods rely on storing and re-using samples from the previous history while learning new ones; this includes approaches such as GEM [60] and ER [75]. The second category of methods encode the knowledge of the previous tasks in a prior that is used to regularize the training of the new task [2, 59, 117–119]. A classic method in this vein is Learning without Forgetting (LwF) [2], which mitigates forgetting by a regularization term that distills knowledge [38] from the earlier tasks. The network representations from earlier stages are recorded, and are used during training for a new task to regularize the objective by distilling knowledge from the earlier state of the network. Similarly, Elastic Weight Consolidation (EWC) [120] preserves the knowledge of the past tasks through a quadratic penalty on the network parameters important to the earlier tasks. The importance of the parameters is approximated via the diagonal of the Fisher information matrix [121]. The scale of the importance matrix, λ, determines the network's preference towards preserving old representations or acquiring new ones for the current task. In Sec. 5.4.1 we examine the effectiveness of these approaches in mitigating representation forgetting.

Recent works on elucidating the nature of catastrophic forgetting have examined the influence of task sequence [122], network architecture [123], and change in representation similarity [1]. Our work is related in spirit to [1] as we pursue measuring how much forgetting has occurred on the learned representation and we additionally study this for intermediate representations. One significant difference is that in [1], the authors use linear CKA (centred Kernel Alignment) [124] to measure the similarity between intermediate representations influenced by forgetting, while in our work we measure how much forgetting has occurred on the representations using LP.

Several recent works have also studied the behavior of networks with increasing model capacity. In [125] the authors examine several common architectures under

the task incremental setting and demonstrate that pre-training is essential to combat forgetting and to achieve high performance on all tasks. They conclude that training only with larger models yields no benefit for continual learning. Our analysis revisit this setting and take a closer look at how representation forgetting is affected with increase in model width and depth.

Several works [42] have focused on modifying the last layer of a classification network to make more effective use of the representation for prior tasks. This indirectly highlights the fact that the last layer can be modified to yield better performance on prior tasks. Particularly [42, 89] use a buffer of old examples at training time to improve learning and then use them at evaluation time to construct a class mean prototype. This allows for more effective use of the representations of the network. These works consider settings where the CL methods are used to control forgetting, while we also emphasize that naive continuation of training under task shift can yield strong representations. Our work can also be seen as both a way to explain and to motivate the need for such approaches.

Self-supervised learning (SSL) is becoming increasingly popular in visual representation learning. Some of the best performing methods rely on contrastive learning [29, 126]. These methods have been recently evaluated in a limited continual learning setting [127] where a sequence was trained on non-iid unsupervised streaming data and then applied in transfer learning settings on multiple datasets. However, forgetting was not evaluated. In contrast our work, which also uses a SSL loss, focuses on the LP evaluation and the study of representation forgetting with respect to previously seen distributions. Contrastive methods are also often used in the supervised setting, for example, the recently popular SupCon loss [35]. In [105] and [89] the use of SupCon is proposed in the online class-incremental setting in combination with experience replay. Our work too considers SupCon as one of the supervised representation learning approaches. However distinct from the other works we consider it in the offline task-incremental setting. We do not look at its use in combination with replay or other approaches, but study the effect of standard finetuning with SupCon loss, distinct from [105], where it is used to facilitate separation of contrasts between old and new classes, specific to the class incremental setting.

5.3 Background and Methods

In this section we review the key tools used in our analysis including linear probes, centered kernel alignment, and contrastive loss functions. Finally we discuss how the nearest mean of exemplars approach can be used in the context of non-rehearsal based methods, such as finetuning with SupCon, as a simple continual learning method that also facilitates rapid remembering.

5.3.1 Linear Probes for Representation Forgetting

Following the work in SSL [29] and in the analysis of intermediate representations [114] we evaluate the adequacy of representations by an optimal linear classifier using training data from the original task. A linear classifier is trained on top of the frozen activations of the base network given the training instances of a certain dataset. The test set accuracy obtained by using LP on that dataset is used as a proxy to measure the quality of the representations. The difference in performance of the LP before and after a new task is introduced, acts as a surrogate measure to the amount of forgetting observed by the representations and is referred to as representation forgetting.

Formally, for a given model f_{θ_i} computed from time step i of a task sequence, we compute its optimal classifier $W_i^* = \arg\min_{W_i} \mathcal{L}(W_i f_{\theta_i}(X_i), Y_i)$, where $\mathcal{L}$ is the objective function, and X_i and Y_i correspond to the data from task i. To assess representation forgetting between θ_a and a model at a later point in the sequence, say θ_b, we evaluate $T(W_a f_{\theta_a}(X_a), Y_a) - T(W_b f_{\theta_b}(X_a), Y_a)$ where T is the task performance (accuracy).

5.3.2 Centered Kernel Alignment

CKA [124] is a recent popular approach to compare representations. It has been commonly used to compare representations across depth as well as across models from different tasks in the CL settings [1]. Given a dataset comprised of m samples, and their representations X and Y, with features n_x and n_y respectively, $X \in \mathbb{R}^{m \times n_x}$ and $Y \in \mathbb{R}^{m \times n_y}$, the, typically used, linear CKA between X and Y is given as $\frac{\left\|Y^T X\right\|_F^2}{\|X^T X\|_F^2 \|Y^T Y\|_F^2}$. This similarity metric has the advantage of being invariant to scaling and orthogonal transformations. However, being a simple linear alignment comparison it is not invariant to general classes of invertible transformations. Furthermore, relative comparisons of CKA metrics are challenging to interpret compared to task performance degradation. CKA has been used in [1] to

compare the intermediate representations of a model in consecutive task increments t and $t+1$. Ramasesh *et al.* [1] proxy the CKA similarity between the intermediate representations of the model f_{θ_t} and $f_{\theta_{t+1}}$ to measure representation forgetting. Thus, under this paradigm, a high value of CKA similarity is interpreted as minimal representation forgetting. One limitation of this metric for representation forgetting is its inability to distinguish between positive and negative backward transfer (see Sec. 5.4.4). This is addressed when measuring representation forgetting via LP.

5.3.3 Supervised and Unsupervised Contrastive Loss

Contrastive loss functions have become increasingly popular in representation learning, particularly visual representation learning. They have led to large advances in unsupervised learning [29,126]. As well they are becoming a popular alternative to cross-entropy (CE) in the supervised setting [35,36], referred to as SupCon. Given a representation f_θ, often consisting of a primary network and a projection, the SupCon loss for a minibatch X is given by:

$$\sum_{\mathbf{x}_i \in \mathbf{X}} \frac{-1}{|P(i)|} \sum_{\mathbf{x}_p \in P(i)} \log \frac{\texttt{sim}(f_\theta(\mathbf{x}_p), f_\theta(\mathbf{x}_i))}{\sum_{\mathbf{x}_a \in \mathbf{X}/x_i} \texttt{sim}(f_\theta(\mathbf{x}_a), f_\theta(\mathbf{x}_i))}$$

Where $\texttt{sim}(a,b) = \exp(\frac{a^T b}{\tau \|a\| \|b\|})$ and $P(i)$ represents the same class samples as x_i from the minibatch, and the denominator is taken over all other samples. Note that we consider SupCon in the naive setting, thus all minibatches are from the current task in our evaluations of SupCon. Similar to this, in the unsupervised setting the SimCLR loss [29] is given by:

$$-\sum_{\mathbf{x}_i \in \mathbf{X}} \log \frac{\texttt{sim}(f_\theta(\mathbf{x}_p(i)), f_\theta(\mathbf{x}_i))}{\sum_{\mathbf{x}_n \in \mathbf{A}(i)} \texttt{sim}(f_\theta(\mathbf{x}_n), f_\theta(\mathbf{x}_i))}$$

Where $A(i)$ corresponds to all minibatch examples and their data augmentations except x_i, and $x_p(i)$ represents an augmented version of x_i.

5.3.4 Exemplars and Fast Remembering

Many CL methods utilize buffers of exemplars [42] that are constantly updated. Typically, these samples are used repeatedly to train the model [75]. In [42, 89], the samples in the memory are also used to continuously estimate a class mean for old samples using the new representation. This is then used to construct a nearest mean of exemplars (NME) classifier, which can be seen as a fast way to construct a strong classifier that requires only a small amount of data. Instead of relying on the

same exemplars during training and inference, one can use a small set of exemplars from the task distribution at the end of a task sequence to construct an NME based classifier in combination with a non-CL specific representation learning method such as SupCon [35]. Specifically the learner maintains a class mean for any class it has encountered. These class means are updated either by using a stored set of samples that is only used at inference or upon encountering an old task again, obtaining a small set of new samples to facilitate fast remembering. Notably, unlike the prior work on NME classifiers in continual learning, we don't suggest using exemplars as a rehearsal memory during the training, but as a method for fast remembering of class means for old tasks during evaluation time. This has the advantage of not increasing the computational complexity of training and not needing additional overhead in storing or retrieving samples, except at the end of a task sequence or upon re-encountering a task. Specifically, it can facilitate rapid remembering; in situations without prior data stored. Upon encountering a new task a model with minimal representation forgetting can rapidly adapt by updating just its class means.

5.4 Experiments

We perform evaluations in several CL scenarios, focusing on the task-incremental setting. The evaluations are based on LP and observed accuracy. Observed accuracy refers to the standard accuracy used in the CL literature. Specifically we measure observed accuracy, A_{ij}, as the accuracy of the model after step i on the test data of task j. Similarly, the average observed accuracy at the end of the sequence is $\frac{1}{T}\sum_{t\in T} A_{T,t}$ as used in [2]. Similarly we can measure the LP accuracy for each step i and task j as well as the average LP accuracy.

Datasets We use an ImageNet transfer setting based on [2], a common SplitCIFAR100 [128] setting (split into 10 tasks), and reproduce the SplitCIFAR10 (split into 2 tasks) setting from [1]. Finally, to evaluate in very long task sequence regimes, we use a downsampled version of the entire ImageNet dataset (ImageNet32 [129]) split into 200 tasks of 5 classes each. For the ImageNet transfer setting, we use a sequence consisting of the standard ImageNet (LSVRC 2012 subset) [110], MIT Scenes [130] for indoor scenes classification (5,360 samples over 67 classes), Caltech-UCSD Birds (CUB) [131] for classification of bird species (6,033 samples over 200 classes), and Oxford Flowers [132] for flower classification (2,040 samples over 102 classes). The use of this sequence allows us to complement the standard

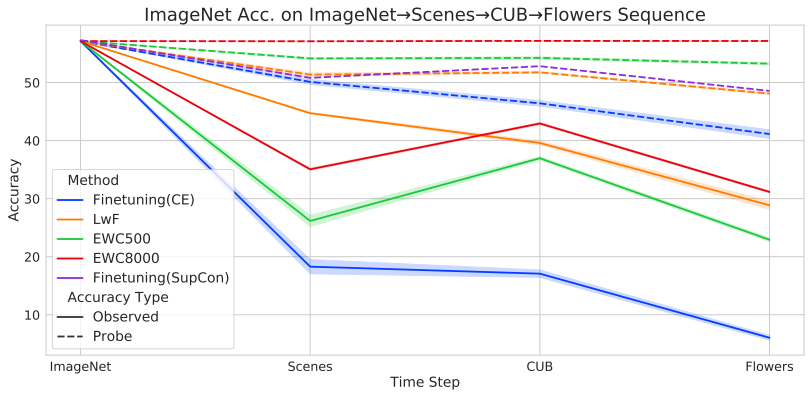


Figure 19: Performance on ImageNet during the sequence (ImageNet→Scenes→CUB→Flowers) using ResNet18. We observe that although observed accuracy heavily degrades the LP accuracy, in finetuning does not decay as drastically and can rival LP accuracy of methods such as LwF and EWC. Moreover, we observe that the LP Accuracy of SupCon training, which has no control for forgetting, outperforms the LwF, a method designed for CL. Note EWC with $\lambda = 8k$ is the best performing method in terms of LP and observed acc., however it does not perform well on the current task (see Tab. 10).

long task sequence benchmarks with a more realistic and diverse larger scale sequence. Optimization hyperparameters for training are detailed in the Appendix.

Methods Compared Our work focuses on evaluating naive finetuning based approaches using CE and SupCon [35] loss functions, as well as a set of representative CL methods. For regularization-based baselines, we consider two of the most popular methods, which do not utilize memory of any past samples: LwF [2] and EWC [120]. For rehearsal-based baselines, which continuously store past samples we focus on Experience Replay (ER). Indeed a number of recent works have illustrated that ER, particularly with increase in buffer size, is a strong baseline [1,75] and rivals or exceeds other rehearsal based methods such as iCaRL [42] and GEM [60]. Hence we use ER with both a small buffer, $M = 5$ samples per class, and a relatively large buffer, $M = 20$ samples per class.

5.4.1 Observed vs LP accuracy

In this section we study the observed vs LP accuracy for various task sequences and methods, in both supervised and unsupervised settings.

Method	Acc. Scenes	Acc. CUB	Acc. Flowers
FT (CE)	$56.9\% \pm 1.1$	$54.5\% \pm 2.6$	$89.3\% \pm 1.1$
LwF	$57.6\% \pm 1.5$	$43.1\% \pm 2.9$	$85.3\% \pm 0.5$
$\text{EWC}_{\lambda:0.5k}$	$52.5\% \pm 1.1$	$47.8\% \pm 2.5$	$85.9\% \pm 1.6$
$\text{EWC}_{\lambda:8k}$	$42.1\% \pm 1.5$	$38.3\% \pm 0.9$	$79.1\% \pm 1.0$
FT (SupCon)	$57.1\% \pm 1.2$	$50.4\% \pm 1.0$	$85.3\% \pm 0.9$

Table 10: Observed accuracy of the current task in the sequence ImageNet→Scenes→CUB→Flowers using ResNet architecture. Although $\text{EWC}_{\lambda:8k}$ attains relatively poor performance on the current task, it achieves the highest LP and observed accuracy for the previously seen tasks (see Fig. 19). Moreover, the SupCon training achieves comparably high accuracy on the current task (even surpassing CE on Scenes) while suffering from relatively small representation forgetting (see Fig. 19).

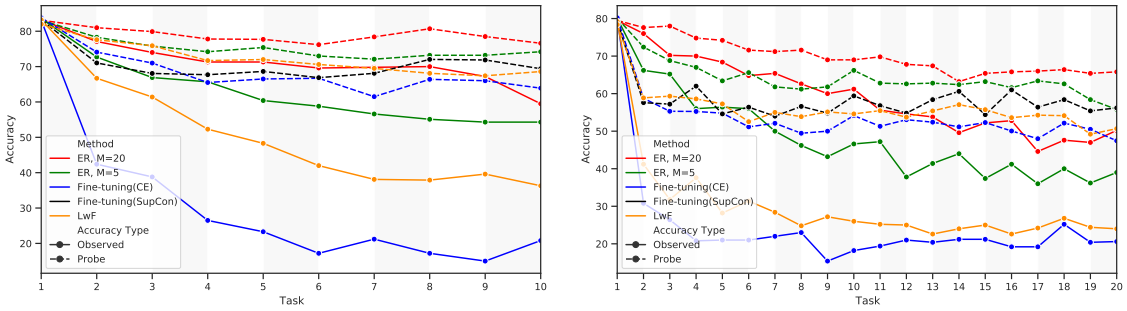


Figure 20: 10-Task SplitCIFAR100 and 20-Task SplitMiniImageNet. We compare observed accuracy and linear probe accuracy Naively finetuning with CE does poorly if using the observed accuracy. However using the LP based evaluation we observe that performance gap to other methods is lower. Furthermore when finetuning is performed instead with the SupCon loss function LP performance can rival that of LwF.

ImageNet Transfer We consider models trained on the large ImageNet data and subsequently applied to different tasks in the sequence. We take the setting of [2], which considers the ImageNet [110] transfer to various datasets, in particular CUB [133], and Scenes [130]. We extend this setting by including Flowers [132] in the task sequence. To reduce the computation of experiments we do random resize crops to 64×64 and utilize ResNet-18 for these experiments. Additional results further confirming our observations with larger crop size are given in the Appendix. As mentioned earlier, in addition to LwF, we also examine EWC [120] under two conditions: (a) large λ value ($8k$), so the network is inclined to preserve the knowledge important to the previous tasks, and (b) small λ value ($0.5k$), so the network is

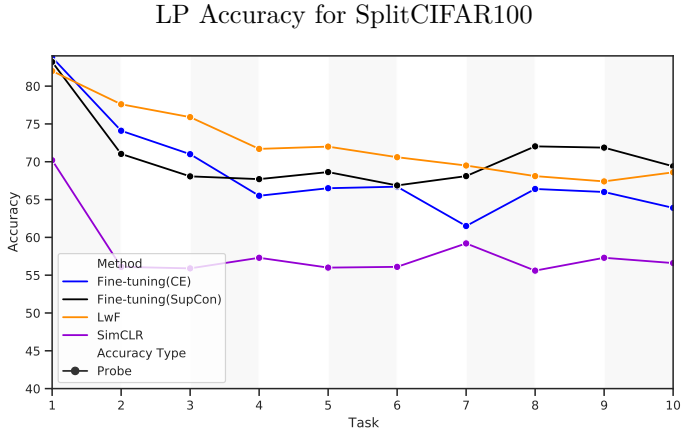


Figure 21: SplitCIFAR100 comparison of unsupervised Linear Probe accuracies on task 1 with supervised finetuning CE and SupCon as well as LwF. We observe that LwF and CE based finetuning can decay over time, while the unsupervised learning (SimCLR) has an initial drop and stays relatively flat.

encouraged to perform competitively on the current task.

We report observed accuracy and LP accuracy on ImageNet validation set as the model is trained on the task sequence ImageNet $\rightarrow$ Scenes $\rightarrow$ CUB $\rightarrow$ Flowers (see Fig. 19). We also report the observed accuracy on the current task in Tab. 10.

Our evaluation reveals that although the forgetting in terms of the traditional measure is high for finetuning compared to LwF (as shown in [2]), the LP accuracy of these methods suggest less drastic forgetting. Furthermore, the LP performance across finetuning and other methods is not as drastically different as their respective observed accuracies are. We see that the LP accuracy of SupCon based finetuning, which has no explicit control for forgetting, outperforms LwF, a method specifically designed for CL. It also closely tracks the performance of $\text{EWC}_{\lambda:0.5k}$, while outperforming on the current task performance. Indeed, as we can see in Tab. 10, SupCon training achieves comparably high accuracy on the current task (even surpassing CE finetuning on Scenes) with relatively small representation forgetting (see Fig. 19).

SplitCIFAR100 and SplitMiniImageNet We now consider the SplitCIFAR100 with 10 tasks of 10 classes each and the 20 task SplitMiniImageNet setting. We show in Fig. 20 the performance on the first task throughout the sequence for both settings. Similar to the previous case we see: for finetuning with CE the LP based evaluation shows much milder forgetting than observed accuracy. When finetuning with SupCon, LP performance drops initially but then stays relatively flat and even

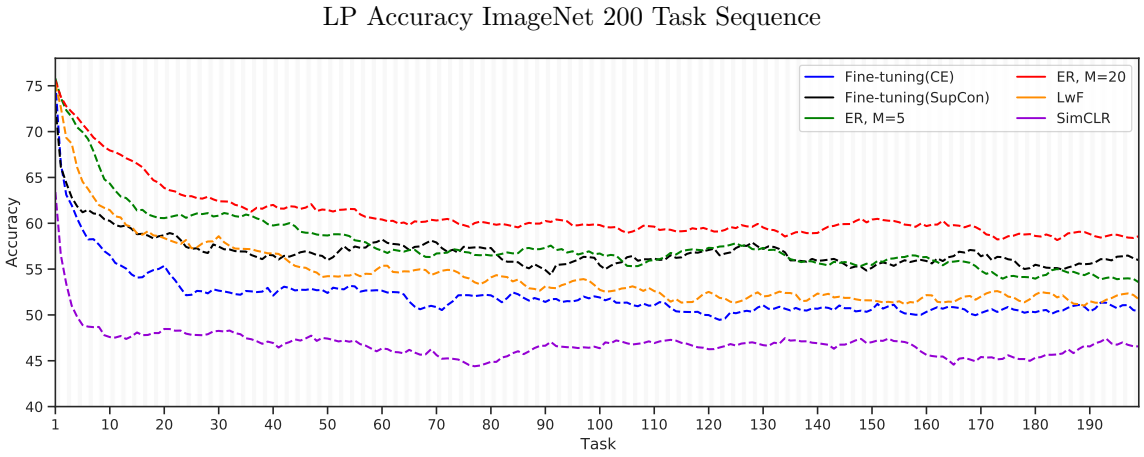


Figure 22: 200-Task ImageNet32. We compare Linear probe accuracy for Tasks 1 data over the whole sequence. As the model observes the later tasks of the sequence, the performance of finetuning with CE reaches LwF, and finetuning with SupCon outperforms ER with 5 samples per class.

increases, suggesting that positive backward transfer is occurring. Overtime, SupCon outperforms the LwF [2] approach without any specific CL based control in both task sequences. It also obtains performance that becomes close to ER with 5 samples per class without access to any previous data. For the longer MiniImageNet task sequence we see that over time even the strong ER based baselines which train on old data demonstrate a reduced LP performance, while finetuning baselines remain relatively flat and even increase in the case of SupCon. This suggests that in very long sequences, finetuning baselines can be competitive to the more computationally expensive CL methods.

Utilizing SplitCIFAR100 we also consider the unsupervised representation learning case where linear probes follow naturally as a common evaluation setting. The literature on evaluation of continual learning methods in the unsupervised setting is limited. Hence we directly compare unsupervised and supervised approaches in their representation learning ability when presented with the same task sequences. We focus on the SimCLR loss and evaluate LP performance in comparison to other methods in Fig. 21. We can see in Fig. 21 that the initial LP performance is lower for SimCLR compared to the supervised losses. This is natural as it does not have access to the labeled data. Despite the higher starting accuracy, LwF and finetuning with CE show a decay that continues in the first several tasks following task 1. On the other hand SimCLR decays at the first step but then remains nearly flat over the rest of the sequence, showing a strong resistance to representation forgetting after this inital drop. However SupCon, which utilizes a loss similar to SimCLR in

		Task 1 Acc.	Obs. Acc. Task 1 at T=10	Task 1 LP T=10	LP Acc. All T=10	Avg. Obs. Acc.
Finetuning(CE)	RN18, Width=32	82.2	20.8	64.8	70.8	35.5
	RN18, Width=128	83.3	21.2	70.5	74.2	36.8
	RN101, Width=32	82.9	19.8	67.9	72.4	35.9
ER-M5	RN18, Width=32	82.7	52.1	74.2	75.7	54.8
	RN18, Width=128	83.6	54.8	75.6	77.3	55.4
	RN101, Width=32	83	50.9	74.5	76.1	51.6
ER-M20	RN18, Width=32	82.4	61.3	76	76.4	65.2
	RN18, Width=128	83.2	63.5	78.8	80.1	67
	RN101, Width=32	82.9	60.7	77.1	77.5	63.9
LwF	RN18, Width=32	82.1	36.2	70.1	73.4	47.7
	RN18, Width=128	83.9	37.7	74.8	76.7	49.1
	RN101, Width=32	82.5	35.5	71.0	74.6	46.3

Table 11: Final Accuracy of 10 task SplitCIFAR100 sequence with variable width and depth in the offline setting. M indicates the number of samples per task used in the ER buffer. We observe that simple finetuning and LwF baselines show large forgetting, which do not improve significantly with width or depth. On the other hand, the LP evaluation reveals that representation quality for finetuning and LwF becomes closer to strong CL methods that store samples and also use more compute, *e.g.* ER. Furthermore, LP evaluations reveal LwF with wider models can rival ER.

the supervised setting, shows the best of both worlds, has an initial drop and then illustrates gradual backward transfer properties.

200 Task Sequence - SplitImageNet32 We now consider a much longer sequence than typically studied in the literature to allow us to observe whether the trends we have seen so far continue to hold. Using Imagenet32 we construct 200 tasks of 5 classes each. Fig. 22 shows the performance on the first task throughout the whole sequence. We see that in a very long sequence of tasks, the previous trends are kept. Specifically, we see that as the model reaches the later stages of the sequence, finetuning with CE reaches LwF, and finetuning with SupCon outperforms the competitive baseline of ER with a small buffer *without* access to buffer data. Furthermore, we observe as in the previous section that SimCLR performance stays flat. In the supplementary material we also demonstrate that this pattern is not limited to the first task but is maintained for other tasks along the sequence.

5.4.2 Effects of Increased Model Capacity

Next, we use linear probes to evaluate the effect of increased model width and depth. Recently [125] has suggested that increased model size must be strictly combined with pre-training in order to get increased robustness to catastrophic forgetting. We revisit

		Task 1 Acc.	Obs. Acc. Task 1 at T=10	Task 1 LP T=10	LP Acc. All T=10	Avg. Obs. Acc.
Finetuning(CE)	RN18, Width=32	18.6	12.2	39.8	36.4	22.3
	RN18, Width=128	19.4	12.7	42.3	41.7	19.8
	RN101, Width=32	14.6	11.8	28.2	29.4	14.5
ER-M5	RN18, Width=32	18.8	27.3	36.0	40.1	33.8
	RN18, Width=128	19.5	28.9	54.7	47.9	31.6
	RN101, Width=32	15.0	24.7	37.1	30.4	24.3
ER-M20	RN18, Width=32	18.4	32.0	46.8	43.5	34.7
	RN18, Width=128	20.0	31.8	51.2	50.7	32.5
	RN101, Width=32	14.5	25.4	36.5	33.9	24.3
LwF	RN18, Width=32	18.5	13.4	29.5	36.0	22.7
	RN18, Width=128	19.7	18.3	34.6	39.1	22.1
	RN101, Width=32	14.8	11.1	25.4	22.8	16.8

Table 12: Final Accuracy of 10 task SplitCIFAR100 sequence in the Online Setting. LP evaluations show that width substantially improves online representation learning, while observed Avg Accuracies suggest it decreases. Increasing depth on the other hand appears to be less effective in the online setting.

this in the context of both wider and deeper models on a SplitCIFAR100 sequence of 10 tasks with 10 classes each. Table 11 shows the results using a Resnet18 with a much wider model (128 vs. 32) and then a much deeper model (101 layers). We report both the LP accuracy of task 1 at the end of the sequence and the average of LP accuracies for all the tasks at the end of the sequence.

First, we can see that as in the other cases, the LP accuracy of finetuning is higher than observed accuracy, suggesting that forgetting is less catastrophic than what is indicated by observed accuracy. Secondly, we see that finetuning evaluated using the observed accuracy is particularly deceptive in revealing how the model representations changes with increasing capacity. The observed and accuracy of task 1 are relatively close despite increasing capacity (wider or deeper) while the corresponding LP accuracies show substantial gaps. Using observed accuracy one would conclude that increasing width and capacity of the model without applying any CL specific method does not reduce forgetting. This is consistent with the observations of [125], which evaluates only on observed accuracy. However, if we observe the LP accuracy, it reveals a more clear picture of what occurs at the representation level, suggesting that larger models can indeed reduce forgetting even when trained from scratch without explicit control of forgetting. Moreover, we see that at the representation level as model capacity increases, naive finetuning becomes much closer in performance to costly (and under privacy constraints unusable) CL methods such as ER, which require more compute and memory.

In comparing depth and width we also see some key distinctions - increasing width

Method	Obs Acc. Task 1 at T=10	Avg. Obs. Acc.
Finetuning(CE)	20.8%	35.5%
ER-M5	52.1%	54.8%
ER-M20	61.3%	65.2%
LwF	36.2%	47.7%
Finetune(SupCon) + NME-M5	48.0%	53.9%

Table 13: Final Accuracy of 10-task SplitCIFAR100 sequence comparing only the observed accuracy and SupCon+NME. Supcon+NME gives superior performance to CL specific methods such as LwF and nearly matches the performance of ER with a similar memory size while not needing access to the memory during task training.

appears to help more than increasing depth. For ER we also see that increasing depth yields a lower observed accuracy, while the LP evaluation suggests the representations are similar. Similarly, in Tab. 12 we report the results for the online task-incremental setting [60, 75]. In this setting, momentum tends to be detrimental to performance, thus we use a fixed learning rate of 0.01 with no momentum. We see similar behavior to the previous case, the larger models can end up appearing to do worse if we consider observed accuracy, but perform better using LP evaluation. Wider models appear to do particularly well in the online setting while deeper models have degraded LP accuracy in this setting. Finally we see that LwF which is a regularization method performs poorly in this setting. Indeed regularization based methods do poorly in the online setting. This suggests that amongst methods without access to a replay buffer, finetuning may provide the best representation learning.

5.4.3 Low-Cost Remembering with SupCon

The observed low representation forgetting properties of finetuning with SupCon loss suggest if we can approximate a classifier using its representation it would allow for low cost remembering upon encountering a previously observed task. We thus evaluate the use of the NME in combination with SupCon. As discussed in Sec 3.4 such an approach allows a simpler alternative to ER methods and moreover facilitates fast remembering not relying on a buffer and repeatedly training the model with old samples. We use the SplitCIFAR100 dataset to compare against several CL specific methods such as LwF and ER in Tab. 13. We use a memory with $M = 5$ samples per class for this. We chose the exemplars at random to simulate re-encountering an old task. We observe that just applying the simple approximation with a small

ResNet: Network Acc. on T-1 after T-2 training: 63.64%				
Block	LP Acc. Post T-1	LP Acc. Post T-2	Δ Acc.	CKA*
B-0	63.54%	64.62%	+1.08%	0.97
B-1	68.24%	69.50%	+1.26%	0.93
B-2	71.62%	71.34%	−0.28%	0.88
B-3	77.64%	76.52%	−1.12%	0.78
B-4	80.06%	78.98%	−1.08%	0.31
B-5	85.82%	80.10%	−5.72%	0.22
VGG: Network Acc. on T-1 after T-2 training: 57.88%				
B-0	67.94%	66.86%	−1.08%	0.95
B-1	73.60%	72.52%	−1.08%	0.93
B-2	78.58%	75.68%	−2.90%	0.85
B-3	81.54%	75.48%	−6.06%	0.66

Table 14: Representation forgetting of Task 1 measured via optimal linear probes (LP) on ResNet and VGG. The Accuracy degradation of LP trained on activations of stages (blocks of convolutions) before and after observing Task 2 suggests that the representations are still highly useful for Task 1 despite training on Task 2. *Note CKA results are taken from [1] for comparison.

number of samples allows for a rapid recovery of the performance with the finetuning approach alone, exceeding the performance of LwF on overall accuracy and task 1 accuracy. The overall performance is close to that of ER with the same memory size and slightly below the ER performance on task 1 at the end of the sequence. On the other hand ER requires samples to be available during the entire training sequence, requires the addition of extra algorithmic elements specifically to control forgetting, and uses substantially more compute ($\approx 2\times$ that of the finetuning step in this case).

5.4.4 Depth-wise Probes and Comparison to CKA

We consider a 2-task SplitCIFAR10 setting from [1]. We use the same models and training procedures and subsequently evaluate forgetting. In Tab. 14, we study the shift in representations of each block of the network by measuring the performance of LP on task 1 before and after training the network on task 2.

First we see that the observed accuracy decreases from 85% to 63%, suggesting large degradation in performance and large forgetting. However, following the optimal classifier evaluation protocol the accuracy degradation is seen to be only 5.7%, without any CL method applied to control forgetting. This suggests that the representations are still highly useful for Task 1 despite training on Task 2. Second, similar to [1] we

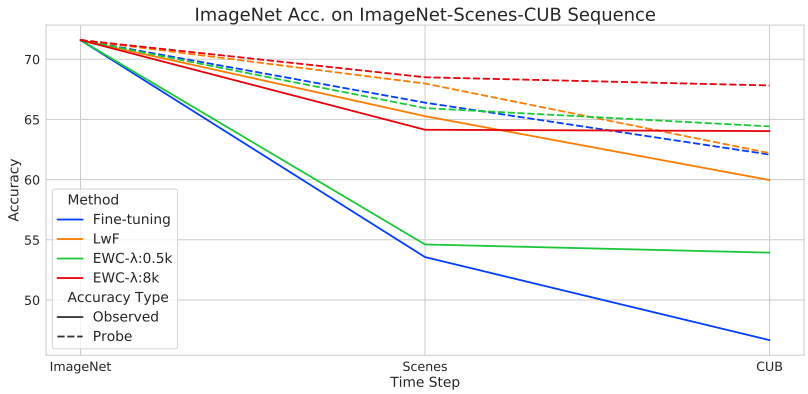


Figure 23: Performance on ImageNet during the transfer sequence (ImageNet→Scenes→CUB) using VGG-16. We observe that although observed accuracy heavily degrades, the LP accuracy for finetuning does not decay as drastically and can rival LP accuracy of methods such as LwF and EWC. We evaluate methods which do not rely on storing data from Task 1 to replay during training. Note EWC with $\lambda = 8k$ is the best performing method in terms of LP and observed accuracy, however it does not perform well on the current task (see Tab. 15).

Method	Acc. Scenes	Acc. CUB
Finetuning	74.70%	74.39%
LwF	74.78%	75.23%
$\text{EWC}_{\lambda:0.5k}$	74.70%	74.72%
$\text{EWC}_{\lambda:8k}$	72.69%	71.44%

Table 15: Observed accuracy of the current task in the sequence ImageNet→Scenes→CUB using VGG-16 architecture. Although $\text{EWC}_{\lambda:8k}$ attains relatively poor performance on the current task, it achieves the highest LP and observed accuracy for the previously seen tasks (see Fig. 23).

note that the forgetting is concentrated at the top layers. Indeed early layers in the network experience almost no representation forgetting and in some cases improve their usefulness with regards to Task 1.

Ramasesh *et al.*'s [1] analysis also showed forgetting occurring in early layers to a lower degree than in higher layers and suggested that forgetting is extreme in the upper layer representations. Specifically, the authors measured linear CKA [124] performance between layers (given in Tab. 14) showing that this similarity metric dropped progressively from close to 1 to 0.2 for both ResNet and VGG models. However, our evaluation suggests that forgetting does not exist in lower layers and also the loss in information is less catastrophic at higher layers than suggested by [1].

	ImageNet (T-1) → CUB (T-2)	
	Obs. T-2 Acc.	Obs. T-1 Acc.
Finetune- [2]	73.1%	50.7%
Finetune-Ours	74.5%	50.9%
LwF- [2]	72.5%	60.6%
LwF-Ours	75.7%	63.6%
	ImageNet (T-1) → Scenes (T-2)	
Finetune- [2]	74.6%	62.7%
Finetune-Ours	74.7%	53.6%
LwF- [2]	74.9%	66.8%
LwF-Ours	74.8%	65.3%

Table 16: Reproduction of the results reported in [2]. Note that we observe a slight difference in our reproduced results due to stochasticity of training neural networks, and removing the warm-up step.

Observed Acc. on ImageNet: 71.59%			
	ImageNet (T-1) → CUB (T-2)		
	T-1 Acc. @ T-2	LP Acc. @ T-2	T-2 Acc.
FT	50.89%	64.49%	74.51%
LwF	63.58%	67.23%	75.65%
$\text{EWC}_{\lambda:8k}$	60.28%	67.46%	72.70%
$\text{EWC}_{\lambda:0.5k}$	50.78%	63.99%	74.53%
	ImageNet (T-1) → Scenes (T-2)		
FT	53.56%	66.39%	74.70%
LwF	65.27%	67.98%	74.78%
$\text{EWC}_{\lambda:8k}$	64.14%	68.50%	72.69%
$\text{EWC}_{\lambda:0.5k}$	54.61%	65.94%	74.70%

Table 17: Forgetting of Task 1 measured via optimal linear probes (LP). Note that although the forgetting is much higher for finetuning compared to LwF, the LP accuracy is nearly identical, especially for the ImageNet → Scenes task.

5.4.5 ImageNet→ Scenes→ CUB

For ease of experiments, in Sec. 5.4.1 we have used a lower resolution, 64×64, a light ResNet-18 model, and a modern rapid training scheduler. On the other hand in this section we reproduce the results of the original paper (Li *et al.*) [2] using a VGG-16 model and 224×224 input size and then carry out our LP analysis further confirming our observations. We take the setting of [2], which considers the ImageNet [110] transfer to various datasets, in particular CUB [133], and Scenes [130]. We use the same model architecture (VGG-16) and training procedures described in [2], which proposes LwF, closely reproducing their ImageNet → Scenes as the first step in the sequence (see Tab. 16). We also include an EWC baseline under two conditions: (a) large λ value ($\lambda = 8k$), so the network is inclined to preserve the knowledge important to the previous tasks, and (b) small λ value ($\lambda = 0.5k$), so the network is encouraged to perform competitively on the current task. The results are shown in Figure 23 and Table 15.

We first note that our results for the first task switch are consistent with those reported in [2] (see Tab. 17). Fig. 23 reveals that although the forgetting in terms of the traditional measure is high for finetuning compared to LwF (as shown in [2]), the LP accuracy of these methods suggest a much less drastic forgetting. Furthermore, the LP performance across finetuning and other methods is not as drastically different as their respective observed accuracies are. Indeed, we observe that on the third task, finetuning outperforms LwF in representation forgetting on ImageNet. Similarly

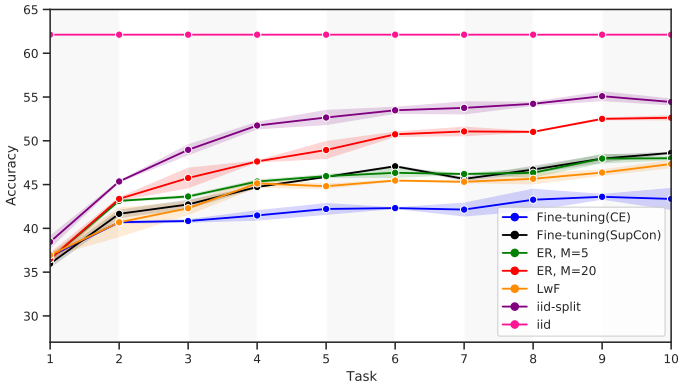


Figure 24: All-LP anytime evaluation plot on SplitCIFAR100 10-task sequence. All-LP is a probe trained on all, *i.e.* seen and unseen tasks, training data and evaluated on all test data. We compare this with splitting iid data into 10 subsets trained in sequence, denoted as iid-split.

EWC does not clearly outperform naive finetuning. For example, if using one hyper-parameter for the regularization term the performance closely tracks finetuning. On the other hand using $\lambda = 8k$ we observe the best LP performance on ImageNet through the task sequence but degraded current task performance as seen in Tab. 15.

Although we followed the training procedure as closely as possible to the ones reported by [2], the results are slightly different from the ones reported in [2] due to (a) not using the task-head warm-up step, where the backbone network is first frozen and the newly added task head is trained until convergence (warm-up), and then the entire network is trained until convergence, and (b) stochasticity of training neural networks. Table 16 highlights these differences.

5.4.6 Comparing Overall Representation Improvement

In addition to representation forgetting, we consider also measuring how much a representation improves overall as seen by a linear probe trained and evaluated on data from the union of all current and future tasks. We can evaluate this by training at each step in the sequence an “All-LP", that is a linear probe trained on all the training data and evaluated on all test data. A natural baseline to compare such an approach to is splitting iid data into 10 subsets trained in sequence (we denote this iid-split). The results of this evaluation are shown for SplitCIFAR100 in Figure 24. We observe again that SupCon exceeds LwF methods and competes with the small replay-sized ER method.

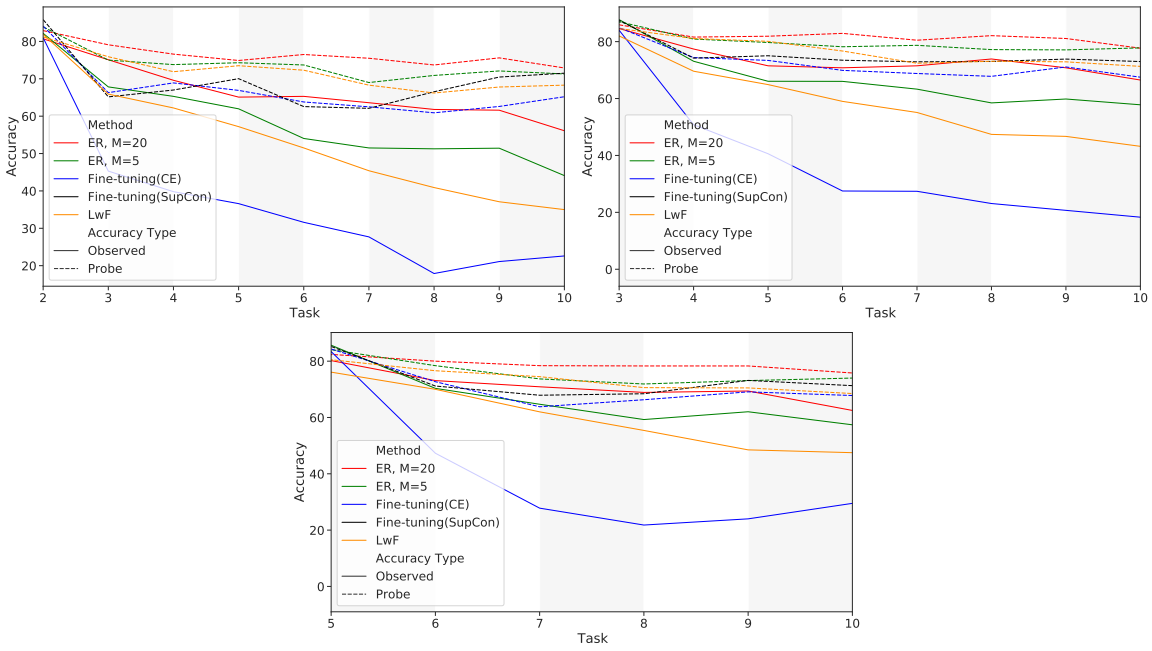


Figure 25: LP and Observed accuracy for Task 2 (upper left), 3 (upper right), and 5(bottom) on 10-Task SplitCIFAR100.

5.4.7 Representation Forgetting for Other Tasks

Complementing the results in Sec. 5.4.1, we report the observed and LP accuracies for methods when measured for tasks beyond Task 1. Specifically for SplitCIFAR100 we show trends for Task 2, 3, and 5 in Fig. 25. We observe similar trends as reported for Task 1 in Sec. 5.4.1.

5.4.8 Training Details

ImageNet → Scenes → CUB → Flowers we use images downsampled and cropped at 64×64 (224×224 size results are shown in Sec. 5.4.5) in this task sequence. We train a ResNet-18 [4], using AdamW [134] optimizer with a learning rate (LR) of $1e$-5 and a weight decay of $5e$-4. We first train the model on ImageNet via (1) CE loss, or (2) SupCon loss for 500 epochs. Next, for each task in the sequence, we train the model via early stopping [135], and report the observed accuracy over the test set (see Tab. 10).

The LP classifiers are trained using the same optimizer and weight decay value. However, for faster convergence, in this setting, we use a cosine LR scheduler [136] with an initial LR of $1e$-4. The results of LP classifiers are reported on the ImageNet validation set. All the training images in this section undergo, random crop, random

horizontal flip, and color jitter of 0.5.

SplitCIFAR100, MiniImageNet, ImageNet32 For our SplitCIFAR100 10-task sequence, MiniImageNet 20-task sequence, and ImageNet32 200-task sequence we use SGD optimizer with LR of 0.05, momentum of 0.9, and weight decay of $1e{-}4$. We train the models for 50 epochs for SplitCIFAR100 and MiniImageNet, and for 80 epochs for ImageNet32. The augmentation pipeline consists of random crop, random horizontal flip, and color jitter of 0.5. For the LP classifiers, we train each for 20 epochs, using AdamW [134] optimizer with a LR of $1e{-}3$ and a weight decay of $5e{-}4$.

For the SplitCIFAR10 2-task sequence from [1] we use the hyper-parameters and training conditions mentioned in [1] for both VGG [137] and ResNet [4] architectures. For the LP classifiers, we train each for 70 epochs, using AdamW [134] optimizer with a LR of $1e{-}3$ and a weight decay of $5e{-}4$. For LP training we use data augmentation in all cases, except the SplitCIFAR10 to obtain the most accurate measure of the model's ability to linearly separate the data of interest. For SplitCIFAR10 2-task we choose not to use any data augmentation because the source result ([1]) does not rely on data augmentation for training. We opt not to use data augmentation in our LP evaluation for a fair comparison.

5.5 Summary

We have highlighted the importance of evaluating representations and not just task accuracy in CL. Our results suggest a) representation forgetting under naive finetuning in supervised settings is not as catastrophic as other metrics suggest b) We demonstrate that without evaluation of features, the effects of model size on forgetting and representation learning will be misinterpreted. c) We show that the self-supervised SimCLR loss and supervised SupCon loss have lesser representation of forgetting in long task sequences, maintaining or increasing performance on early tasks. These results open up potential new directions for approaches in continual learning. One such direction of using memories that are not available to the learner during training is evaluated with promising initial results.

Limitations and Future Work Our work focuses on comparing linear probe performance as a proxy of knowledge retained from past tasks. However, task performance may not be the only criterion to fully evaluate knowledge retention related to past data. Another limitation of our work is that it currently focuses on

the task-incremental setting and does not consider the important class-incremental setting, a subject for future studies. Finally, though our work studies a diverse task sequence ImageNet $\rightarrow$ Scenes $\rightarrow$ CUB $\rightarrow$ Flowers, to fully understand the behavior of representation forgetting, results over more distant tasks may be needed (e.g. ImageNet $\rightarrow$ Sketch Images [138, 139]).

Chapter 6

Replay-Free Continual Learning with Evolving Class Prototypes

This work was the fruit of collaboration with MohammadReza Davari, Sudhir Mudur, Rahaf Aljundi, and Eugene Belilovsky. Currently in submission as a conference paper [140].

6.1 Introduction

Continual Learning (CL) aims to continuously acquire knowledge from an ever-changing stream of data. The goal of the learner is to continuously incorporate new information from the data stream while retaining previously acquired knowledge. Performance decay of the system on the older samples due to the loss of previously acquired knowledge is referred to as catastrophic forgetting [7], which represents a great challenge in CL. Thus CL algorithms are typically designed to control for catastrophic forgetting while observing additional restrictions such as memory and computation constraints.

Some of the early work in modern continual learning such as LwF [2] and EWC [117] propose solutions that do not require storage of past data. However, in complex settings with long task sequences, these techniques tend to under-perform by a wide margin compared to the idealized joint training [75]. Many recent high-performing approaches in this area maintain existing samples in some form of buffer, allowing them to be reused for distillation [42], replay [75, 83], or as part of gradient alignment constraints [40, 60]. These approaches have been shown to be more efficient and have become a predominant approach for many state-of-the-art continual learning systems [3]. On the other hand in many cases when training on a new task it may

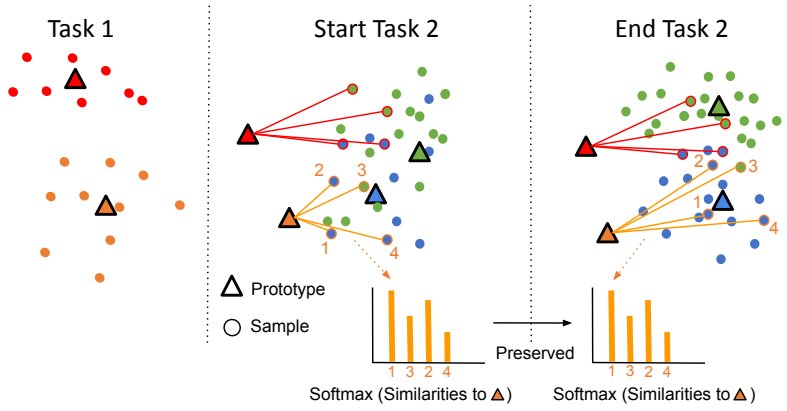


Figure 26: Illustration of our Prototype-Sample Relation Distillation (PRD). For each prior task prototype, we preserve the relative ordering of samples in the mini-batch. This gives flexibility for representations to adapt to new tasks while maintaining relevant positions of past prototypes. Illustrated for the prototype in orange 4 samples in the minibatch are ranked 1 through 4 based on similarity. PRD attempts to preserve this ranking while learning the new task.

be prohibited to store prior data. For example, prior task data may be sensitive (e.g. medical data) or it may consist of proprietary data that is not aimed for release. Moreover, methods relying on prior task data tend to grow the storage with the number of tasks [40, 83], which can be prohibitive under severe storage constraints. Thus developing methods that can match or exceed the efficiency of data-storage-based methods is of great importance.

Recently [107] observed that for many continual learning tasks, the representational power of deep networks trained with naive fine-tuning can remain remarkably efficient for representing both new and old task data. In particular, it was observed that when performing continual learning with the Supervised Contrastive Loss [141] and no CL constraints, the efficiency of representations on old task data tends to match that of complex CL data-storage methods. These observations relied on an oracle measure of the deep representations and did not provide a practical solution. In order to link the powerful representation learning to the effective prediction of prior class data we can consider alternatives for making the final prediction. An approach previously taken in the continual learning literature is to use the notion of class prototypes [142], vector representations whose similarity to new sample representation can give predictions of the target class. If we take the observation that representations of old classes are already well separated [107] then an efficient continual learner can be obtained by simply maintaining correct estimates of the class prototypes for past classes.

In this work we propose an effective way to maintain relevant class prototype positions based on avoiding changes in the relative similarity of new task data and old class prototypes. This allows us to leverage the powerful supervised contrastive loss for learning representations while maintaining a clear link to prototypes capable of providing predictions for past tasks. Our approach is illustrated in Fig. 26. Here each prototype is compared to all samples within a mini-batch. Our proposed method, Prototype-Sample Relation Distillation (PRD), maintains the relative relation of each prototype, specifically constraining the softmax distribution over samples in the mini-batch to not change. This effectively allows representations to adapt to new classes, while keeping prototypes from old classes relevant. We now summarize our *overall contributions* in this work:

- We propose a *novel* CL method, PRD, that does not rely on prior data storage during training or inference.
- In a variety of challenging settings (task and class-incremental), datasets (SplitMiniImagenet [143], SplitCIFAR100 [128], Imagenet-32 [107]), and task sequence lengths (20 to 200), we demonstrate that PRD leads to large improvements over both replay-based and replay-free methods.
- Throughout several experiments, we demonstrate that our method not only achieves strong control of forgetting previously observed tasks but also leads to improved plasticity in learning new tasks.

In the following section, Sec. 6.2, we summarize the related work, and then describe the essence of our proposed solution, Sec. 6.3. We demonstrate the effectiveness of our approach in Sec. 6.4.

6.2 Related Work

The primarily goal of many CL methods is to mitigate the catastrophic forgetting phenomenon, while optimizing the forward and backward knowledge transfer between tasks is seen as secondary objective. One branch of algorithms address the issue of catastrophic forgetting by modifying and growing the model architecture as new tasks are observed [16,67,115,116]. Under the fixed architecture constraints, the algorithms can be divided into two categories. The first and more popular branch is the rehearsal methods. These methods store and re-use samples of the past tasks while observing the new ones [60, 75]. The second family of approaches is the regularization based methods. These methods preserve the previously learned information by imposing

penalty terms on the objective of the new tasks, including popular methods such as LwF [2] and EWC [43, 117], where the former imposes a knowledge distillation penalty [38] on the objective of the newly observed task and the latter a quadratic penalty based on Fisher information matrix [121].

Recently, several works have considered the use of SupCon loss [141] in continual learning [71, 101, 144]. These studies have been largely focused on the combination of SupCon loss [141] with replay buffers in the online setting, and do not consider the notion of class prototypes in the replay-free setting. [107] demonstrated that the use of the SupCon loss in the offline setting yields more effective "representation forgetting" (forgetting as measured by an oracle training of a linear probe). However, a direct application of this observation of the SupCon loss was not proposed in this prior work.

[145] proposed a prototype based evolution strategy that continually updated prototypes using a momentum update combined with taking the mean of stored exemplars. Contrary to the present work this method focused on the online setting and leveraging stored data. Furthermore, it did not exploit the efficient and stable representation properties of contrastive supervised learning.

Knowledge distillation [38], where a student network attempts to mimic the behavior of a teacher network is a popular technique in deep representation learning [38, 146, 147], often used to reduce model size. Classical distillation techniques have been applied in various contexts in CL. [42, 148] utilized a distillation loss alongside replayed examples, constraining the current model to give similar outputs. [149] recently proposed to use a triplet loss alongside a contrastive distillation term. Here samples are constrained to have similar distances under current and previous models. By contrast we apply a relation distillation that constrains the relative distances of old class prototypes to samples to be preserved.

Another application of classical distillation techniques closely related to our work [150] proposed a method that does not require storage of prior task data. This approach relies on constraining the distance between embeddings of the old and new models combined with a cross entropy term. The constraint here can be analogous to a traditional distillation term, while our approach focuses on relation distillation to prototypes. [150] also utilizes a self-supervised learning objective based on rotation of images to enhance the representation learning similar to [151].

Relation distillation has recently been used in teacher-student methods [152]. Unlike conventional knowledge distillation techniques that attempt to make student network representations similar to teacher networks, relation distillation maintains

relative distances between a set of points. In the present work we apply a related idea in the context of continual learning, maintaining relative relationships between prototypes and current task samples.

6.3 Methodology

We consider a general continual learning setting where a learner is faced with a possibly never-ending stream of data divided into separate training sessions. At each session S_t, a set of data $\mathbf{X}^t$ and their respective labels $\mathbf{Y}^t$ are drawn from a distribution D_t characterized by $P(\mathbf{X}, \mathbf{Y}|T = t)$. When learning new sessions, it is assumed that access to the samples form previous sessions is restricted. This definition covers both task-incremental settings where $(\mathbf{X}^t, \mathbf{Y}^t)$ represent a separate task, class-incremental scenario where changes in $P(\mathbf{X})$ induces a shift on $P(\mathbf{Y})$, and the domain-incremental learning where changes in $P(\mathbf{X})$ does not affect $P(\mathbf{Y})$. We consider a neural network composed of an encoder f that maps an input sample x to its features representation $f(x) \in \mathbb{R}^d$ and a projection head g that projects the features onto another latent space $g \circ f(x) \in \mathbb{R}^k$ where $k < d$. Our goal is to minimize the objective loss $\mathcal{L}_t$ on the new session data while not increasing the objective loss of the previously learned sessions $\mathcal{L}_{i\ \forall i<t}$.

A common approach to control the loss of the previously encountered sessions is to use a buffer of stored samples and reuse them upon encountering new sessions. In our approach we do not require access to the past data, instead, we approximate the behavior of the previously seen classes of objects via a set of prototypes. In our approach upon visiting a new session, we employ a novel distillation term to approximate the now-inaccessible loss of the previous sessions and set to restrict this surrogate loss in order to control the loss of the previously seen sessions. In the following sections, we introduce the different parts of the objective function used to optimize the model at each step.

Supervised Contrastive Learning Supervised Contrastive Learning [141] is a powerful representation learning method observed to be useful in many downstream tasks. [107] employed Supervised Contrastive training for continual learning and showed that the learned representations are less prone to forgetting compared to that learned with Cross Entropy loss (CE). In this work, we build on this observation and propose a solution to jointly train the representation and classification head in an incremental fashion. In order to optimize the representation for the task being

learned, we apply a supervised contrastive loss on the incoming data.

$$\mathcal{L}_{SC}(\mathbf{X}) = -\sum_{\mathbf{x}_i \in \mathbf{X}} \frac{1}{|A(i)|} \mathcal{L}_{SC}(\mathbf{x}_i) \tag{21}$$

Where each sample's loss is given:

$$\mathcal{L}_{SC}(\mathbf{x}_i) = \sum_{\mathbf{x}_p \in A(i)} \log \frac{h(g \circ f(\mathbf{x}_p), g \circ f(\mathbf{x}_i))}{\sum_{\mathbf{x}_a \in \mathbf{X}/x_i} h(g \circ f(\mathbf{x}_a), g \circ f(\mathbf{x}_i))} \tag{22}$$

Where $h(a, b) = \exp(\texttt{sim}(a, b)/\tau)$ and $\texttt{sim}(a, b) = \frac{a^T b}{\|a\|\|b\|}$. Here $A(i)$ represents the set of samples that form positive pairs with x_i i.e. augmented views of $\mathbf{x}_i$ and other samples of the same class $\{\mathbf{x}_j | y_j = y_i\}$. Note that this loss is composed of tightness terms between positive pairs and contrast terms with negative pairs [153].

Prototype Learning without Contrasts In order to easily link the discriminative representations learned by optimizing $\mathcal{L}_{SC}(\mathbf{X})$ to a final class level prediction we consider the notion of class prototypes [71, 142], which allow us to score a sample's representation with respect to each class. A simple solution for learning the class prototypes is to apply the Softmax in combination with the Cross Entropy loss, for a given sample yielding

$$-\texttt{sim}(\mathbf{p}, f_\theta(\mathbf{x}_i)) + \log\Big(\sum_{\mathbf{p}_k \in \mathbf{P}} h(\mathbf{p}_k, f_\theta(\mathbf{x}_i))\Big) \tag{23}$$

However, it has been shown that in the class-incremental setting, the softmax combined with cross-entropy produces a large interference with previously learned classes due to terms that suppress previous classes logits [71, 154]. Here, we propose instead to learn class prototypes that are representatives of each class samples using only the first term in this loss, referred to as the "tightness" term [153]. For each class c we initialize a random prototype $\mathbf{p}_c \in \mathbb{R}^d$. We want to optimize these prototypes to be representatives of current classes samples without introducing any suppression to prototypes of previous classes. To achieve this we use a loss term considering only positive pairs of class samples and their corresponding prototypes where we aim at maximizing the similarity of these pairs:

$$\mathcal{L}_p(\mathbf{X}) = -\frac{1}{|\mathbf{X}|} \sum_{\mathbf{x}_i, y_i \in \mathbf{X}, \mathbf{Y}} \texttt{sim}(\mathbf{p}_{y_i}, \texttt{sg}[f_\theta(\mathbf{x}_i)]) \tag{24}$$

Here *sg* denotes the stop gradient operations. The suggested loss contains only a tightness term, *i.e.*, contrast-free, which doesn't have a direct effect on previous

classes prototypes. From this loss term, we aim to only optimize the prototypes and not to change the samples representations as this is taken care of by equation 21. Note that contrary to [71] which also uses prototype-based learning, we do not include any contrastive terms for the prototype learning, the learning of class separations being left to $\mathcal{L}_{SC}$. Note that we utilize the stop-gradient operation so that the learning of the prototypes does not interfere with the representation learning or previous prototypes.

Once prototypes are obtained we can now directly perform predictions at test time by using the similarity of the sample representation and the set of prototypes to decide on the nearest class prototype.

Prototypes-Samples Similarity Distillation Our prototypes are learned in isolation for each task. However, as we update our feature extractor using the supervised contrastive loss Eq. equation 21 prototypes of previous task classes will become outdated leading to the forgetting of previously learned classes. As shown in [107] this forgetting may correspond simply to movement in the decision boundary, despite classes still being well separated. To update old prototypes as we update our representation, we propose a similarity distillation term using new classes data as a proxy for old data. Before the start of a new training session, we compute the prototypes' similarities to each sample of new classes. During the new training session, we propose to minimize the KL divergence between the similarities distribution of prototypes to minibatch samples, enforcing current similarities to be similar to previous similarities.

Consider the current model and set of prototypes for previous classes $f_{\theta_t}, \mathbf{P}_o^t$ along with their corresponding model and prototype from the end of the previous task $f_{\theta_{t-1}}, \mathbf{P}_o^{t-1}$. For an incoming mini-batch $\mathbf{X}$ and a corresponding prototype we can consider the softmax output $\mathcal{P}_t(\mathbf{p}_k^t, \mathbf{X})$, where the i^{th} entry is given:

$$\mathcal{P}_t(\mathbf{p}_k^t, \mathbf{X})_i = \frac{h(\mathbf{p}_k^t, f_{\theta_t}(\mathbf{x}_i))}{\sum_{\mathbf{x_j} \in \mathbf{X}} h(\mathbf{p}_k^t, f_{\theta_t}(\mathbf{x}_j))} \tag{25}$$

Denoting for shorthand $\mathcal{P}_t(\mathbf{p}_k^t, \mathbf{X})$ as $\mathcal{P}_t(k)$ we can now construct a relation distillation term as the KL-divergence between prototype-samples similarity distribution estimated with the model at session $t-1$ and during the current session t.

$$\mathcal{L}_d(\mathbf{P}) = \sum_{\mathbf{p}_k \in \mathbf{P}_o} KL\Big(\mathcal{P}_t(k) \,||\, \mathcal{P}_{t-1}(k)\Big) \tag{26}$$

Note that this is distinct from distillation approaches where we compute the

similarities for each sample over existing classes. As illustrated in Fig.26 the relative positions of samples to the prototypes are encouraged to remain the same by our loss. This results in flexibility in the representations in order to adapt to new classes while keeping the relative distances of many samples to the prototype as similar as possible. Our overall training objective is thus given as a combination of these three terms:

$$\mathcal{L}(\mathbf{X}) = \mathcal{L}_{sc}(\mathbf{X}) + \alpha\mathcal{L}_{p}(\mathbf{X}, P_c) + \beta\mathcal{L}_{d}(\mathbf{X}, P_o)$$

6.4 Experiments

In this section, we evaluate our proposed method on a wide range of challenging CL settings. In Sec. 6.4.1 we focus on the *task-incremental* (multi-head) setting, where we compare our method with other replay-free methods. Sec. 6.4.2 is dedicated to *class-incremental* setting, where the shared output layer poses an enormous challenge that drives most work to employ a replay buffer, unlike our replay-free solution. The evaluations are based on *average observed accuracy*. Specifically, we measure observed accuracy, A_{ij}, as the accuracy of the model after step i on the test data of task j. Similarly, the average observed accuracy at the end of the sequence is $\frac{1}{T}\sum_{t\in T} A_{T,t}$ as used in *e.g.* [2].

Datasets We use Split-CIFAR100 [128], Split-MiniImageNet, and ImageNet32 [107, 129] as the benchmarks for both multi-head and single-head settings. Split-CIFAR100 [128] comprises 20 tasks, each containing a disjoint set of 5 labels. The classes splits are constructed as in [75]. All CIFAR experiments process 32×32 images. Split-MiniImageNet divides the MiniImagenet dataset into 20 disjoint tasks of 5 labels each. Images are 84×84. ImageNet32 [129] is a downsampled (32×32) version of the entire ImageNet [5] dataset split into 200 tasks of 5 classes each. We use ImageNet32 in order to compare methods' performance in very long sequence scenarios.

Baselines Although our proposed method does not use any replay buffer, we consider in our evaluation both *replay-free* and *replay-based* methods, as replay-based have been shown to outperform other approaches in the continual learning setting [42, 48, 75, 98]. We consider the following replay-free methods in our evaluations:

LwF [2]: knowledge distillation based on the softmax probabilities and current task data is used to limit forgetting.

EWC [120]: estimate an importance value for each parameter in the network and penalize changes on parameters deemed important for previous tasks.

SPB [150]: A recent method that also utilizes contrastive learning and addresses the setting without replay data. We were unable to effectively reproduce this approach to the accuracies described as code was not provided. However, we compare our approach directly to the reported results in the setting studied in the original work [150] in Sec. 6.4.2.

iid: The learner is trained with the same number of epochs as other baselines on the whole data, in a single task containing all the classes.

The incorporated replay-based baselines are as follows:

ER [75]: Experience Replay with a buffer of a fixed size. In our experiments, we used buffer sizes of 5, 20, and 50 samples per class based on the evaluation setting. Note this is a very strong baseline that exceeds most methods, particularly with large buffers (50 samples) [107].

iCaRL [42]: A distillation loss alongside binary cross-entropy loss is used during training. Samples are classified based on the closest class prototypes.

ER-AML [71]: Utilizes SupCon loss, alongside a replay buffer, to reduce the representation drift of previously observed classes.

ER-ACE [71]: Similar to ER-AML, ER-ACE intends to reduce the representation drift of old classes by introducing a modified version of the standard softmax-cross-entropy.

Hyperparameter selection For each method, optimal hyperparameters were selected via a grid search performed on the validation set. The selection process was done on a per-dataset basis, that is we picked the configuration which maximized the accuracy averaged over different settings. We found that for our method, the same hyperparameter configuration worked well across all settings and datasets. All necessary details to reproduce our experiments can be found in the supplementary materials.

6.4.1 Evaluations on Task-Incremental Setting

We evaluate on Split-CIFAR100, Split-MiniImagenet, and ImageNet32 using the protocol from [48] with 100 epochs of training on each task. Note that for each evaluation point we report the mean and standard error over 3 runs.

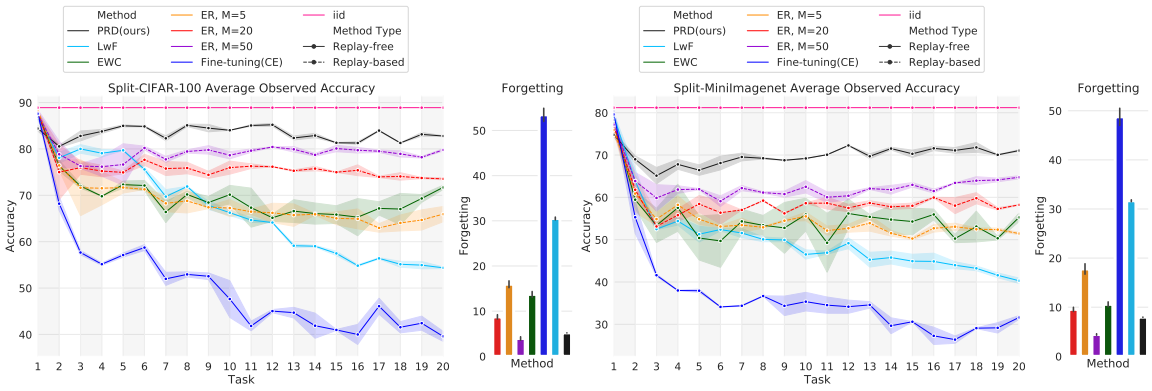


Figure 27: *Task-incremental* accuracy on 20-Task Split-CIFAR100(left) and Split-MiniImageNet(right). We observe that PRD widely outperforms other baselines without storing any previous task data, and as well exceeds the performance of ER with a very large buffer.

Split-CIFAR100 and Split-MiniImageNet We consider Split-CIFAR100 and Split-MiniImageNet with 20 tasks of 5 classes each. The results can be found in Fig. 27, for Split-CIFAR100 and Split-MiniImageNet, using different buffer sizes for ER. In this setting, we can observe that our proposed method *consistently outperforms* other methods by a significant margin. Even though our method does not utilize previous tasks' data in any form, it still outperforms ER with 50 replay samples per class, nearly closing the gap with the oracle *iid* setting. From Fig. 27, we can observe that during the whole continual sequence of tasks, the average accuracy of our method on the observed classes remains relatively similar and even increases at several points during the sequence, *e.g.* 12^{th} task, suggesting a good trade-off between stability and plasticity of the model.

Although our method, in terms of average observed accuracy, outperforms other baselines with a considerable margin, a little higher forgetting rate can be observed compared to the strong replay-based baseline, *i.e.* ER with 50 replay samples. In Sec. 6.4.3, we show that our proposed method, in terms of plasticity, *i.e.* ability to learn new tasks, is comparable to naive fine-tuning, which is the upper bound among existing CL methods due to the absence of constraints on preserving previous tasks (i.e., lacking stability). This observation suggests that our method is able to preserve previous tasks' information without losing the ability to learn new tasks.

ImageNet32 - Long Task Sequence We now consider a longer sequence than typically studied which allows us to observe whether the trends we have seen so far continue to hold. Using Imagenet32 we construct 200 tasks of 5 classes each. Fig. 28

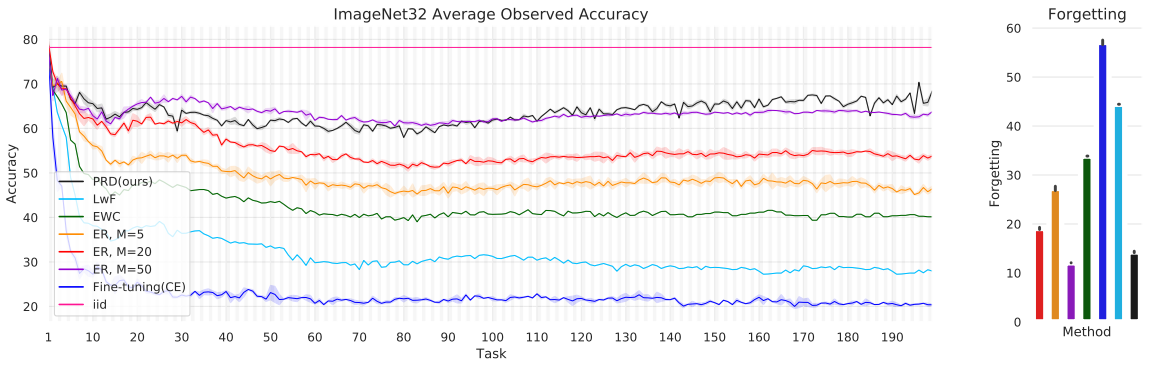


Figure 28: *Task-incremental* accuracy on 200-Task ImageNet32. On this long sequence, PRD matches a baseline with a large replay buffer. Although other methods degrade overtime, the average accuracy of PRD improves due to the cumulative effect of maintaining better plasticity.

shows the average observed accuracy throughout the whole 200 tasks sequence. We see that in a very long sequence of tasks, the previously established observations about our method holds. Specifically, we see that as the model reaches the later stages of the sequence, our method outperforms the competitive baseline of ER with 50 replay samples *without* utilizing previous tasks' data in any form. Note that the number of stored data points leveraged by ER increases as we proceed in the sequence. Furthermore, we observe as in the previous section that the average observed accuracy of our proposed method not only stays relatively the same during the beginning but also starts increasing as the model reaches the middle of the sequence, *i.e.* the 90^{th} task. This observation suggests that our method is able to efficiently learn new tasks features while preserving the previous tasks' information.

6.4.2 Evaluations on Class-Incremental Setting

In addition to the experiments in the *task-incremental* setting, to further verify the effectiveness of our method in mitigating representation forgetting with no access to prior task data, we also evaluate on the more challenging *class-incremental* setting where we examine the ability to incrementally learn a shared classifier. Note that in all results each method is run 3 times, and we report the mean and standard error.

Split-CIFAR100 and Split-MiniImageNet Fig. 29 shows the average observed *class-incremental* accuracy of the model over the 20 task sequence of Split-CIFAR100 and Split-MiniImageNet. Note that the *replay-based* methods are plotted in dashed lines. We can see that our method, with no access to previous tasks data, not only

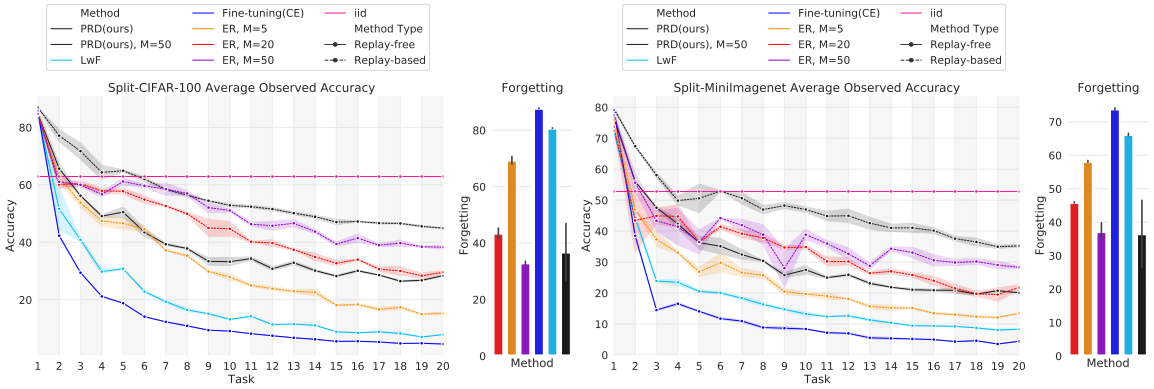


Figure 29: *Class-incremental* accuracy on 20-Task Split-CIFAR100(left) and Split-MiniImageNet(right). We observe that PRD outperforms not only other *replay-free* baselines but also ER, M=5, and is on par with ER, M=20, without storing any data. We also observe that with additional replay samples, PRD M=50 outperforms ER M=50 with the same number of replay samples.

outperforms other *replay-free* methods but also beats ER with 5 replay samples and is on par with ER with 20 replay samples per class. From Fig. 29, we can observe that the average accuracy of our method drops initially, probably due to the drift of the old tasks' prototypical features, but stays relatively the same from the middle of the sequence. This observation also suggests that in longer tasks sequences the learned prototypical features of old classes remain useful, even in the absence of any replay data.

In the following section, Sec. 6.4.2, we perform a thorough experiment on the effect of different replay buffer sizes, showing that our method beats the state-of-the-art replay-based methods with fewer stored samples.

Leveraging stored samples Our method targets incremental learning in scenarios where no stored samples are allowed. However, in this section, we investigate if our can method benefits from the availability of a few stored samples.

When utilizing replay data we follow the standard approach of ER-based methods, sampling half the training data of the mini-batch from the previous data. The subsequent optimization problem is kept the same. Note that now the relation distillation will also see data from past tasks that directly correspond to the prototypes. Sec. 6.4.2 compares our method with different buffer sizes to other replay-based methods. It can be seen that our method can successfully leverage the available data and further improve the performance achieving high gains over the state-of-the-art in low buffer regime. This suggests our method is highly effective in both limited

Method	Split-CIFAR100, K = 20				Split-MiniImageNet, K = 20			
	$M = 0$	$M = 5$	$M = 20$	$M = 50$	$M = 0$	$M = 5$	$M = 20$	$M = 50$
iid	65.3	65.3	65.3	65.3	54.5	54.5	54.5	54.5
Fine-tuning	4.6	4.6	4.6	4.6	4.4	4.4	4.4	4.4
ER [75]	-	$15.2_{\pm 0.7}$	$29.6_{\pm 0.8}$	$38.3_{\pm 0.9}$	-	$13.4_{\pm 0.2}$	$21.7_{\pm 0.8}$	$28.7_{\pm 0.5}$
iCaRL [42]	-	$19.8_{\pm 0.5}$	$28.6_{\pm 0.7}$	$32.9_{\pm 0.5}$	-	$16.2_{\pm 0.1}$	$22.8_{\pm 0.3}$	$26.1_{\pm 0.2}$
ER-AML [71]	-	$21.4_{\pm 0.8}$	$35.3_{\pm 0.6}$	$42.4_{\pm 0.8}$	-	$17.1_{\pm 0.3}$	$26.3_{\pm 0.7}$	$32.3_{\pm 0.2}$
ER-ACE [71]	-	$22.8_{\pm 0.5}$	$35.7_{\pm 0.2}$	$43.3_{\pm 0.2}$	-	$18.8_{\pm 0.1}$	$27.1_{\pm 0.5}$	$34.2_{\pm 0.5}$
PRD(Ours)	$\mathbf{27.8}_{\pm 0.2}$	$\mathbf{32.0}_{\pm 0.4}$	$\mathbf{39.5}_{\pm 0.4}$	$\mathbf{45.1}_{\pm 0.5}$	$\mathbf{20.0}_{\pm 0.1}$	$\mathbf{25.7}_{\pm 0.3}$	$\mathbf{31.3}_{\pm 0.5}$	$\mathbf{35.8}_{\pm 0.4}$

Table 18: *Class-incremental* results on 20-Task Split-CIFAR100 and Split-MiniImageNet datasets using different buffer sizes. We observe even with no replay samples (M=0) PRD outperforms all of the replay-based baselines with 5 replay samples. With a small number of replay samples, *e.g.* M=5, PRD widely outperforms other replay-based methods, suggesting the ability of our method to utilize replay samples while maintaining a good performance with no access to prior data.

Method	Split-CIFAR100		ImageNet-Sub	
	K=6	K=11	K=6	K=11
iid	73.4	73.2	82.0	82.7
Fine-tuning	22.3	12.6	23.6	13.2
LwF-E [2, 155]	57.0	56.8	65.5	65.6
EWC-E [120, 155]	56.3	55.4	65.2	64.1
MAS-E [59, 155]	56.9	56.6	65.8	65.8
SDC [155]	57.1	56.8	65.6	65.7
SPB [150]	60.9	60.4	68.7	67.2
PRD (Ours)	**64.3**	**63.7**	**71.8**	**70.3**

Table 19: Pre-trained Initialization. We report average *cumulative* incremental accuracies over all tasks are reported. PRD exceeds recent proposals in this challenging setting.

and no replay data settings.

Pre-trained Initialization To further measure the class-incremental performance of our method and allow direct comparison to [150], we also evaluate our method on Split-CIFAR100 [128] and ImageNet-Subset [42, 156] using the protocol and constraints from [150]. In these settings, half the classes are used for an initial pre-training phase. ImageNet-Subset contains 100 classes randomly sampled from ImageNet. Following [150], we randomly select the first 50 classes as the 1-*st* phase

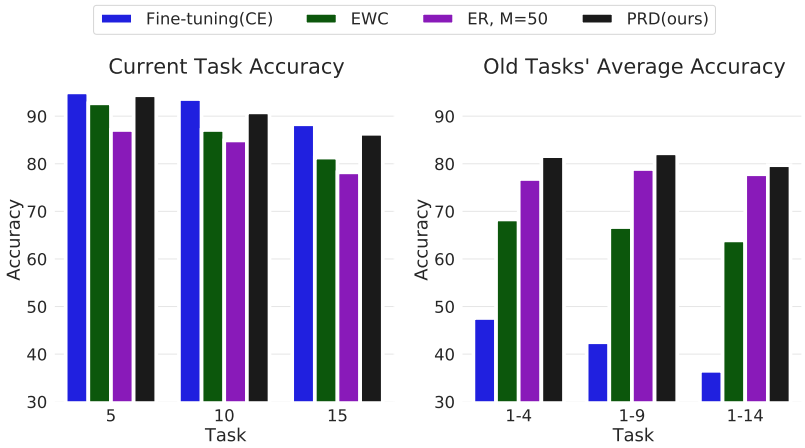


Figure 30: *Task-incremental* Split-CIFAR100. Accuracy on the current task(left) and average accuracy over previous tasks(right). We observe that PRD performs well on the current task while having low forgetting.

and evenly split the remaining 50 classes for K-1 phases. Similar to [150], for this experiment, we evaluate our models with K=6 and 11 phases on both Split-CIFAR100 and ImageNet-Subset datasets, *i.e.*, after the 1^{st} phase, we incrementally add 5 or 10 new classes at each phase. Following [150], we report the average *cumulative* incremental accuracy over all phases. All results are averaged over three runs.

Tab. 19 shows average *cumulative* incremental accuracies (as used in [150]) over all phases on Split-CIFAR100 and ImageNet-Subset. We observe that our method exceeds the recent proposal of [150] in this setting as well as beating strong baselines such as SDC. Note that [150] also applied a self-supervised objective which we do not include as we were unable to obtain source code for these experiments, and this was a complementary approach that can enhance our method as well.

6.4.3 Analysis and Ablations

PRD Balances Plasticity and Stability A continual learner should be able to easily integrate new knowledge by learning new tasks (plasticity) while benefiting from prior knowledge to improve performance on the current task (forward transfer). Continual learning methods are often characterized by a trade-off in plasticity and stability. Stability refers to the ability to retain the knowledge of prior tasks [157], often measured by the forgetting metric. We have thus far observed in our experiments that PRD has relatively low forgetting, for example in Fig. 27, it is shown for CIFAR-100 it has the lowest forgetting, only slightly improved on by the ER-M50 a baseline with a large replay buffer. Both ER-M50 and PRD have good stability, but their

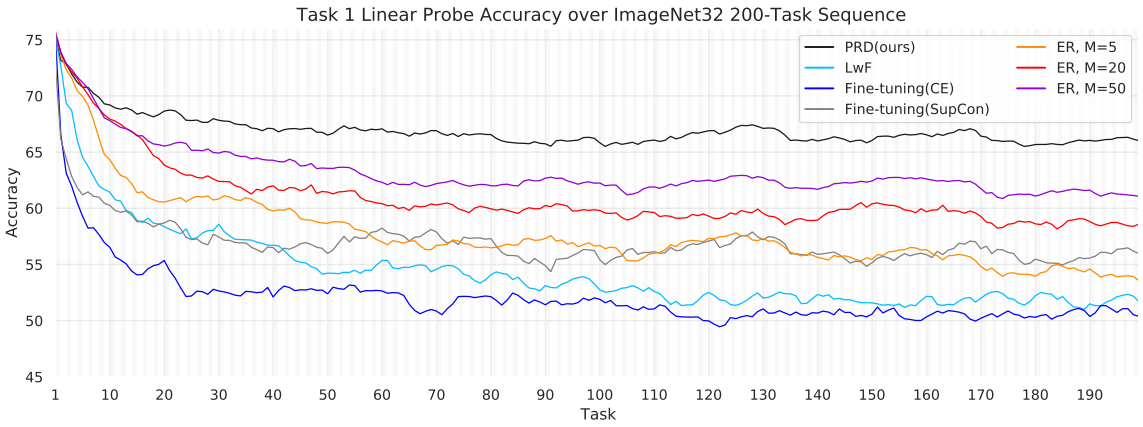


Figure 31: Task 1 LP accuracy over 200-Task ImageNet32. We compare Linear probe accuracy for Tasks 1 data over the whole sequence. We can observe that during a long sequence, the performance of our method, *i.e.* PRD, not only stays relatively flat but also increase at some points in the later stages of the sequence, suggesting its ability to preserve the information of observed tasks.

plasticity can be difficult to gauge in long tasks sequences directly from observed accuracy. For example a task can be very poorly learned during a session but learned later on thanks to the replay buffer. We thus directly compare the current task performance separately from the old task performance corresponding to PRD, ER M=50, and EWC on task-incremental CIFAR100, corresponding to the results in Fig. 27. The results are shown for tasks 5,10, and 15 in Fig. 30.

We first observe that all methods progressively degrade in current task accuracy. Since tasks are sampled uniformly from the set of possible tasks we can assume this corresponds to a gradual reduction in plasticity. This is consistent with many previous observations of continual learning systems [157, 158]. On the other hand, we observe that compared to other strong baselines like EWC and ER, M=50, the current task accuracy of PRD is substantially higher, while the old task accuracy is as well, being largely maintained as training progresses. Thus PRD provides a strong tradeoff in plasticity and stability. If we observe the behavior of ER, M=50, we see that its old task accuracy is sometimes increasing (for example at task 10). Overall, we can state that models trained by our method, PRD, exhibit plasticity close to constraint-free fine-tuning while showing the best stability.

Representation Forgetting Following [107], we evaluate the representation forgetting of our method against other baselines with a *Linear Probe*, denoted as LP in the rest of this section. Similar to the definition of observed accuracy in Sec. 6.4,

Distillation Coefficient(β)	Dataset Split-CIFAR100 (K=20)	Split-MiniImageNet (K=20)	ImageNet32 (K=200)
$\beta = 0.$	$39.4_{\pm 1.5}$	$31.2_{\pm 0.9}$	$21.3_{\pm 0.8}$
$\beta = 1.$	$80.0_{\pm 0.5}$	$59.3_{\pm 0.3}$	$55.4_{\pm 0.4}$
$\beta = 2.$	$82.1_{\pm 0.3}$	$63.7_{\pm 0.3}$	$59.2_{\pm 0.4}$
$\beta = 4.$	$\mathbf{83.5}_{\pm 0.4}$	$\underline{68.3}_{\pm 0.4}$	$62.7_{\pm 0.2}$
$\beta = 8.$	$\underline{83.1}_{\pm 0.4}$	$\mathbf{70.9}_{\pm 0.5}$	$\underline{65.1}_{\pm 0.3}$
$\beta = 16.$	$82.7_{\pm 0.2}$	$67.2_{\pm 0.5}$	$\mathbf{67.5}_{\pm 0.2}$

Table 20: Ablation study on the effect of relation distillation coefficient, β in Eq. equation 6.3. The reported numbers are *Task-incremental* average observed accuracy. We can observe that $\beta = 0$, having no distillation, results in very low average accuracy over all tasks. On the other hand, when the sequence is very long, *e.g.* 200 task ImageNet32, a higher coefficient value for the distillation loss results in better overall average accuracy.

we can measure the LP accuracy for each step i and task j as well as the average LP accuracy.

Similar to [107], we construct 200 tasks of 5 classes using ImageNet32 dataset in the task-incremental setting. Fig. 31 shows the performance of the model on the first task throughout the whole continual sequence. We observe that although the naive SupCon exceeds replay with M=5 in terms of representation forgetting on the first task, PRD provides a very substantial improvement. This suggests that we not only benefit from stabilizing the prototypes but the representation itself greatly benefits from PRD, avoiding forgetting at the representation level.

Ablation on Prototypes-Samples Similarity Distillation As discussed in Sec. 6.4.3, continual learning methods are characterized by a trade-off in plasticity and stability. Stability refers to the ability to retain the knowledge of prior tasks [157], often measured by the forgetting metric. According to our observations from Sec. 6.4.3, one can observe the PRD has relatively low forgetting while maintaining high plasticity in learning new tasks. PRD controls the stability-plasticity trade-off mostly using the coefficient for *prototype-sample relation distillation* loss. Here we do an ablation on the effect of our prototype-sample relation distillation loss in three datasets, Split-CIFAR100, Split-MiniImageNet, and ImageNet32. Tab. 20 presents the performance of our method with different *coefficient* values(β) for our prototype-sample relation distillation loss.

We can observe that using a coefficient value of 0 ($\beta = 0$), *i.e.* having no relation

	Finetune	EWC [117]	LwF [2]	PRD (ours)
AMCA	40.5	62.5	63.7	**65.1**

Table 21: Domain-incremental setting using the CLAD-C dataset [3]. All methods use a ResNet-50 [4] architecture, pre-trained on ImageNet [5], and use a batch size of 32. Our results highlight the versatility of our method and its applicability to real-life continual learning scenarios. Moreover, it suggests that our method is cable of performing under severe class imbalance and drastic distribution shifts, without having access to past data.

distillation loss, Eq. equation 6.3, results in very low average accuracy for all of the three datasets. This observation shows the importance of relation distillation loss in remembering old tasks' information. Further, we can observe using different values of β, with shorter task sequences like 20 task Split-CIFAR100, does not affect the overall average performance of the model across all tasks. On the other hand, with long task sequences, *e.g.* 200 task ImageNet32, a higher coefficient value for the distillation loss results in less forgetting and better overall average accuracy.

6.4.4 Evaluations on Domain-Incremental Setting

We evaluate our approach using the CLAD-C dataset [3], under the conditions laid out in [3]. The dataset contains images recorded via dashcams over 3 days. The shift between night and day constitutes the task boundaries, hence overall we have 6 tasks. The objective of each tasks is to correctly classify an image into one of 6 possible classes of objects: 1. pedestrian 2. cyclist 3. car 4. truck 5. bus 6. tricycle. The dataset reflects the real-world distribution of these objects. Hence, certain classes are rarely observed (e.g. the tricycle class) and others are seen more frequently (e.g. the car class). As the night and day changes in the data stream and we are introduced to new tasks, the image distribution changes, sometimes so drastic that leads to the absence of a few classes. This fact, along with the in-task class imbalance of the data makes the CLAD-C dataset [3] a challenging, yet realistic, benchmark.

The training data contains of overall 22,249 objects distributed over 6 tasks. We report our results using the final Average Mean Class Accuracy (AMCA) on the test data which contains 69,881 objects spanning over both day and night. The AMCA for T tasks each containing C classes is given by:

$$\text{AMCA} = \frac{1}{|T||C|} \sum_{t \in T} \sum_{c \in C} A_c^t \tag{27}$$

where A_c^t is the accuracy of the class c for the task t. The results are given in Tab. 21. All methods in Tab. 21 use a ResNet-50 [4] architecture, pre-trained on ImageNet [5],

and use a batch size of 32. Our results highlight the versatility of our method and its applicability to real-life continual learning scenarios. Moreover, it suggests that our method is cable of performing under severe class imbalance and drastic distribution shifts, without having access to past data.

6.4.5 Training Details

In this section, we provide additional details regarding the baselines and hyperparameters. In all experiments, we leave the **batch size** and the **number of epochs** fixed at **128** and **100**. The model architecture (θ) is also kept constant, which is a regular ResNet-18 model, where the dimensions of the last linear layer change depending on the input height and width.

The augmentation pipeline is consistent across all experiments, consisting of the random crop, random horizontal flip, color jitter of 0.4, and random grayscale.

Hyperparameter Selection All results in the paper have been either implemented by us or adapted from [71], with the exception of SPB [150], where results were taken from the original paper since there was no public codebase for that baseline at the time of submission. For each method, a grid search was run on the possible hparams, which we detail below. In the following, we list the hyperparameters that we included in our grid search. The best values for each parameter are underlined.

PRD(ours)

- LR: [<u>0.01</u>, 0.005, 0.001]
- SupCon Temperature: [<u>0.1</u>, 0.2, 0.3]
- Relation Distillation Coefficient(β): [1., 2., <u>4.</u>, 8., <u>16.</u>]
- Prototypes Coefficient(α): [1., <u>2.</u>, 4.]

EWC [120]:

- LR: [0.01, <u>0.005</u>, 0.001]
- Lambda Coefficient: [20, 50, 100, 200, 500, 1000]

LwF [2], **ER** [75], and **iCaRL** [42]:

- LR: [0.01, <u>0.005</u>, 0.001]

Similar to [71], for ER, rehearsal begins as soon as the buffer is not empty. Also when samples are being fetched from the buffer, we do not exclude classes from the current task.

ER-ACE [71]:

- LR: [0.01, 0.005, 0.001]

Following [71] implementation, for the masking loss, we simply use `logits.maskedfill(mask, -1e9)` to filter out classes that should not receive gradient.

ER-AML [71]:

- LR: [0.01, 0.005, 0.001]
- SupCon Temperature: [0.1, 0.2, 0.3]

6.5 Summary

We proposed a novel approach for Replay-Free Continual learning that effectively leverages relationship distillation alongside supervised contrastive learning. On a wide array of evaluations, our method is shown to provide good trade-offs in stability and plasticity, leading to large improvements over replay-free baselines and allows us to exceed performance of replay-based methods. Moreover, we showed that our method can effectively utilize additional replay samples, outperforming the state-of-the-art replay-based methods in the class-incremental setting. These observations open up potential new directions for approaches in replay-free continual learning.

Chapter 7

Conclusion

In this thesis, we addressed various challenges in online continual learning and replay-free continual learning settings. We proposed novel methods to handle these challenges and evaluated our proposed methods on various datasets with state-of-the-art metrics.

In Chapter 3, we showed that the standard loss function applied excessive pressure on old class representations in online continual learning settings. To address this challenge, we proposed two modifications of the loss function that treated incoming and replay data asymmetrically. Our proposed method achieved strong performance with minimal or no additional cost, and we raised the standard for high-quality evaluation in online continual learning by considering a wide number of baselines and metrics.

In Chapter 4, we introduced a new method to handle online one-class incremental learning without the need for negative contrasts. We demonstrated that our method outperformed strong baselines and was also applicable and highly competitive in traditional online continual learning settings. Our approach can also be adapted to supervised settings, particularly in continual classification. Future work can explore the possibility of applying our approach in the offline one-class incremental setting.

In Chapter 5, we emphasized the importance of evaluating representations and not just task accuracy in continual learning. Our results suggested that representation forgetting under naive finetuning in supervised settings was not as catastrophic as other metrics suggest. We also showed that the self-supervised SimCLR loss and supervised SupCon loss had lesser representation forgetting in long task sequences, maintaining or increasing performance on early tasks. Additionally, we evaluated a direction of using memories that are not available to the learner during training with promising initial results.

In Chapter 6, we proposed a novel approach for replay-free continual learning that leveraged relationship distillation alongside supervised contrastive learning. Our method provided good trade-offs in stability and plasticity, leading to large improvements over replay-free baselines and exceeded the performance of replay-based methods. Moreover, our method effectively utilized additional replay samples. Our observations opened up potential new directions for approaches in replay-free continual learning.

Overall, our proposed methods addressed various challenges in continual learning and replay-free continual learning settings, and our results showed that these methods outperformed state-of-the-art baselines on various datasets. Our findings provide new directions for future research in the field of continual learning.

References

[1] V. V. Ramasesh, E. Dyer, and M. Raghu, "Anatomy of catastrophic forgetting: Hidden representations and task semantics," *arXiv preprint arXiv:2007.07400*, 2020. xv, 16, 55, 57, 58, 59, 60, 67, 68, 72

[2] Z. Li and D. Hoiem, "Learning without forgetting," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 12, pp. 2935–2947, 2017. xv, 55, 59, 60, 61, 62, 63, 69, 70, 74, 77, 81, 82, 86, 90, 91

[3] E. Verwimp, K. Yang, S. Parisot, H. Lanqing, S. McDonagh, E. Pérez-Pellitero, M. De Lange, and T. Tuytelaars, "Clad: A realistic continual learning benchmark for autonomous driving," *arXiv preprint arXiv:2210.03482*, 2022. xvi, 74, 90

[4] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016. xvi, 71, 72, 90

[5] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255, Ieee, 2009. xvi, 20, 81, 90

[6] D. L. Silver, Q. Yang, and L. Li, "Lifelong machine learning systems: Beyond learning algorithms.," in *AAAI Spring Symposium: Lifelong Machine Learning*, pp. 49–55, Citeseer, 2013. 4

[7] M. McCloskey and N. J. Cohen, "Catastrophic interference in connectionist networks: The sequential learning problem," *Psychology of learning and motivation*, vol. 24, pp. 109–165, 1989. 5, 23, 53, 74

[8] R. Ratcliff, "Connectionist models of recognition memory: Constraints imposed by learning and forgetting functions," *Psychological review*, vol. 97, no. 2, pp. 285–308, 1990. 5

[9] J. Kolen and J. Pollack, "Back propagation is sensitive to initial conditions," *Advances in neural information processing systems*, vol. 3, 1990. 5

[10] J. K. Kruschke, "Alcove: an exemplar-based connectionist model of category learning.," *Psychological review*, vol. 99, no. 1, p. 22, 1992. 5

[11] A. Robins, "Catastrophic forgetting, rehearsal and pseudorehearsal," *Connection Science*, vol. 7, no. 2, pp. 123–146, 1995. 5, 15

[12] B. Ans and S. Rousset, "Avoiding catastrophic forgetting by coupling two reverberating neural networks," *Comptes Rendus de l'Académie des Sciences-Series III-Sciences de la Vie*, vol. 320, no. 12, pp. 989–997, 1997. 5

[13] Z. Li and D. Hoiem, "Learning without forgetting," in *European Conference on Computer Vision*, pp. 614–629, Springer, 2016. 5, 20, 25

[14] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, *et al.*, "Overcoming catastrophic forgetting in neural networks," *Proceedings of the national academy of sciences*, p. 201611835, 2017. 5, 19

[15] S.-W. Lee, J.-H. Kim, J.-W. Ha, and B.-T. Zhang, "Overcoming catastrophic forgetting by incremental moment matching," *arXiv preprint arXiv:1703.08475*, 2017. 5

[16] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, and R. Hadsell, "Progressive neural networks," *arXiv preprint arXiv:1606.04671*, 2016. 5, 21, 55, 76

[17] J.-B. Grill, F. Strub, F. Altché, C. Tallec, P. H. Richemond, E. Buchatskaya, C. Doersch, B. A. Pires, Z. D. Guo, M. G. Azar, *et al.*, "Bootstrap your own latent: A new approach to self-supervised learning," *arXiv preprint arXiv:2006.07733*, 2020. 6, 12, 47, 48

[18] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *nature*, vol. 323, no. 6088, pp. 533–536, 1986. 9

[19] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015. 10

[20] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 58, no. 1, pp. 267–288, 1996. 10

[21] R. Zhang, P. Isola, and A. A. Efros, "Colorful image colorization," in *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part III 14*, pp. 649–666, Springer, 2016. 11

[22] M. Noroozi and P. Favaro, "Unsupervised learning of visual representations by solving jigsaw puzzles," in *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part VI*, pp. 69–84, Springer, 2016. 11

[23] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, *et al.*, "Photo-realistic single image super-resolution using a generative adversarial network," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4681–4690, 2017. 11

[24] O. Elharrouss, N. Almaadeed, S. Al-Maadeed, and Y. Akbari, "Image inpainting: A review," *Neural Processing Letters*, vol. 51, pp. 2007–2028, 2020. 11

[25] C. Doersch, A. Gupta, and A. A. Efros, "Unsupervised visual representation learning by context prediction," in *Proceedings of the IEEE international conference on computer vision*, pp. 1422–1430, 2015. 11

[26] H. Lee, S. J. Hwang, and J. Shin, "Self-supervised label augmentation via input transformations," in *International Conference on Machine Learning*, pp. 5714–5724, PMLR, 2020. 11

[27] P. Baldi, "Autoencoders, unsupervised learning, and deep architectures," in *Proceedings of ICML workshop on unsupervised and transfer learning*, pp. 37–49, JMLR Workshop and Conference Proceedings, 2012. 11

[28] A. Brock, J. Donahue, and K. Simonyan, "Large scale gan training for high fidelity natural image synthesis," *arXiv preprint arXiv:1809.11096*, 2018. 11

[29] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," *arXiv preprint arXiv:2002.05709*, 2020. 12, 28, 54, 56, 57, 58

[30] Y. Tian, D. Krishnan, and P. Isola, "Contrastive multiview coding," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XI 16*, pp. 776–794, Springer, 2020. 12

[31] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, "Momentum contrast for unsupervised visual representation learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9729–9738, 2020. 12, 28

[32] I. Misra and L. v. d. Maaten, "Self-supervised learning of pretext-invariant representations," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 6707–6717, 2020. 12

[33] Z. Wu, Y. Xiong, S. X. Yu, and D. Lin, "Unsupervised feature learning via non-parametric instance discrimination," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3733–3742, 2018. 12

[34] T. Chen, S. Kornblith, K. Swersky, M. Norouzi, and G. E. Hinton, "Big self-supervised models are strong semi-supervised learners," *Advances in neural information processing systems*, vol. 33, pp. 22243–22255, 2020. 12

[35] P. Khosla, P. Teterwak, C. Wang, A. Sarna, Y. Tian, P. Isola, A. Maschinot, C. Liu, and D. Krishnan, "Supervised contrastive learning," in *Advances in Neural Information Processing Systems* (H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, eds.), vol. 33, pp. 18661–18673, Curran Associates, Inc., 2020. 12, 24, 27, 28, 54, 56, 58, 59, 60

[36] F. Graf, C. Hofer, M. Niethammer, and R. Kwitt, "Dissecting supervised constrastive learning," in *International Conference on Machine Learning*, pp. 3821–3830, PMLR, 2021. 12, 58

[37] X. Chen and K. He, "Exploring simple siamese representation learning," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 15750–15758, 2021. 12

[38] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015. 13, 20, 55, 77

[39] R. Volpi, D. Larlus, and G. Rogez, "Continual adaptation of visual representations via domain randomization and meta-learning," in *Proceedings*

of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 4443–4453, 2021. 14

[40] A. Chaudhry, M. Ranzato, M. Rohrbach, and M. Elhoseiny, "Efficient lifelong learning with a-gem," in *ICLR 2019*. 16, 20, 26, 32, 40, 74, 75

[41] T. Lesort, A. Stoian, and D. Filliat, "Regularization shortcomings for continual learning," *arXiv preprint arXiv:1912.03049*, 2019. 16, 20, 25

[42] S.-A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert, "icarl: Incremental classifier and representation learning," in *Proc. CVPR*, 2017. 16, 17, 18, 20, 28, 31, 50, 51, 56, 58, 60, 74, 77, 81, 82, 86, 91

[43] A. Chaudhry, P. K. Dokania, T. Ajanthan, and P. H. Torr, "Riemannian walk for incremental learning: Understanding forgetting and intransigence," *arXiv preprint arXiv:1801.10112*, 2018. 16, 19, 77

[44] S. Hou, X. Pan, C. C. Loy, Z. Wang, and D. Lin, "Learning a unified classifier incrementally via rebalancing," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 831–839, 2019. 17, 18, 21, 26, 47

[45] F. M. Castro, M. J. Marín-Jiménez, N. Guil, C. Schmid, and K. Alahari, "End-to-end incremental learning," in *Proceedings of the European conference on computer vision (ECCV)*, pp. 233–248, 2018. 17, 18

[46] Y. Gal, R. Islam, and Z. Ghahramani, "Deep bayesian active learning with image data," in *International conference on machine learning*, pp. 1183–1192, PMLR, 2017. 18

[47] A. Chaudhry, M. Rohrbach, M. Elhoseiny, T. Ajanthan, P. K. Dokania, P. H. Torr, and M. Ranzato, "On tiny episodic memories in continual learning," *arXiv preprint arXiv:1902.10486*, 2019. 18, 47, 50, 51

[48] R. Aljundi, L. Caccia, E. Belilovsky, M. Caccia, L. Charlin, and T. Tuytelaars, "Online continual learning with maximally interfered retrieval," in *Advances in Neural Information Processing (NeurIPS)*, 2019. 18, 23, 24, 25, 26, 30, 31, 32, 33, 34, 36, 50, 51, 81, 82

[49] T. L. Hayes, K. Kafle, R. Shrestha, M. Acharya, and C. Kanan, "Remind your neural network to prevent catastrophic forgetting," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part VIII 16*, pp. 466–483, Springer, 2020. 18

[50] A. Iscen, J. Zhang, S. Lazebnik, and C. Schmid, "Memory-efficient incremental learning through feature adaptation," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XVI 16*, pp. 699–715, Springer, 2020. 18

[51] T. Lesort, A. Gepperth, A. Stoian, and D. Filliat, "Marginal replay vs conditional replay for continual learning," in *International Conference on Artificial Neural Networks*, pp. 466–480, Springer, 2019. 18

[52] R. Kemker and C. Kanan, "Fearnet: Brain-inspired model for incremental learning," *arXiv preprint arXiv:1711.10563*, 2017. 18

[53] A. Ayub and A. R. Wagner, "Eec: Learning to encode and regenerate images for continual learning," *arXiv preprint arXiv:2101.04904*, 2021. 18

[54] H. Shin, J. K. Lee, J. Kim, and J. Kim, "Continual learning with deep generative replay," in *Advances in Neural Information Processing Systems*, pp. 2990–2999, 2017. 18

[55] M. Zhai, L. Chen, F. Tung, J. He, M. Nawhal, and G. Mori, "Lifelong gan: Continual learning for conditional image generation," *ArXiv*, vol. abs/1907.10107, 2019. 18

[56] Y. Liu, Y. Su, A.-A. Liu, B. Schiele, and Q. Sun, "Mnemonics training: Multi-class incremental learning without forgetting," in *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition*, pp. 12245–12254, 2020. 18

[57] T. Wang, J.-Y. Zhu, A. Torralba, and A. A. Efros, "Dataset distillation," *arXiv preprint arXiv:1811.10959*, 2018. 18

[58] F. Zenke, B. Poole, and S. Ganguli, "Continual learning through synaptic intelligence," *arXiv preprint arXiv:1703.04200*, 2017. 19

[59] R. Aljundi, F. Babiloni, M. Elhoseiny, M. Rohrbach, and T. Tuytelaars, "Memory aware synapses: Learning what (not) to forget," in *ECCV 2018*, 2018. 19, 55, 86

[60] D. Lopez-Paz *et al.*, "Gradient episodic memory for continual learning," in *Advances in Neural Information Processing Systems*, pp. 6467–6476, 2017. 20, 25, 26, 31, 32, 33, 55, 60, 66, 74, 76

[61] R. Aljundi, M. Lin, B. Goujaud, and Y. Bengio, "Gradient based sample selection for online continual learning," *arXiv preprint arXiv:1903.08671*, 2019. 20, 22, 23, 26, 32, 36, 47

[62] M. Farajtabar, N. Azizan, A. Mott, and A. Li, "Orthogonal gradient descent for continual learning," 2019. 20

[63] P. Zhou, L. Mai, J. Zhang, N. Xu, Z. Wu, and L. S. Davis, "M2kd: Multi-model and multi-level knowledge distillation for incremental learning," *arXiv preprint arXiv:1904.01769*, 2019. 21

[64] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9, 2015. 21

[65] P. Dhar, R. V. Singh, K.-C. Peng, Z. Wu, and R. Chellappa, "Learning without memorizing," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 5138–5146, 2019. 21

[66] J. Xu and Z. Zhu, "Reinforced continual learning," *arXiv preprint arXiv:1805.12369*, 2018. 21

[67] R. Aljundi, P. Chakravarty, and T. Tuytelaars, "Expert gate: Lifelong learning with a network of experts," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 21, 55, 76

[68] Y. Marc'Aurelio Ranzato and L. B. S. C. Y. LeCun, "A unified energy-based framework for unsupervised learning," in *Proc. Conference on AI and Statistics (AI-Stats)*, vol. 24, 2007. 21

[69] C. Fernando, D. Banarse, C. Blundell, Y. Zwols, D. Ha, A. A. Rusu, A. Pritzel, and D. Wierstra, "Pathnet: Evolution channels gradient descent in super neural networks," *arXiv preprint arXiv:1701.08734*, 2017. 21

[70] J. Serrà, D. Surís, M. Miron, and A. Karatzoglou, "Overcoming catastrophic forgetting with hard attention to the task," *arXiv preprint arXiv:1801.01423*, 2018. 21, 25

[71] L. Caccia, R. Aljundi, N. Asadi, T. Tuytelaars, J. Pineau, and E. Belilovsky, "New insights on reducing abrupt representation change in online continual learning," *arXiv preprint arXiv:2203.03798*, 2022. 22, 77, 79, 80, 82, 86, 91, 92

[72] S. Farquhar and Y. Gal, "Towards robust evaluations of continual learning," *arXiv preprint arXiv:1805.09733*, 2018. 22

[73] M. De Lange, R. Aljundi, M. Masana, S. Parisot, X. Jia, A. Leonardis, G. Slabaugh, and T. Tuytelaars, "Continual learning: A comparative study on how to defy forgetting in classification tasks," *arXiv preprint arXiv:1909.08383*, 2019. 22, 25, 32, 49

[74] Z. Borsos, M. Mutnỳ, and A. Krause, "Coresets via bilevel optimization for continual learning and streaming," *arXiv preprint arXiv:2006.03875*, 2020. 23, 26

[75] A. Chaudhry, M. Rohrbach, M. Elhoseiny, T. Ajanthan, P. K. Dokania, P. H. Torr, and M. Ranzato, "Continual learning with tiny episodic memories," *arXiv preprint arXiv:1902.10486*, 2019. 23, 26, 27, 30, 31, 33, 50, 55, 58, 60, 66, 74, 76, 81, 82, 86, 91

[76] H. Ahn, J. Kwak, S. Lim, H. Bang, H. Kim, and T. Moon, "Ss-il: Separated softmax for incremental learning," *arXiv preprint arXiv:2003.13947*, 2020. 25, 26, 32

[77] C. Zeno, I. Golan, E. Hoffer, and D. Soudry, "Task agnostic continual learning using online variational bayes," *arXiv preprint arXiv:1803.10123*, 2018. 25, 26

[78] G. M. van de Ven and A. S. Tolias, "Three scenarios for continual learning," *arXiv preprint arXiv:1904.07734*, 2019. 25

[79] F. Normandin, F. Golemo, O. Ostapenko, M. Riemer, P. Rodriguez, J. Hurtado, K. Khetarpal, T. Lesort, L. Charlin, I. Rish, and M. Caccia, "Sequoia - towards a systematic organization of continual learning research," *Github repository*, 2021. 25

[80] T. Lesort, M. Caccia, and I. Rish, "Understanding continual learning settings with data distribution drift analysis," *arXiv preprint arXiv:2104.01678*, 2021. 25

[81] R. Aljundi, F. Babiloni, M. Elhoseiny, M. Rohrbach, and T. Tuytelaars, "Memory aware synapses: Learning what (not) to forget," *arXiv preprint arXiv:1711.09601*, 2017. 25

[82] X. He, J. Sygnowski, A. Galashov, A. A. Rusu, Y. W. Teh, and R. Pascanu, "Task agnostic continual learning via meta learning," *ArXiv*, vol. abs/1906.05201, 2019. 25

[83] M. Caccia, P. Rodriguez, O. Ostapenko, F. Normandin, M. Lin, L. Caccia, I. Laradji, I. Rish, A. Lacoste, D. Vazquez, *et al.*, "Online fast adaptation and knowledge accumulation: a new approach to continual learning," *arXiv preprint arXiv:2003.05856*, 2020. 25, 32, 74, 75

[84] O. Ostapenko, P. Rodriguez, M. Caccia, and L. Charlin, "Continual learning via local module composition," in *Thirty-Fifth Conference on Neural Information Processing Systems*, 2021. 25

[85] J. Von Oswald, D. Zhao, S. Kobayashi, S. Schug, M. Caccia, N. Zucchet, and J. Sacramento, "Learning where to learn: Gradient sparsity in meta and continual learning," *Advances in Neural Information Processing Systems*, vol. 34, 2021. 25

[86] D. Shim, Z. Mai, J. Jeong, S. Sanner, H. Kim, and J. Jang, "Online class-incremental continual learning with adversarial shapley value," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, pp. 9630–9638, 2021. 26

[87] B. Tang and D. S. Matteson, "Graph-based continual learning," *arXiv preprint arXiv:2007.04813*, 2020. 26

[88] P. Buzzega, M. Boschini, A. Porrello, D. Abati, and S. Calderara, "Dark experience for general continual learning: a strong, simple baseline," *arXiv preprint arXiv:2004.07211*, 2020. 26, 32, 40, 50, 51

[89] Z. Mai, R. Li, H. Kim, and S. Sanner, "Supervised contrastive replay: Revisiting the nearest class mean classifier in online class-incremental continual learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3589–3599, 2021. 26, 56, 58

[90] B. Zhao, X. Xiao, G. Gan, B. Zhang, and S. Xia, "Maintaining discrimination and fairness in class incremental learning," *arXiv preprint arXiv:1911.07053*, 2019. 26

[91] Y. Wu, Y. Chen, L. Wang, Y. Ye, Z. Liu, Y. Guo, and Y. Fu, "Large scale incremental learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 374–382, 2019. 26

[92] R. Aljundi, K. Kelchtermans, and T. Tuytelaars, "Task-free continual learning," in *CVPR 2019*, 2018. 27

[93] H.-J. Chen, A.-C. Cheng, D.-C. Juan, W. Wei, and M. Sun, "Mitigating forgetting in online continual learning via instance-aware parameterization," *Advances in Neural Information Processing Systems*, vol. 33, 2020. 27

[94] D. Rolnick, A. Ahuja, J. Schwarz, T. P. Lillicrap, and G. Wayne, "Experience replay for continual learning," *arXiv preprint arXiv:1811.11682*, 2018. 27

[95] H. Qi, M. Brown, and D. G. Lowe, "Low-shot learning with imprinted weights," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5822–5830, 2018. 28, 48

[96] E. Hoffer and N. Ailon, "Deep metric learning using triplet network," in *International workshop on similarity-based pattern recognition*, pp. 84–92, Springer, 2015. 29

[97] D. Shim, Z. Mai, J. Jeong, S. Sanner, H. Kim, and J. Jang, "Online class-incremental continual learning with adversarial shapley value," *arXiv e-prints*, pp. arXiv–2009, 2020. 31, 32, 33, 34, 51

[98] X. Ji, J. Henriques, T. Tuytelaars, and A. Vedaldi, "Automatic recall machines: Internal replay, continual learning and the brain," *arXiv preprint arXiv:2006.12323*, 2020. 31, 32, 33, 34, 81

[99] J. S. Vitter, "Random sampling with a reservoir," *ACM Transactions on Mathematical Software (TOMS)*, vol. 11, no. 1, pp. 37–57, 1985. 31

[100] A. Prabhu, P. H. Torr, and P. K. Dokania, "Gdumb: A simple approach that questions our progress in continual learning," in *European Conference on Computer Vision*, pp. 524–540, Springer, 2020. 32, 33, 36

[101] N. Asadi, S. Mudur, and E. Belilovsky, "Tackling online one-class incremental learning by removing negative contrasts," *arXiv preprint arXiv:2203.13307*, 2022. 46, 77

[102] I. J. Goodfellow, M. Mirza, D. Xiao, A. Courville, and Y. Bengio, "An empirical investigation of catastrophic forgetting in gradient-based neural networks," *arXiv preprint arXiv:1312.6211*, 2013. 46, 53

[103] A. Douillard, M. Cord, C. Ollion, T. Robert, and E. 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*, pp. 86–102, Springer, 2020. 47

[104] L. Caccia, E. Belilovsky, M. Caccia, and J. Pineau, "Online learned continual compression with adaptive quantization modules," *Proceedings of the 37th International Conference on Machine Learning*, 2020. 47

[105] L. Caccia, R. Aljundi, T. Tuytelaars, J. Pineau, and E. Belilovsky, "Reducing representation drift in online continual learning," *arXiv preprint arXiv:2104.05025*, 2021. 47, 48, 50, 51, 53, 56

[106] M. Caron, I. Misra, J. Mairal, P. Goyal, P. Bojanowski, and A. Joulin, "Unsupervised learning of visual features by contrasting cluster assignments," *arXiv preprint arXiv:2006.09882*, 2020. 49

[107] M. Davari, N. Asadi, S. Mudur, R. Aljundi, and E. Belilovsky, "Probing representation forgetting in supervised and unsupervised continual learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16712–16721, 2022. 53, 75, 76, 77, 78, 80, 81, 82, 88, 89

[108] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013. 53

[109] I. Goodfellow, Y. Bengio, and A. Courville, "Deep learning." Book in preparation for MIT Press, 2016. 53

[110] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015. 53, 59, 61, 69

[111] X. He, J. Sygnowski, A. Galashov, A. A. Rusu, Y. W. Teh, and R. Pascanu, "Task agnostic continual learning via meta learning," *arXiv preprint arXiv:1906.05201*, 2019. 54

[112] R. Hadsell, D. Rao, A. A. Rusu, and R. Pascanu, "Embracing change: Continual learning in deep neural networks," *Trends in cognitive sciences*, 2020. 54

[113] E. Oyallon, "Building a regular decision boundary with deep networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5106–5114, 2017. 54

[114] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *European conference on computer vision*, pp. 818–833, Springer, 2014. 54, 57

[115] X. Li, Y. Zhou, T. Wu, R. Socher, and C. Xiong, "Learn to grow: A continual structure learning framework for overcoming catastrophic forgetting," in *International Conference on Machine Learning*, pp. 3925–3934, PMLR, 2019. 55, 76

[116] A. Rosenfeld and J. K. Tsotsos, "Incremental learning through deep adaptation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 42, no. 3, pp. 651–663, 2018. 55, 76

[117] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, *et al.*, "Overcoming catastrophic forgetting in neural networks," *arXiv preprint arXiv:1612.00796*, 2016. 55, 74, 77, 90

[118] F. Zenke, B. Poole, and S. Ganguli, "Improved multitask learning through synaptic intelligence," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2017. 55

[119] C. V. Nguyen, Y. Li, T. D. Bui, and R. E. Turner, "Variational continual learning," *arXiv preprint arXiv:1710.10628*, 2017. 55

[120] F. Huszár, "On quadratic penalties in elastic weight consolidation," *arXiv preprint arXiv:1712.03847*, 2017. 55, 60, 61, 82, 86, 91

[121] I. J. Myung, "Tutorial on maximum likelihood estimation," *Journal of mathematical Psychology*, vol. 47, no. 1, pp. 90–100, 2003. 55, 77

[122] C. V. Nguyen, A. Achille, M. Lam, T. Hassner, V. Mahadevan, and S. Soatto, "Toward understanding catastrophic forgetting in continual learning," *arXiv preprint arXiv:1908.01091*, 2019. 55

[123] G. Arora, A. Rahimi, and T. Baldwin, "Does an LSTM forget more than a CNN? an empirical study of catastrophic forgetting in NLP," in *Proceedings of the The 17th Annual Workshop of the Australasian Language Technology Association*, (Sydney, Australia), pp. 77–86, Australasian Language Technology Association, 4–6 Dec. 2019. 55

[124] S. Kornblith, M. Norouzi, H. Lee, and G. Hinton, "Similarity of neural network representations revisited," in *International Conference on Machine Learning*, pp. 3519–3529, 2019. 55, 57, 68

[125] V. V. Ramasesh, A. Lewkowycz, and E. Dyer, "Effect of scale on catastrophic forgetting in neural networks," in *International Conference on Learning Representations*, 2022. 55, 64, 65

[126] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, "Momentum contrast for unsupervised visual representation learning," *arXiv preprint arXiv:1911.05722*, 2019. 56, 58

[127] D. Hu, Q. Lu, L. Hong, H. Hu, Y. Zhang, Z. Li, A. Shen, and J. Feng, "How well self-supervised pre-training performs with streaming data?," *arXiv preprint arXiv:2104.12081*, 2021. 56

[128] A. Krizhevsky, G. Hinton, *et al.*, "Learning multiple layers of features from tiny images," tech. rep., University of Toronto, 2009. 59, 76, 81, 86

[129] P. Chrabaszcz, I. Loshchilov, and F. Hutter, "A downsampled variant of imagenet as an alternative to the cifar datasets," *arXiv preprint arXiv:1707.08819*, 2017. 59, 81

[130] A. Quattoni and A. Torralba, "Recognizing indoor scenes," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pp. 413–420, IEEE, 2009. 59, 61, 69

[131] P. Welinder, S. Branson, T. Mita, C. Wah, F. Schroff, S. Belongie, and P. Perona, "Caltech-UCSD Birds 200," Tech. Rep. CNS-TR-2010-001, California Institute of Technology, 2010. 59

[132] M.-E. Nilsback and A. Zisserman, "Automated flower classification over a large number of classes," in *Proceedings of the Indian Conference on Computer Vision, Graphics and Image Processing*, Dec 2008. 59, 61

[133] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie, "The Caltech-UCSD Birds-200-2011 Dataset," Tech. Rep. CNS-TR-2011-001, California Institute of Technology, 2011. 61, 69

[134] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," *arXiv preprint arXiv:1711.05101*, 2017. 71, 72

[135] R. Caruana, S. Lawrence, and L. Giles, "Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping," *Advances in neural information processing systems*, pp. 402–408, 2001. 71

[136] I. Loshchilov and F. Hutter, "Sgdr: Stochastic gradient descent with warm restarts," *arXiv preprint arXiv:1608.03983*, 2016. 71

[137] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014. 72

[138] P. Sangkloy, N. Burnell, C. Ham, and J. Hays, "The sketchy database: learning to retrieve badly drawn bunnies," *ACM Transactions on Graphics (TOG)*, vol. 35, no. 4, pp. 1–12, 2016. 73

[139] D. Ha and D. Eck, "A neural representation of sketch drawings," *arXiv preprint arXiv:1704.03477*, 2017. 73

[140] N. Asadi, M. Davar, S. Mudur, R. Aljundi, and E. Belilovsky, "Prototype-sample relation distillation: Towards replay-free continual learning," 2023. 74

[141] P. Khosla, P. Teterwak, C. Wang, A. Sarna, Y. Tian, P. Isola, A. Maschinot, C. Liu, and D. Krishnan, "Supervised contrastive learning," *Advances in Neural Information Processing Systems*, vol. 33, pp. 18661–18673, 2020. 75, 77, 78

[142] M. De Lange and T. Tuytelaars, "Continual prototype evolution: Learning online from non-stationary data streams," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 8250–8259, 2021. 75, 79

[143] O. Vinyals, C. Blundell, T. Lillicrap, D. Wierstra, *et al.*, "Matching networks for one shot learning," *Advances in neural information processing systems*, vol. 29, 2016. 76

[144] H. Cha, J. Lee, and J. Shin, "Co2l: Contrastive continual learning," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 9516–9525, 2021. 77

[145] M. De Lange and T. Tuytelaars, "Continual prototype evolution: Learning online from non-stationary data streams," *arXiv preprint arXiv:2009.00919*, 2020. 77

[146] Y. Tian, D. Krishnan, and P. Isola, "Contrastive representation distillation," *arXiv preprint arXiv:1910.10699*, 2019. 77

[147] J. Zhu, S. Tang, D. Chen, S. Yu, Y. Liu, M. Rong, A. Yang, and X. Wang, "Complementary relation contrastive distillation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9260–9269, 2021. 77

[148] K. Javed and F. Shafait, "Revisiting distillation and incremental classifier learning," in *Asian conference on computer vision*, pp. 3–17, Springer, 2018. 77

[149] T. Barletti, N. Biondi, F. Pernici, M. Bruni, and A. Del Bimbo, "Contrastive supervised distillation for continual representation learning," in *International Conference on Image Analysis and Processing*, pp. 597–609, Springer, 2022. 77

[150] G. Wu, S. Gong, and P. Li, "Striking a balance between stability and plasticity for class-incremental learning," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 1124–1133, 2021. 77, 82, 86, 87, 91

[151] F. Zhu, X.-Y. Zhang, C. Wang, F. Yin, and C.-L. Liu, "Prototype augmentation and self-supervision for incremental learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5871–5880, 2021. 77

[152] W. Park, D. Kim, Y. Lu, and M. Cho, "Relational knowledge distillation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3967–3976, 2019. 77

[153] M. Boudiaf, J. Rony, I. M. Ziko, E. Granger, M. Pedersoli, P. Piantanida, and I. B. Ayed, "A unifying mutual information view of metric learning: cross-entropy vs. pairwise losses," in *European conference on computer vision*, pp. 548–564, Springer, 2020. 79

[154] H. Ahn, J. Kwak, S. Lim, H. Bang, H. Kim, and T. Moon, "Ss-il: Separated softmax for incremental learning," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 844–853, October 2021. 79

[155] L. Yu, B. Twardowski, X. Liu, L. Herranz, K. Wang, Y. Cheng, S. Jui, and J. v. d. Weijer, "Semantic drift compensation for class-incremental learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6982–6991, 2020. 86

[156] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, pp. 1097–1105, 2012. 86

[157] M. Mermillod, A. Bugaiska, and P. Bonin, "The stability-plasticity dilemma: Investigating the continuum from catastrophic forgetting to age-limited learning effects," 2013. 87, 88, 89

[158] S. Dohare, A. R. Mahmood, and R. S. Sutton, "Continual backprop: Stochastic gradient descent with persistent randomness," *arXiv preprint arXiv:2108.06325*, 2021. 88