UNIVERSITÀ DEGLI STUDI
DI TRENTO

DEPARTMENT OF INFORMATION ENGINEERING AND COMPUTER SCIENCE
**ICT International Doctoral School**

# Self-supervised Learning Methods for Vision-based Tasks

Victor Guilherme Turrisi da Costa

Advisor

Prof. PhD. Niculae Sebe

Università degli Studi di Trento

January 2024

# Abstract

*Dealing with large amounts of unlabeled data is a very challenging task. Recently, many different approaches have been proposed to leverage this data for training many machine learning models. Among them, self-supervised learning appears as an efficient solution capable of training powerful and generalizable models. More specifically, instead of relying on human-generated labels, it proposes training objectives that use "labels" generated from the data itself, either via data augmentation or by masking the data in some way and trying to reconstruct it. Apart from being able to train models from scratch, self-supervised methods can also be used in specific applications to further improve a pre-trained model. In this thesis, we propose to leverage self-supervised methods in novel ways to tackle different application scenarios. We present four published papers: an open-source library for self-supervised learning that is flexible, scalable, and easy to use; two papers tackling unsupervised domain adaptation in action recognition; and one paper on self-supervised learning for continual learning. The published papers highlight that self-supervised techniques can be leveraged for many scenarios, yielding state-of-the-art results.*

**Keywords**

Self-supervised Learning, Unsupervised Domain Adaptation, Continual Learning.

# Contents

# List of Tables

# List of Figures

# Chapter 1

# Introduction

## 1.1 Background

In recent years, Artificial Intelligence (AI) saw remarkable progress in the fields of Computer Vision (CV) [120, 132, 95, 90, 137], Natural Language Processing (NLP) [173, 13, 165, 156, 193] and Audio processing [10, 176]. More specifically, in CV, there was remarkable progress in many tasks such as image and video classification, object localization and tracking, segmentation, image generation, image representation, and visual question answering. In turn, this allowed applications in various fields, such as healthcare, robotics, fundamental math, and chemistry. In NLP, recent generative models can provide a conversation-like experience for answering questions, performing code generation, and translation. In Audio Processing, recent methods can generate speech, music, and sound effects, and perform speech recognition.

These breakthroughs and advances were made possible due to Deep Learning (DL), a subfield of Machine Learning (ML), that is concerned with Artificial Neural Networks (ANNs), which are algorithms inspired by how the human brain functions. ANNs are composed of neurons that process information sequentially in the form of a hierarchy of layers. Moreover,

given an input, an ANN will process this input one layer at a time, using the output of a layer as the input to the next layer until the model outputs its final result. Then, given a differentiable loss function that quantifies how poorly the model is performing, the ANN will be updated to reduce this loss, effectively learning from the data, by computing the per-parameter gradients using backpropagation. Although they are very flexible, they also present several disadvantages, *e.g.*, overfitting, their lack of interpretability, the huge amount of data needed to train these methods, and the difficulties in updating these models without forgetting previous information.

### 1.1.1 Learning Paradigms

Machine Learning and Deep Learning can be divided into three main types of learning paradigms: **supervised learning**, **unsupervised learning**, and **reinforcement learning** [12].

**Supervised learning** consists of training a model given labeled data. This means that the excepted output of the model is provided. Additionally, it can be further divided into classification if the expected output is a categorical value, or regression if the output is a continuous variable [12]. To provide a more concrete example, consider a problem where, given an image, the expected output is a label that represents which class the image is from, *e.g.*, a cat, dog, or a car. To train an ML model for this task, one would first need to collect human annotations for the images. Then, the model is trained by using a loss function that considers both the output of the model as well as the expected output. Although this approach is very powerful and has many applications, a clear disadvantage is the need for labeled data. Furthermore, while some labels might be easier to obtain, *e.g.*, the class of the object in the images, others are much harder and require more expert annotators, *e.g.*, a per-pixel segmentation mask of the

objects in an image. Additionally, labels encode a specific human bias for what is important in the data, so the data itself might have conflicting information with the label, meaning that the model will learn to ignore certain elements of the data [83, 7].

On the other hand, **unsupervised learning** consists of training a model without any labels with the goal of learning hidden structures in data. Some traditional methods include autoencoders, dimensionality reduction methods, *e.g.* principal component analysis (PCA), or clustering approaches, *e.g.*, K-means [12]. Autoencoders consist of training two models, one encoder and one decoder, such that the encoder compresses the input data into a lower-dimensional representation and the decoder tries to reconstruct the input data given the lower-dimensional representation [66]. PCA consists of finding the principal components in the data, such that these components capture the largest variations in the data [12]. Lastly, clustering methods try to create groups of similar data points given some distance metric [12].

More recently, **self-supervised learning** (SSL) emerged as a powerful subcategory of unsupervised learning. It consists of leveraging the data itself as the training signal [7]. For instance, a model can be trained to reconstruct masked parts of the data [71, 183] or predict the colors of a gray-scale image [94]. By doing this *proxy* task, the model is able to learn strong feature representations of the data. However, unlike supervised learning, it might not be possible to readily apply it to downstream tasks since the learned representations are not correlated with the task of interest. Nonetheless, after pre-training a model with self-supervised learning, one can simply employ much smaller models, *e.g.* a linear classifier, to learn the final mapping from the features to the desired outputs in a supervised way. More details of self-supervised learning are present in Chapter 2.

Lastly, **reinforcement learning** is a paradigm that learns by interacting with an environment [12]. Given an environment, *e.g.* a game, the reinforcement learning algorithm has a set of actions that it can perform, *e.g.* move to the left, move to the right, and jump, that will modify the environment in some way. Then, the environment will produce some form of reward or penalty, *e.g.* if the player beats the level, it will receive a reward, but if the player loses a life, it will receive a penalty. The model learns by continuously interacting with the environment in order to increase the amount of expected reward gained.

This work focuses on the supervised and unsupervised learning paradigms. More specifically, it leverages self-supervised learning methods for different application areas to make use of the vast amounts of unlabeled data that are available.

## 1.2 Contributions

Here, we present the main contributions of this thesis. They can be divided into three main areas, default self-supervised learning, unsupervised domain adaptation (UDA), and continual learning (CL).

**Open-source Library.** One issue with SSL is that it has a high entry barrier. This is due to the presence of a lot of "moving pieces", *e.g.*, many methods, hyper-parameters, loss functions, training tricks, important architectural choices, and high training costs [7]. The issue is further accentuated when newer methods are developed. They usually also introduce novel training tricks that are not bound to specific methods, *e.g.*, multi-cropping [19] or an asymmetric data augmentation pipeline [69]. This means that older methods can be revisited and improved by applying novel tricks. Because of these reasons, we proposed a unified, open-source li-

brary for learning self-supervised models for vision, called **solo-learn** [40]. It encompasses many methods, training tricks, model architectures, and quality-of-life features such as multi-gpu training, automatic logging, and checkpointing.

**Unsupervised Domain Adaptation.** The first application area that we tackled was Unsupervised Domain Adaptation. UDA methods try to leverage two datasets, one labeled and the other unlabeled, to train a model that performs well on a test dataset, which is related to the unlabeled dataset. Because it needs to learn from unlabeled data, leveraging self-supervised methods is an interesting and powerful approach. We proposed two methods to tackle the UDA scenario. In the first method, we introduce a novel synthetic dataset for UDA in action recognition while also proposing a method that leverages multiple variations of the contrastive loss to learn from the unlabeled data. In the second paper, we introduce a method specifically targeted for Transformers [174], using an information bottleneck loss similar to [191] to train the model given the labeled and unlabeled data.

**Continual Learning.** Lastly, we target the domain of continual learning. It consists of methods that allow a model to continuously learn with incoming data, without losing information about previously seen samples. We posed this problem in the unsupervised setting, where no labels are available. Then, we proposed and systematically evaluated a simple method that relied on a self-supervised loss and matched distillation loss to allow the model to retain information learned from previous data.

## 1.3 Publications

This thesis is composed of a set of publications on the topic of self-supervised learning and its applications to two other areas, unsupervised domain adaptation and continual learning. It contains the following publications, organized in chronological order:

1. *Solo-learn: A Library of Self-supervised Methods for Visual Representation Learning.* **V. G. T. da Costa***, E. Fini*, M. Nabi, N. Sebe, and E. Ricci. Journal of Machine Learning Research (JMLR), 2022 [40]. Link to the code: `https://github.com/vturrisi/solo-learn`.

2. *Dual-head Contrastive Domain Adaptation for Video Action Recognition.* **V. G. T. da Costa**, G. Zara, P. Rota, T. Oliveira-Santos, N. Sebe, V. Murino and E. Ricci. Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV), 2022 [41]. Link to the code: `https://github.com/vturrisi/CO2A`.

3. *Unsupervised domain adaptation for video transformers in action recognition.* **V. G. T. da Costa**, G. Zara, P. Rota, T. Oliveira-Santos, N. Sebe, V. Murino and E. Ricci. Proceedings of the International Conference on Pattern Recognition (ICPR), 2022 [43]. Link to the code: `https://github.com/vturrisi/UDAVT`.

4. *Self-Supervised Models are Continual Learners.* E. Fini*, **V. G. T. da Costa***, X. Alameda-Pineda, E. Ricci, K. Alahari, and J. Mairal. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2022 [57]. Link to the code: `https://github.com/DonkeyShot21/cassle`.

## 1.4 Other Publications

In addition to the publications that are contained in this thesis, other publications were done during the PhD period on other topics. A list of these publications is presented below:

1. *Simplifying Open-Set Video Domain Adaptation with Contrastive Learning.* G. Zara, **V. G. T. da Costa**, S. Roy, P. Rota and E. Ricci. arXiv preprint arXiv:2301.0332, 2023 [190]. Link to the code: `https://github.com/gzaraunitn/COLOSEO`.

2. *Bayesian prompt learning for image-language model generalization.* M. M. Derakhshani, E. Sanchez, A. Bulat, **V. G T. da Costa**, C. G. M. Snoek, G. Tzimiropoulos and B. Martinez. Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), 2023 [47]. Link to the code: `https://github.com/saic-fi/BayesianPrompt-Learning`.

3. *Diversified in-domain synthesis with efficient fine-tuning for few-shot classification.* V. G. T. da Costa*, N. Dall'Asen*, Y. Wang, N. Sebe and E. Ricci. arXiv preprint arXiv:2312.03046, 2023 [39]. Link to the code: `https://github.com/vturrisi/disef`.

## 1.5 Structure of the Thesis

First, in Chapter 2, a comprehensive review of the state-of-the-art self-supervised techniques is presented, which is then followed by a review of unsupervised domain adaptation and continual learning methods.

The rest of the thesis is divided into four main technical chapters that present four main papers on self-supervised learning and its applications.

In Chapter 3, we present the paper entitled "solo-learn: a library of self-supervised methods for visual representation learning powered by Pytorch Lightning". There, we developed a Python-based library to unify self-supervised learning methods and tricks. As the area is very fast-paced, techniques that are proposed by newer papers are not readily available to older methods. Because of that, we initially unified 13 state-of-the-art methods.

In Chapter 4, we present the paper entitled "Dual-head contrastive domain adaptation for video action recognition". We tackle the problem of unsupervised domain adaptation for the video domain. First, we propose a novel method that consists of two heads, one that is trained with a normal cross-entropy loss, and the other that is trained via a contrastive loss. As both heads share the same backbone, the goal is to finetune a pre-trained video-based model such that it adapts to a novel domain. There, domain adaptation is carried out by the contrastive head. Additionally, we also present a novel synthetic dataset, which was the first proposed benchmark for the UDA setting from the synthetic to real domain.

We further study the UDA scenario in Chapter 5. There, we present the paper entitled "Unsupervised Domain Adaptation for Video Transformers in Action Recognition". We adopted a transformer-based video model instead of the CNN-based one. Then, we proposed a novel two-phase training that involved a source-only fine-tuning of a partially frozen model, followed by an adaptation phase using an information bottleneck-based loss.

Chapter 6 presents the paper entitled "Self-supervised Models are Continual Learners". We focused on the problem of continual learning by posing it as a self-supervised problem. Commonly, continual learning consists of fine-tuning a model as new data arrives. However, this data is usually

labeled. We posed the problem in the unsupervised setting by assuming that no labels are available for new data. Additionally, we also provide a novel method for continual learning that is simple and effective, relying on self-supervised learning to adapt the model.

Lastly, in Chapter 7, we present the conclusions and future research directions.

# Chapter 2

# State of the Art

Here, we present a general overview of the state-of-the-art for self-supervised learning, unsupervised domain adaptation, specifically for the action recognition domain, and continual learning.

## 2.1 Self-supervised Learning

Self-supervised learning is a general training strategy that can leverage vast amounts of unlabeled data [7]. Its objective is to learn an encoder network $\mathcal{E}$, *e.g.*, a CNN or a Transformer model, using data $X = \{x_1, x_2, ...x_3\}$, where $x_i$ is a data sample, *e.g.* an image, video or text sentence, that does not contain any additional information like labels or a segmentation mask. Although there are self-supervised methods for many types of data, here we target only vision. More specifically, the methods in this section consider data samples to be images, but the same concepts can be generalized to videos.

Many different self-supervised methods have been proposed in the last few years exploring different strategies to learn from $X$. They can be divided into four main families: deep metric learning, self-distillation, canonical correlation analysis, and masked image modeling [7]. Nonetheless, all

methods follow the same basic principle of learning $\mathcal{E}$ such that it produces similar features for images with similar semantics.

### 2.1.1 Deep Metric Learning Methods

Deep metric learning is a family of methods that rely on encouraging similar feature representations for different transformations of the same input sample (positives) while encouraging different representations *w.r.t* other samples (negatives). These methods are all contrastive and rely on the InfoNCE loss for training [170]. The main representatives are SimCLR [28], MoCo [72, 29, 31] and NNCLR [52].

**SimCLR** [28] starts by building a set of positives $P_i$ and negatives $N_i$ for each sample in the batch $B = \{x_1, x_2, ...x_n\}$. Positive samples are defined as different transformations, or views, of the same input data $x_i$, usually achieved via some form of data augmentation, *e.g.*, cropping, and color jittering. Using $T$, a data augmentation function, SimCLR produces a positive pair $(x_i^a, x_i^b)$, where $x_i^a = T(x_i)$ and $x_i^b = T(x_i)$. On the other hand, negatives are other samples in the batch, *e.g.*, $x_i$ and $x_j$ are negatives if $i \neq j$. In practice, SimCLR uses only two views, so $x_i^a$ and $x_i^b$ are considered as positives given instance $i$ and all the other samples $(x_j^a, x_j^b)_{j=1}^n$, with $j \neq i$, are considered as negatives. These augmented images are forwarded through the encoder $\mathcal{E}$, which is coupled with a small MLP projector $\mathcal{P}$, producing feature representations $z = \mathcal{P}(\mathcal{E}(x))$. Note that $\mathcal{P}$ is used only during training and discarded afterward. For simplicity, only a single instance $x$ is shown, but all $(x_p^a, x_p^b)_{p=1}^n$ pairs are forwarded through $\mathcal{P}(\mathcal{E}(.))$. Finally, the contrastive loss InfoNCE [170] is applied for each sample $i$ in the batch as:

$$L_i^{a,b} = -log \frac{e^{(sim(z_i^a, z_i^b))/\tau)}}{e^{(sim(z_i^a, z_i^b))/\tau)} + \sum_v^{\{a,b\}} \sum_{j=1}^n \mathbb{1}_{j \neq i} e^{(sim(z_i^a, z_j^v)/\tau)}}, \tag{2.1}$$

where $sim(.)$ is a similarity function, *e.g.* cosine similarity and $\tau$ is a temperature parameter. In practice, the loss is symmetrized by swapping the views, computing $L_i = \frac{L_i^{a,b} + L_i^{b,a}}{2}$ for each sample and averaging the loss across the batch as:

$$L = \frac{1}{n} \sum_{i=1}^{n} L_i. \tag{2.2}$$

Intuitively, the InfoNCE loss is minimized when the positives are closer together and negatives are far apart, *i.e.* positive pairs have similar feature representations whereas negative pairs have dissimilar feature representations.

**MoCo** [72] differs from SimCLR as it does not use the same encoder $\mathcal{E}$ and projector $\mathcal{P}$ for both views of the data. Instead, it leverages two versions of $\mathcal{P}(\mathcal{E}(.))$, the default one, called the online model, and another that is a momentum copy $\mathcal{P}^{\mathcal{M}}(\mathcal{E}^{\mathcal{M}}(.))$ of the online model, called the offline model. Instead of updating $\mathcal{P}^{\mathcal{M}}(\mathcal{E}^{\mathcal{M}}(.))$ via gradient descent, we simply update it as a momentum-based exponential moving average of $\mathcal{P}(\mathcal{E}(.))$, *i.e.*, $\theta_{\mathcal{M}} = m\,\theta_{\mathcal{M}} + (1-m)\,\theta$, where $\theta_{\mathcal{M}}$ is a parameter in the offline model, $\theta$ is a parameter on the online model and $m$ is a momentum coefficient that controls the update rate. Additionally, instead of relying on other instances in the batch to construct the set of negatives, it keeps a bounded queue $Q$ of the previously seen instances and uses them as negatives. This makes the number of negatives per instance much larger than in SimCLR, as it is not bounded by batch size. Finally, the InfoNCE loss for MoCo is computed as:

$$L_i = -log\, \frac{e^{(sim(q_i,k_i))/\tau)}}{\sum_{j=1}^{Q} e^{(sim(q_i,k_j)/\tau)}}, \tag{2.3}$$

where $q = \mathcal{P}(\mathcal{E}(x^a))$ is a "query" and $k = \mathcal{P}^{\mathcal{M}}(\mathcal{E}^{\mathcal{M}}(x^b))$ is a "key". As for SimCLR, MoCo also swaps views and computes the symmetric loss, such that, $q = \mathcal{P}(\mathcal{E}(x^b))$ and $k = \mathcal{P}^{\mathcal{M}}(\mathcal{E}^{\mathcal{M}}(x^a))$. **MoCo v2** [29] improves the

set of augmentations used and leverages more recent training strategies. **MoCo v3** [31] drops the idea of a separate queue and uses the instances in the batch to compute the keys $k$ as it finds that large enough batches are sufficient for producing a good set of negatives.

**NNCLR** [52] builds upon SimCLR, introducing the idea of using the nearest neighbor of sample $i$ as its positive instead of $i$ itself. For that, it keeps a support set in the form of a queue $Q$, similar to MoCo [72], which is used to retrieve the nearest neighbor of a sample. Then, the second view of $i$, *i.e.*, $x_i^b$, is replaced with its nearest neighbor in $Q$ as $nn_i = NN(x_i^b, Q)$, constructing the positive pair $(x_i^a, nn_i)$. Lastly, the same loss in SimCLR is applied considering $(x_i^a, nn_i)$ as the positive pair for $i$ and $(x_j^a, x_j^b)_{j=1}^n$, with $j \neq i$, as the negatives.

### 2.1.2 Self-distillation Methods

Self-distillation methods consist of feeding two different augmented views of the same input data through two parallel encoders, usually an online and an offline encoder, as in [73], and mapping one output to the other via a predictor network. Unlike deep metric learning methods, self-distillation methods do not rely on negatives to avoid collapsed solutions, which occur when the model outputs the same values for all samples. Instead, these methods employ other strategies, such as asymmetric architectures or the stop gradient operation. Its main representatives consist of BYOL [69], SimSiam [30] and DINO [21, 120].

**BYOL** [69] proposes many improvements upon previous methods. It leverages the idea of having an online and an offline model as in MoCo [72], but instead of using a symmetric model, where both views are processed by the same number of layers/blocks, it introduces an additional predictor network $\mathcal{H}$ to the online model to create an asymmetric structure. Second,

it replaces the contrastive loss with a simple mean-squared error (MSE) loss, defined as:

$$L = 2 - 2\,\frac{p_i^a}{||p_i^a||}\,\frac{z_i^b}{||z_i^b||}, \tag{2.4}$$

where $p_i^a = \mathcal{H}(\mathcal{P}(\mathcal{E}(x_i^a)))$ is produced by the online model and $z_i^b = \mathcal{P}^{\mathcal{M}}(\mathcal{E}^{\mathcal{M}}(x_i^b))$ is produced by the offline model. As in previous methods, the loss is also symmetrized by using $p_i^b = \mathcal{H}(\mathcal{P}(\mathcal{E}(x_i^b)))$ and $z_i^a = \mathcal{P}^{\mathcal{M}}(\mathcal{E}^{\mathcal{M}}(x_i^a))$. Lastly, it also proposes to have a set of asymmetric augmentation pipelines, such that $T$ is represented as $T^a$ and $T^b$ that apply distinct operations, such that $x_i^a = T^a(x_i)$ and $x_i^b = T^b(x_i)$.

**SimSiam** [30] is conceptually similar to BYOL but simplifies the method by not using a momentum copy of the model. Instead, it shows that only a stop gradient operation and the asymmetry of the model are enough to avoid collapse. The stop gradient operation avoids that gradients are backpropagated through part of the computational graph. Finally, SimSiam uses the same MSE loss from BYOL, with $p_i^a = \mathcal{H}(\mathcal{P}(\mathcal{E}(x_i^a)))$, $p_i^b = \mathcal{H}(\mathcal{P}(\mathcal{E}(x_i^b)))$, $z_i^a = stop\_grad(\mathcal{P}(\mathcal{E}(x_i^a)))$ and $z_i^b = stop\_grad(\mathcal{P}(\mathcal{E}(x_i^b)))$.

**DINO** [21] uses only the encoder $\mathcal{E}$, projector $\mathcal{P}$ and their momentum copies $\mathcal{E}^{\mathcal{M}}$ and $\mathcal{P}^{\mathcal{M}}$. Then, given $z_i^a = \mathcal{P}(\mathcal{E}(x_i^a))$ and $z_i^b = \mathcal{P}^{\mathcal{M}}(\mathcal{E}^{\mathcal{M}}(x_i^a))$, it computes probability distribution of the student as $ps_i = softmax(z_i^a)$ and the probability distribution of the teacher as $pt_i = stop\_grad(softmax($ $centering(z_i^b)))$. The *centering* operation consists of subtracting $C$, the a moving average representation of $z$, from $z$. Then, the objective consists of minimizing the cross-entropy loss across the two distributions, computed as:

$$L = H(pt_i, ps_i), \text{ where } H(a, b) = -a\,log\,b. \tag{2.5}$$

The loss is also symmetrized by computing $z_i^b = \mathcal{P}(\mathcal{E}(x_i^b))$, $z_i^a = \mathcal{P}^{\mathcal{M}}(\mathcal{E}^{\mathcal{M}}(x_i^a))$, $ps_i = softmax(z_i^b)$ and $pt_i = stop\_grad(softmax(centering(z_i^a)))$. In **DINO v2** [120] the authors propose a set of improvements to DINO. They

replaced the centering operation with the Sinkhorn-Knopp centering from [20]; applied randomly masking to the patches and a loss to reconstruct the feature representations of the masked patches as in [197]; added newer regularization strategies; and performed distillation from a larger model to train smaller models.

### 2.1.3 Canonical Correlation Analysis Methods

Canonical correlation analysis methods work by inferring the relationship between two variables, leveraging either cluster assignments or cross-covariance matrices. Its main representatives consist of SwAV [19], Barlow Twins [191], and VICReg [9].

**SwAV** [19] consists of clustering one view of the images in the batch and then trying to predict its cluster assignments given the other view. More specifically, it performs clustering via the Sinkhorn-Knopp algorithm, which produces soft cluster assignments instead of hard assignments. Then, given one view, it tries to predict the soft cluster assignments of the other view. Additionally, it also proposes multi-cropping, which consists of leveraging multiple smaller crops, instead of the commonly used two crops. Unlike simply increasing the number of large crops, using smaller crops is much more efficient to compute.

**Barlow Twins** [191] works by computing the cross-correlation matrix between two augmented versions of the same instance, $x_i^a$ and $x_i^b$ and approximating it to the identity matrix. This naturally avoids collapsed solutions as it decorrelates different features while maximally correlating the same features across views. Unlike other methods, it employs a simple symmetric structure without a momentum model. Given views $x_i^a$ and $x_i^b$ it produces $z_i^a = \mathcal{P}(\mathcal{E}(x_i^a))$ and $z_i^b = \mathcal{P}(\mathcal{E}(x_i^b))$. Then, it computes the

cross-correlation for feature $f$ as $C_f f$ as :

$$C_{f,f}^{a,b} = \frac{\sum_i z_{i,f}^a \, z_{j,f}^b}{\sqrt{\sum_i (z_{i,f}^a)^2} \sqrt{\sum_i (z_{i,f}^b)^2}}, \tag{2.6}$$

where $i$ indexes different instances and $f$ indexes a specific feature in the vector. Lastly, it computes its loss as:

$$L = \sum_f (1 - C_{f,f})^2 + \lambda \sum_f \sum_{g \neq f} (C_{f,g})^2. \tag{2.7}$$

Note that $C$ is a square matrix that contains the cross-correlation values of all feature pairs, with values between -1, indicating perfect anti-correlation, and 1, indicating perfect correlation. Minimizing the loss consists of making $C$ approximate the identity, making diagonal elements closer to 1 and off-diagonal elements closer to 0.

**VICReg** [9] builds upon Barlow Twins [191] but decomposes the desired properties of the loss into individual terms. The loss consists of three individual terms. The first enforces invariance between samples by minimizing the MSE error of positive samples. The second enforces variance across the representations of different samples by applying a hinge loss to maintain the standard deviation over instances of the batch above a given threshold, defined as:

$$v(Z) = \frac{1}{f} \sum_f max(0, \gamma - S(z_{.,f}, \epsilon), \tag{2.8}$$

where $f$ is a feature, $z_{.,f}$ is a vector containing all values of $f$ for the instances in the batch, $\gamma = 1$, $\epsilon$ a small value to prevent numerical instability and $S(x, \epsilon) = \sqrt{Var(x) + \epsilon}$ the regularized standard deviation. The third decorrelates different features across instances in the batch, reusing $C_{f,g}^{a,b} = \sum_f \sum_{g \neq f} C_{f,g}^2$ from Barlow Twins [191]. Then, the final loss is defined as:

$$L = \lambda \, MSE(Z^a, Z^b) + \mu[v(Z^a) + v(Z^b)] + \nu[C_{f,g}^{a,a} + C_{f,g}^{b,b}], \tag{2.9}$$

where $Z^a$ contains all $z_i^a$ of the batch.

### 2.1.4 Masked Image Modelling Methods

Masked image modeling methods are a family of methods that rely on corrupting the input image and then reconstructing it. Its main representatives are BEiT [8], MAE [71], SimMIM [183] and iBOT [197].

**BEiT** [8] tries to perform masked image modeling by classifying masked patches from a set of visual tokens. First, it learns an image tokenizer in the form of a discrete variational autoencoder. Then, the training objective consists of feeding a masked image $x'$, that is produced by randomly masking patches in an image $x$, through the encoder $\mathcal{E}$ followed by a small model auxiliary network that classifies each patch into the set of possible image tokens from the first step. The loss is then a simple cross-entropy loss for classification.

**MAE** [71] consists of a encoder-decoder transformer. Given the encoder model $\mathcal{E}$ and an auxiliary decoder $\mathcal{D}$, the training objective consists of reconstructing masked regions of an input image $x$. First, $x$ is masked to produce $x'$ and then fed to $\mathcal{E}$ to produce $z = \mathcal{E}(x')$. Then, $z$ is concatenated with learnable mask patches $m$ and given to the decoder $\mathcal{D}$, producing, $o = \mathcal{D}(z+m)$. Lastly, the loss is simply an MSE loss between the reconstructed masked patches in $o$ and the original image $x$.

**SimMIM** [183] is similar to MAE, but proposes a series of simplifications. First, instead of feeding $\mathcal{E}$ with only $x'$, it feeds it with both $x'$ and the mask patches $m$. Second, it replaces the transformer decoder $\mathcal{D}$ with a linear layer. Lastly, it uses an L1 loss instead of the MSE loss (L2).

**iBOT** [197] uses two transformer networks, the online and the offline models. However, instead of feeding the online network with the whole

image, it randomly masks some of its patches. Then, it has two losses. First, the self-distillation loss from DINO [21] between the [CLS] tokens of the two views, that were processed by the two different models. Second, it performs self-distillation between the masked patch features produced by the online model and the unmasked patch features produced by the offline model.

## 2.2 Unsupervised Domain Adaptation

Commonly, deep learning models are built under the assumption that the training dataset (called source dataset in the UDA literature) and the test dataset (called target dataset in the UDA literature) have the same data and label distribution. However, this is not usually the case in real-world applications due to a phenomenon called domain shift, where the data distribution between the source and the target data changes.

To address this problem, unsupervised domain adaptation methods try to leverage supervised source data and unsupervised target data to train, or adapt, a model that performs well on the test target data, which is a test set from the target domain. More formally, given source data $\mathcal{S} = \{(X_i^S, y_i^S)\}^{N^S}$, where $X_i^S$ can be an image, video, or any other input data, $y_i^S$ is its respective label and $N$ is the size of the dataset, and training target data $\mathcal{T} = \{(X_i^T)\}^{N^T}$, that does not contain labels, the goal is to learn a function $f_\theta : X \rightarrow y$, where $\theta$ is the set of parameters of a DL model, that performs well on the test target data. Note that although the target data does not have any labels available for training, the test target data contains labels that are used to evaluate a model.

UDA methods usually perform adaptation by aligning the source and target data distributions [104, 17, 138, 110, 169], leveraging adversarial

learning methods [168, 103, 76] or even self-supervised learning strategies [184, 182, 2, 154].

## 2.3 Continual Learning

Continual Learning is an area that studies the capacities of models to learn sequentially with new data. The main issue with feeding sequential data to a model, especially a DL model, is a phenomenon called catastrophic forgetting. This refers to the fact that the model forgets old information as it is fed new data. So, the majority of CL focuses on proposing techniques to adapt models or part of them, continually, as the data distribution changes, focusing on preserving past knowledge.

According to [45], they can be divided into three major groups: replay-based [121, 136, 134, 15, 26, 106], regularization-based [58, 99, 144, 91, 192, 4, 23, 51, 79, 25, 179, 24], and parameter isolation methods [139, 142]. Replay-based methods consist of keeping track of important past data and using it with new data to update the model. On the other hand, regularization-based methods work by keeping previous versions of the model and making sure that the current model does not deviate a lot from them. Lastly, parameter isolation methods work by continuously adding new parameters to the model to adapt it to new data.

# Chapter 3

# solo-learn: a library of self-supervised methods for visual representation learning powered by Pytorch Lightning

## 3.1 Introduction

Deep networks trained with large annotated datasets have shown stunning capabilities in the context of computer vision. However, the need for human supervision is a strong limiting factor. Unsupervised learning aims to mitigate this issue by training models from unlabeled datasets. The most prominent paradigm for unsupervised visual representation learning is Self-supervised Learning (SSL), where the intrinsic structure of the data provides supervision for the model. Recently, the scientific community devised increasingly effective SSL methods that match or surpass the performance of supervised methods. Nonetheless, implementing and reproducing such works turns out to be complicated. Official repositories of state-of-the-art SSL methods have very heterogeneous implementations or no implementation at all. Although a few SSL libraries [68, 155] are available, they

assume that larger-scale infrastructures are available or they lack some recent methods. When approaching SSL, it is hard to find a platform for experiments that allows running all current state of the art methods with low engineering effort and at the same time is effective and straightforward to train. This is especially problematic because, while the SSL methods seem simple on paper, replication of published results can involve a huge time and effort from researchers. Sometimes official implementations of SSL methods are available, however, releasing standalone packages (often incompatible with each other) is not sufficient for the fast-paced progress in research and emerging real-world applications. There is no toolbox offering a genuine off-the-shelf catalog of state-of-the-art SSL techniques that is computationally efficient, which is essential for in-the-wild experimentation.

To address these problems, we present `solo-learn`, an open-source framework that provides standardized implementations for a large number of state-of-the-art SSL methods. We believe `solo-learn` will enable a trustworthy and reproducible comparison between the state of the art methods. The code that powers the library is written in Python, using Pytorch [124] and Pytorch Lightning(PL) [157] as back-ends and Nvidia DALI[1] for fast data loading, and supports more modern methods than related libraries. The library is highly modular and can be used as a complete pipeline, from training to evaluation, or as standalone modules.

## 3.2 The `solo-learn` Library: An Overview

Currently, we are witnessing an explosion of works on SSL methods for computer vision. Their underlying idea is to unsupervisedly learn feature

[1] https://github.com/NVIDIA/DALI

representations by enforcing similar feature representations across multiple views from the same image while enforcing diverse representations for other images. To help researchers have a common testbed for reproducing different results, we present `solo-learn`, which is a library of self-supervised methods for visual representation learning. The library is implemented in Pytorch, providing state-of-the-art self-supervised methods, distributed training pipelines with mixed-precision, faster data loading, online linear evaluation for better prototyping, and many other training strategies and tricks presented in recent papers. We also provide an easy way to use the pre-trained models for object detection, via DetectronV2 [180]. Our goal is to provide an easy-to-use library that can be easily extended by the community, while also including additional features that make it easier for researchers and practitioners to train on smaller infrastructures.

### 3.2.1 Self-supervised Learning Methods

We implemented 13 state-of-the-art methods, namely, Barlow Twins [191], BYOL [69], DeepCluster V2 [20], DINO [21], MoCo V2+ [29], NNCLR [52], ReSSL [195], SimCLR [28], Supervised Contrastive Learning [86], SimSiam [30], SwAV [20], VICReg [9] and W-MSE [53].

### 3.2.2 Architecture

In Figure 3.1, we present an overview of how a training pipeline with `solo-learn` is carried out. In the bottom, we show the packages and external data at each step, while at the top, we show all the defined variables on the left and an example of the newest defined variable on the right. First, the user interacts with `solo.args`, a subpackage that is responsible for handling all the parameters selected by the user and providing automatic setup. Then, `solo.methods` interacts with `solo.losses`

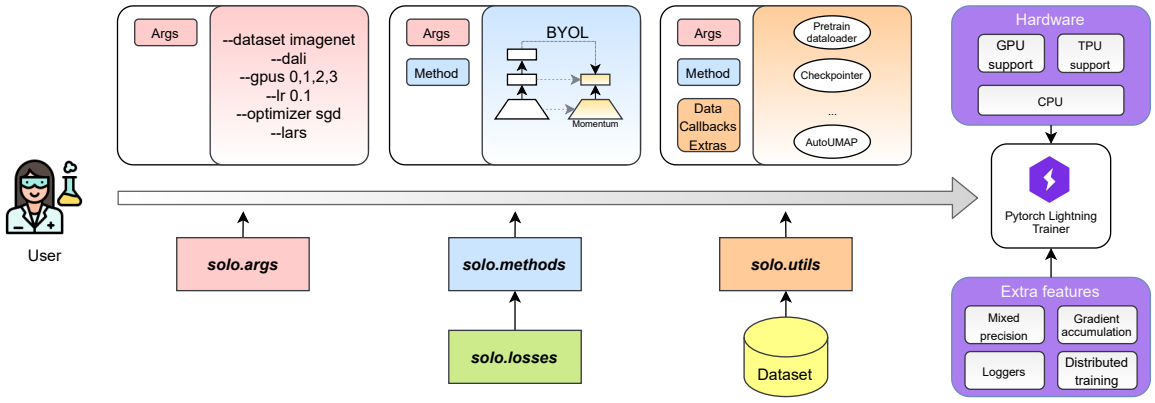



Figure 3.1: Overview of `solo-learn`.

to produce the selected self-supervised method. While `solo.methods` contains all implemented methods, `solo.losses` contains the loss functions for each method. Afterwards, `solo.utils` handles external data to produce the pretrain dataloader, which contains all the transformation pipelines, model checkpointer, automatic UMAP visualization of the features, other backbone networks, such as ViT [49] and Swin [102], and many other utility functionalities. Lastly, this is given to a PL trainer, which provides hardware support and extra functionality, such as, distributed training, automatic logging results, mixed precision and much more. We note that although we show all subpackages working together, they can be used in a standalone fashion with minor modifications. Apart from that, we have documentations in the folder **docs**, downstream tasks in **downstream**, unit tests in **tests** and pretrained models in **zoo**.

### 3.2.3 Comparison to Related Libraries

The most related libraries to ours are VISSL [68] and Lightly [155], which lack some of our key features. First, we support more modern SSL methods, such as BYOL, NNCLR, SimSiam, VICReg, W-MSE and others. Second, we target researchers with fewer resources, namely from 1 to 8 GPUs,

allowing much faster data loading via DALI. Lastly, we provide additional utilities, such as automatic linear evaluation, support to custom datasets and automatically generating UMAP [113] visualizations of the features during training.

## 3.3 Experiments

**Benchmarks.** We benchmarked the available SSL methods on CIFAR-10 [92], CIFAR-100 [92] and ImageNet-100 [46] and made public the pre-trained checkpoints. For Barlow Twins, BYOL, MoCo V2+, NNCLR, SimCLR and VICReg, hyperparameters were heavily tuned, reaching higher performance than reported on original papers or third-party results. Tab. 3.1 presents the top-1 and top-5 accuracy values for the online linear evaluation. For ImageNet-100, traditional offline linear evaluation is also reported. We also compare with the results reported by Lightly in Tab. 3.3.

**Nvidia DALI vs traditional data loading.** We compared the training speeds and memory usage of using traditional data loading via Pytorch Vision[2] against data loading with DALI. For consistency, we ran three different methods (Barlow Twins, BYOL and NNCLR) for 20 epochs on ImageNet-100. Tab. 3.2 presents these results.

## 3.4 Conclusion

Here, we presented `solo-learn`, a library of self-supervised methods for visual representation learning, providing state-of-the-art self-supervised methods in Pytorch. The library supports distributed training, fast data

[2]https://github.com/pytorch/vision

Table 3.1: Online linear evaluation accuracy on CIFAR-10, CIFAR-100 and ImageNet-100. In brackets, offline linear evaluation accuracy is also reported for ImageNet-100.

| Method | CIFAR-10 | | CIFAR-100 | | ImageNet-100 | |
|---|---|---|---|---|---|---|
| | Acc@1 | Acc@5 | Acc@1 | Acc@5 | Acc@1 | Acc@5 |
| Barlow Twins | 92.10 | 99.73 | 70.90 | 91.91 | 80.38 (80.16) | 95.28 (95.14) |
| BYOL | 92.58 | 99.79 | 70.46 | 91.96 | 80.16 (80.32) | 94.80 (94.94) |
| DeepCluster V2 | 88.85 | 99.58 | 63.61 | 88.09 | 75.36 (75.40) | 93.22 (93.10) |
| DINO | 89.52 | 99.71 | 66.76 | 90.34 | 74.84 (74.92) | 92.92 (92.78) |
| MoCo V2+ | 92.94 | 99.79 | 69.89 | 91.65 | 78.20 (79.28) | 95.50 (95.18) |
| NNCLR | 91.88 | 99.78 | 69.62 | 91.52 | 79.80 (80.16) | 95.28 (95.28) |
| ReSSL | 90.63 | 99.62 | 65.92 | 89.73 | 76.92 (78.48) | 94.20 (94.24) |
| SimCLR | 90.74 | 99.75 | 65.78 | 89.04 | 77.04 (77.48) | 94.02 (93.42) |
| Simsiam | 90.51 | 99.72 | 66.04 | 89.62 | 74.54 (78.72) | 93.16 (94.78) |
| SwAV | 89.17 | 99.68 | 64.88 | 88.78 | 74.04 (74.28) | 92.70 (92.84) |
| VICReg | 92.07 | 99.74 | 68.54 | 90.83 | 79.22 (79.40) | 95.06 (95.02) |
| W-MSE | 88.67 | 99.68 | 61.33 | 87.26 | 67.60 (69.06) | 90.94 (91.22) |

Table 3.2: Speed and memory comparison with and without DALI on ImageNet-100.

| Method | DALI | 20 epochs | 1 epoch | Speedup | Memory |
|---|---|---|---|---|---|
| Barlow Twins | | 1h 38m 27s | 4m 55s | - | 5097 MB |
| | ✓ | 43m 2s | 2m 10s | 56% | 9292 MB |
| BYOL | | 1h 38m 46s | 4m 56s | - | 5409 MB |
| | ✓ | 50m 33s | 2m 31s | 49% | 9521 MB |
| NNCLR | | 1h 38m 30s | 4m 55s | - | 5060 MB |
| | ✓ | 42m 3s | 2m 6s | 64% | 9244 MB |

Table 3.3: Comparison with Lightly on CIFAR10.

| Method | Ours | Lightly |
|---|---|---|
| SimCLR | 90.74 | 89.0 |
| MoCoV2+ | 92.94 | 90.0 |
| SimSiam | 90.51 | 91.0 |

loading and provides many utilities for the end-user, such as online linear evaluation for better prototyping and faster development, many training tricks, and visualization techniques. We are continuously adding new SSL methods, improving usability, documents, and tutorials. Finally, we welcome contributors to help us at `https://github.com/vturrisi/solo-learn`.

# Chapter 4

# Dual-head contrastive domain adaptation for video action recognition

## 4.1 Introduction

Visual recognition models are built under the assumption that the training and test data are drawn from the same distribution. Unfortunately, this assumption rarely holds in practice, leading to a drop in performance on the test data. To address this problem, over the years, several unsupervised domain adaptation (UDA) methods [38] have been developed. UDA approaches leverage relevant knowledge from labelled data in a source domain to learn a model for a different, but related, target domain where no annotations are provided. These methods have already proved to be effective in several image-related tasks, ranging from object recognition [109, 140, 169, 105] to semantic segmentation [78, 194, 77, 32] and object detection [96]. However, so far much less attention has been devoted to video analysis which, compared to image-related applications, is undoubtedly more challenging. In particular, videos introduce one more level of variation in the data, *i.e.* the temporal dimension, which increases the de-

Figure 4.1: Sample sequences from the Mixamo and the Kinetics datasets. Keypoints are also provided for the Mixamo dataset.


mand for hardware and leads to additional complexity. To address UDA in the context of video analysis, researchers have proposed to rethink the traditional strategies for images in order to learn robust classifiers for videos, using domain-invariant deep feature representations [36, 37, 117, 122, 27].

Action recognition [56, 196, 166, 22] is one of the fundamental problems in video analysis. This task is inherently challenging as actions can vary over time according to several factors, such as speed, duration, relative movement between the actor and the camera, and the actor's interaction with surrounding objects. Also, people can perform the same action in different ways, raising a challenging ambiguity. Although Convolutional Neural Networks-based (CNN) approaches have enabled significant advances, this task still poses many open problems. In particular, the important variation derived from video sequences makes the domain shift harder to address compared to the case of images. One way to address the variation issue without increasing the cost of data acquisition is to rely on synthetic data; however, such data still present the challenge of the large domain gap.

This paper advances the state of the art in UDA for video action recognition by proposing a novel two-headed deep architecture. The design of

our model is motivated by the idea of jointly leveraging source supervision, target pseudo-labelling and contrastive learning to mitigate the domain shift arising in video action recognition. Our network consists of a shared encoder that extracts feature representations from clips of source and target videos and aggregates them with an attention mechanism. The video-level features provided by the encoder are then fed to two separate network heads that learn complementary classification models, one based on a cross-entropy loss and the other trained with contrastive losses. A key element of our approach is a novel consistency loss term that enforces the network to produce coherent predictions among the two network heads, resulting in more reliable pseudo-labels for the target samples. Target pseudo-labels and source labels are then jointly exploited by a novel inter-domain contrastive loss, which performs conditional feature alignment among data distributions of different domains, thus counteracting domain shift. Lastly, inspired by recent literature on contrastive learning [28], we leverage video-specific data augmentations, both at clip and video-level, to learn multi-scale spatio-temporal feature representations for target videos. Our Contrastive Conditional domain Alignment approach is named as $CO^2A$.

An important contribution of this work is also the introduction of *Mixamo → Kinetics*, a new large-scale dataset for video action recognition. The proposed dataset is the first benchmark that allows studying the challenging problem of UDA when source data are synthetically generated videos and target data are real Youtube videos. In our dataset, frame sequences in the synthetic domain are generated using realistic motion sequences gathered from Mixamo[1] and rendered using Blender[2], resulting in videos depicting actions performed by 3D avatars with different visual

[1] `https://www.mixamo.com/`

[2] `https://www.blender.org/`

appearances in a randomised 3D scene. Data of the real domain, instead, are obtained from the popular Kinetics dataset [147]. Sample frames for two sequences of our dataset are shown in Figure 4.1.

**Contributions.** To summarise, our contributions are the following: (i) a UDA approach for action recognition in videos that exploits label and pseudo label information for semantic alignment of the source and target data distributions. The proposed method achieves state-of-the-art performance on several challenging benchmarks for action recognition, such as *UCF* $\leftrightarrow$ *HMDB*[27], *UCF* $\leftrightarrow$ *Olimpics Sports*[27] and *Kinetics* $\rightarrow$ *NEC-Drone*[36]; (ii) a novel deep architecture that seamlessly integrates three components: a dual head structure to learn two different but coherent models (based on classification loss and contrastive losses, respectively), an inter-domain contrastive loss, which exploits source labels and target pseudo-labels for domain distribution alignment, and a multi-level contrastive loss for target feature learning; (iii) a novel large-scale synthetic-to-real dataset, *Mixamo* $\rightarrow$ *Kinetics*, devised for testing UDA methods for action recognition.

To the best of our knowledge, the dataset we propose is the only benchmark that will be publicly available for assessing the ability of UDA approaches to transfer knowledge from the synthetic to the real domain in videos.

## 4.2 Related Work

**Action recognition.** Different deep architectures for action recognition have been proposed in the last few years. For instance, in [56], two-stream networks were proposed to jointly use RGB and optical flow frames within two 2D CNNs, modelling temporal information. Zhou *et al.* [196] intro-

duced Temporal Relation Networks, a deep model that employs a specialised pooling layer to model temporal relations between frames. Other works considered 3D CNNs to learn spatio-temporal features. Tran *et al.* [166] proposed C3D, which directly employs 3D convolutions rather than 2D ones. Carreira *et al.* [22] introduced I3D, a deep network that integrates inflated 2D convolutional filters to leverage large-scale pre-trained 2D models. Very recently, some other works have proposed approaches based on contrastive learning for extracting useful motion representations for action recognition [188, 131, 177, 126]. Different from our work, all mentioned studies tackle the traditional supervised action recognition problem (no domain shift).

**UDA for images.** Existing approaches mostly differ on the strategy used to cope with domain shift. One category of methods performs domain distribution alignment by matching statistical moments of the first and second-order of the source and target data distributions [104, 17, 138, 110, 169]. Recently, these methods were improved considering label information during the alignment process [84]. Another prominent strategy in UDA is adversarial training [103, 78, 63, 167], where discriminative and domain-agnostic feature representations are learned by coupling a domain discriminator with the source classification loss. Similar to moment matching methods, the best-performing approaches of this category leverage the semantic information given by classifier predictions to perform adversarial adaptation. Generative adversarial networks [67] have also been considered to address the domain shift [76, 175, 141], as they permit generating target-like images that can then be used to train a target classification model. Furthermore, recent works have considered self-supervised learning and auxiliary tasks, such as predicting rotations [154] or image patches permutations in a jigsaw setting [14] to learn domain-invariant features.

Our work is related to previous methods based on semantic distribution

alignment [103, 84, 116], but innovates over past literature since adaptation is performed thanks to a novel domain contrastive alignment loss. CO$^2$A also shares some similarities with previous UDA approaches with double classifier structure [140]. However, one particular aspect of CO$^2$A is the choice of the supervised contrastive loss [86] for one of the two network heads. Lastly, contrastive learning for UDA has been recently studied in [123, 88]. However, the deep architectures in [123, 88] are radically different from ours and do not exploit contrastive learning within a two-headed neural network. Moreover, these works do not tackle the more challenging problem of video action recognition.

**UDA for action recognition.** Despite its importance in many real-world applications, only a few works addressed the problem of domain shift for action recognition [117, 122, 27, 36, 37]. Chen *et al.* [27] proposed the Temporal Attentive Adversarial Adaptation Network (TA$^3$N), which integrates a temporal relation module to simultaneously learn the temporal dynamics and achieve domain alignment. Pan *et al.* [122] introduced Temporal Co-attention Network (TCoN), a deep architecture with a cross-domain attention module to match the distributions of temporally aligned features between source and target domains. In [36], the problem of UDA for recognising actions was considered in the specific case of videos collected by drones and an adversarial adaptation framework was proposed. Furthermore, in [37], the problem of open-set domain adaptation has been also investigated. Choi *et al.* [37] introduced an attention mechanism to determine discriminative clips and used this information for video-level alignment within an adversarial learning framework. In [117], a domain adaptation approach based on self-supervision and multimodal learning (RGB+optical flow) was proposed for fine-grained first-person view action recognition. RGB+optical flow modalities were also exploited in [149] within a contrastive approach. Whereas [149] considers positives using the

same data on another modality and negatives by perturbing the frames temporally, we consider positives and negatives in different ways: using the real labels for a source only contrastive loss, using real and pseudo-labels for an across domain contrastive loss, and using different augmentations for a target only contrastive. None of these previous works considers a dual-head contrastive framework for learning and aligning source and target video representations.

**UDA benchmarks for action recognition.** Table 4.1 provides an overview of the publicly available benchmarks for UDA and video action recognition along with the previous UDA methods that have considered them. Only three datasets, *HMDB* $\leftrightarrow$ *UCF* [27], *Kinetics* $\rightarrow$ *NEC Drone* [36] and *UCF* $\leftrightarrow$ *Olympic Sports* [27] are available in a third-person view setting. Additionally, two other datasets for domain adaptation in a first-person view setting have been introduced, EPIC Kitchens [44] DA, and, in the hybrid first-person/third-person view settings, Charades-Ego dataset [146]. However, first-person and third-person views setting are quite different in terms of visual appearance. The Jester dataset used in [122] has not been publicly released, whereas the Gameplay dataset considered in [27] only addresses the real $\rightarrow$ synthetic scenario. As shown in Table 4.1, the proposed *Mixamo* $\rightarrow$ *Kinetics* dataset is significantly larger than the existing benchmarks. Furthermore, it can be easily extended in the future by generating more synthetic data. There are other synthetically-generated datasets for action recognition, such as SURREAL [171], SURREACT [172] and 3DPeople [130]. However, all of them render synthetic humans over a static photo background. Furthermore, they have not been generated with the purpose of UDA, and the overlap in terms of categories with existing datasets of real videos, *e.g.* Kinetics [147], is very limited.

Table 4.1: UDA benchmarks for video action recognition

| **Dataset** | **# classes** | **# videos** | $1^{st}$ person | $3^{rd}$ person | **Methods** |
|---|---|---|---|---|---|
| *HMDB ↔ UCF* | 12 | 3,209 | | ✓ | [37] [27] [122] |
| *Kinetics ↔ NEC Drone* | 7 | 994 | | ✓ | [36] [37] |
| *UCF ↔ Olympic Sports* | 6 | 1,145 | | ✓ | [27] [122] |
| *Charades-Ego dataset* | 157 | 4,000 | ✓ | ✓ | [146] [36] |
| *EPIC Kitchens DA* | 8 | $\sim 8,500$ | ✓ | | [117] |
| Mixamo → Kinetics | 14 | 36,195 | | ✓ | This work |

## 4.3 Mixamo → Kinetics dataset

Synthetically generated images and videos are nowadays recognised as an important resource in the computer vision community and are widely used in many tasks. By using computer graphics software and simulators, it is possible to generate large-scale datasets with virtually infinite visual variability and with annotations readily available. However, when models are trained on synthetic data but tested on images and videos from the real world, the problem of domain shift naturally arises. This section describes *Mixamo → Kinetics*, the first large scale dataset for benchmarking domain adaptation methods for action recognition in the challenging task of transferring knowledge from the synthetic to the real domain. Our dataset comprises $36,195$ videos, divided into 14 action categories and two domains, *i.e.*, the source domain (synthetic videos from Mixamo) and the target domain (real videos from Kinetics).

**Source dataset (Mixamo).** It consists of $24,533$ videos synthetically generated using the 3D characters from Mixamo. The dataset comprises videos depicting actions performed by 6 distinct avatars, with different backgrounds, camera positions and random 3D objects in the scene. Also, key-points are provided for each character following the scheme from the COCO dataset [100], but without the key-points for eyes and ears. Each

frame is generated with a resolution of 512 by 512 and the mean length of the videos is 138 frames. To generate each video, we first created a Blender graphic environment; subsequently, for each video, the background and floor images were randomly selected from a set of ∼200 natural images and geometric patterns (*e.g.*, wood floor or other tiling patterns), all collected from the web. Different images were chosen for the background and the floor to avoid the unnatural effect of a "floating" avatar. We further enriched each scene with random 3D objects of varying shape, size and position. Since the objects were positioned using a predefined reference grid around the character, partial occlusions could be performed without the risk of completely hiding the character. Finally, we added to the scene a static sun-like light source and rendered it from 8 different camera angles. Using the light source, it was possible to produce realistic shadows for both the characters and the 3D objects in the scene, which is not possible on datasets that simply place a character in front of a background image. We also plan on generating a larger version of the dataset, with more camera angles and moving light source.

**Target dataset (Kinetics)**. The target dataset was created considering 11,662 videos from 14 action categories extracted from the Kinetics dataset [147]. The overlapping actions between the two datasets are *swing dancing*, *breakdancing*, *salsa dancing*, *throwing*, *capoeira*, *jogging*, *shouting*, *side kick*, *clapping*, *texting*, *golf putting*, *squat*, *punching* and *backflip*. Additional details about the dataset are provided in the supplementary material.

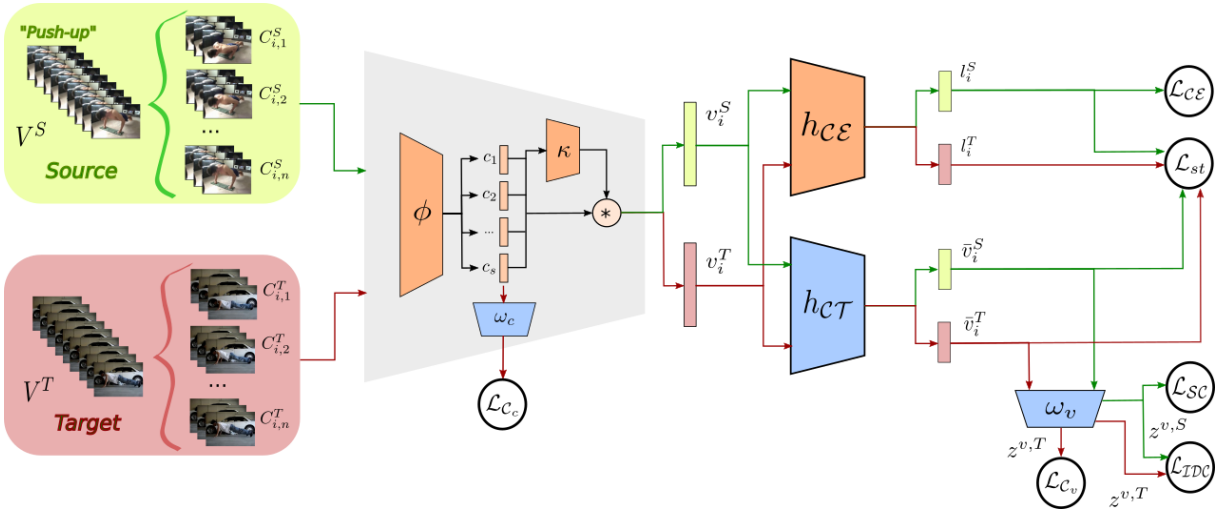



Figure 4.2: Overview of the proposed CO$^2$A approach.

## 4.4 UDA for Action Recognition

### 4.4.1 Problem and Notation

The problem of UDA for action recognition can be formalised as follows. Let $\mathcal{X}$ be the sequence of frames from videos and $\mathcal{Y}$ the set of action categories. Given a labelled source domain $\mathcal{S}$ and an unlabelled target domain $\mathcal{T}$, the aim is to learn a function $f_\Theta : \mathcal{X} \to \mathcal{Y}$, where $\Theta$ denotes a model's parameters, that successfully predicts the corresponding action category from videos of the target domain. Since no annotation is available for the target domain, the training process leverages information from labelled videos of the source domain. The training set $T = T_\mathcal{S} \bigcup T_\mathcal{T}$ is composed by $N_S$ annotated videos in the source domain $T_\mathcal{S} = \{(V_1^S, Y_1^S), \ldots, (V_{N_S}^S, Y_{N_S}^S)\}$ and $N_T$ unlabelled videos from the target domain $T_\mathcal{T} = \{V_1^T, \ldots, V_{N_T}^T\}$. The main challenge of learning $f_\Theta$ lies in addressing the domain shift, *i.e.* the fact that the data from the two domains are drawn from two different distributions $p_\mathcal{S}(V, Y)$ and $p_\mathcal{T}(V, Y)$ over $\mathcal{X} \times \mathcal{Y}$.

### 4.4.2 Proposed Architecture

**Overview.** An overview of the proposed architecture is illustrated in Figure 4.2. First, source and target videos are divided into $K$ non-overlapping parts of equal size, denominated clips. For simplicity, we omitted the fact that we used a minibatch of data and augmented target data. More formally, a video $V_i$ is divided into $K$ clips $C_{i,1}$, ..., $C_{i,K}$ consisting of evenly spaced frames, which are fed to an encoder network $\phi(\cdot)$ that produces clip-level features $\boldsymbol{c}_{i,j} = \phi(C_{i,j})$. From here onward, the video indexes $i$ are omitted for simplicity ($\boldsymbol{c}_j$ indicates $\boldsymbol{c}_{i,j}$). At this level, a self-supervised contrastive loss $\mathcal{L}_{\mathcal{C}_c}$ is applied to learn clip-level feature representations. Following [28], we apply a projection head before computing the contrastive loss. In practice, the clip-level features $\boldsymbol{c}_j$ are passed through a module that outputs $\boldsymbol{z}^{\boldsymbol{c}_j} = \omega_c(\boldsymbol{c}_j)$ to which the contrastive loss is applied.

To produce video representations, clip features $\boldsymbol{c_j}$ are aggregated into video-level feature $\boldsymbol{v} = \sum_{j=1}^{K} \alpha_j \boldsymbol{c}_j$, where the weight vector $\boldsymbol{\alpha} \in \mathbb{R}^K$ is computed using a simple attention module $\kappa(\cdot)$, implemented as multilayer perceptron (MLP) that receives $K$ clip feature vectors as input, *i.e.* $\boldsymbol{\alpha} = \kappa(\boldsymbol{c}_1, \ldots, \boldsymbol{c}_K)$. Subsequently, $\boldsymbol{v}$ is fed to two separate network branches, with a similar base structure, implementing our two-headed architecture. In the first branch, the classification head $h_{\mathcal{CE}}(\cdot)$ produces logits $\boldsymbol{l} = h_{\mathcal{CE}}(\boldsymbol{v})$. In the second branch, $\boldsymbol{v}$ is first provided as input to the contrastive head $h_{\mathcal{CT}}(\cdot)$ to produce features $\bar{\boldsymbol{v}} = h_{\mathcal{CT}}(\boldsymbol{v})$, and then to a projection head $\omega_v$, resulting in a latent vector $\boldsymbol{z}^{\boldsymbol{v}} = \omega_v(\bar{\boldsymbol{v}})$. The first head is trained using a cross-entropy loss $\mathcal{L}_{\mathcal{CE}}$, whereas the second head uses a combination of three contrastive losses: a supervised contrastive loss $\mathcal{L}_{\mathcal{SC}}$ [86] for source data, a self-supervised contrastive loss $\mathcal{L}_{\mathcal{C}_v}$ to learn better video-level representations for the target data and a class-aware inter-domain contrastive loss $\mathcal{L}_{\mathcal{IDC}}$ that aligns the distributions of the features of both

domains. Finally, the stability loss $\mathcal{L}_{ST}$ enforces an agreement between the predictions of the two main network heads $h_{\mathcal{CE}}(\cdot)$ and $h_{\mathcal{CT}}(\cdot)$.

Our network is trained on minibatches of size $3M$, *i.e.* composed of $M^S$ randomly chosen source videos, $M^T$ randomly chosen target videos and their $M^T$ augmented versions. So we end up with $M^S$ source videos and $2M^T$ target videos. In practice, data augmentations are only used for target data to train the unsupervised contrastive loss terms.

Next, we describe in detail the main components of our approach: (i) the proposed multi-scale feature representations, where we used self-supervised contrastive learning to compute both video-level and clip-level features; (ii) our novel contrastive domain alignment loss $\mathcal{L}_{\mathcal{IDC}}$ and (iii) our novel dual-head classifier structure.

**Multi-scale Contrastive Video Feature Learning.** Clip-level and video-level features carry different information about a video [115]. While the first ones focus on sub-parts of an action, the second ones are intended to represent the complete action. This work proposes to leverage the complementary clip-level and video-level information to learn representations on the target domain within a contrastive learning framework. Specifically, we resort to data augmentation on the target domain and, inspired by recent works on video representation learning [131], we define two self-supervised contrastive loss terms, one at video-level and the other at clip-level.

At the video-level, we use the output of the projection head $\omega_v$, which produces the projected video-level features $\boldsymbol{z^v}$, and define the loss:

$$\mathcal{L}^i_{\mathcal{C}_v} = -\sum_{j=1}^{2M^T} \mathbb{1}_{\varphi_{i,j}} \cdot \log \frac{s^{\boldsymbol{z_i^v}, \boldsymbol{z_j^v}}}{\sum_{p=1}^{2M^T} \mathbb{1}_{i \neq p} \cdot s^{\boldsymbol{z_i^v}, \boldsymbol{z_p^v}}}. \tag{4.1}$$

where $s^{\boldsymbol{z}_i, \boldsymbol{z}_j} = \exp(\frac{\boldsymbol{z}_i \cdot \boldsymbol{z}_j / \tau}{||\boldsymbol{z}_i|| \cdot ||\boldsymbol{z}_j||})$, $\tau > 0$ is a temperature parameter, $\mathbb{1}$ is an indicator function which is 1 if its argument is true or 0 otherwise, and

$\varphi_{i,j}$ is true if $i$ and $j$ are two different augmentations of the same video. In practice, this loss has the effect of pulling together representations in the embedding space of augmented versions of the same target video (positive samples), while pushing away those associated with different videos in the same mini-batch (negative samples).

Similarly, at clip-level, we use the output of the projection head $\omega_c$, which produces the projected clip-level features $\boldsymbol{z}^{\boldsymbol{c_j}}$, and define the loss:

$$\mathcal{L}^i_{\mathcal{C}_c} = -\frac{1}{K}\sum_{j=1}^{2M^T}\sum_{u=1}^{K}\mathbb{1}_{\varphi_{i,j}}\log\frac{s^{\boldsymbol{z_i^{cu}},\boldsymbol{z_j^{cu}}}}{s^{\boldsymbol{z_i^{cu}},\boldsymbol{z_j^{cu}}}+\mathrm{Neg}_{i,u}}, \tag{4.2}$$

where $\mathrm{Neg}_{i,u} = \sum_{p=1}^{2M^T}\mathbb{1}_{p\neq i, p\neq j}\sum_{v=1}^{K} s^{\boldsymbol{z_i^{cu}},\boldsymbol{z_p^{cv}}}$ is the set of negatives for instance $i$ and clip $u$. In practice, this loss considers different augmentations of the same clip as positives and clips from different videos as negatives. Selecting different clips from the same video to form the set of negatives, in fact, could be harmful, since an action may span over multiple clips. The final self-supervised contrastive loss on target data is:

$$\mathcal{L}^i_{\mathcal{C}} = \mathcal{L}^i_{\mathcal{C}_c} + \mathcal{L}^i_{\mathcal{C}_v} \tag{4.3}$$

It is worth noting that the two proposed losses are complementary since they use different notions of positives and negatives and operate at different temporal resolutions. Our experimental results (Sec. 4.5) demonstrate the benefit of our multi-scale self-supervised video representation learning strategy.

**Contrastive Domain Alignment.** To specifically address the domain shift problem and perform feature alignment of the source and target data, we introduce a novel inter-domain contrastive loss. Previous works on supervised contrastive learning [86] introduced a loss that has the effect of pulling together in the embedding space samples belonging to the same class while pushing far apart samples from different classes. In practice, in

[86], positive and negative samples are obtained by only considering label information. In this work, we propose to revisit this idea by jointly combining samples from the two domains. However, while each source video $V^S$ has an associated label $Y^S$, for the unsupervised target videos, we compute pseudo-labels $\tilde{Y}^T = \text{argmax } \sigma(\boldsymbol{l}^T)$, where $\sigma$ denotes the softmax operator. We employed a simple pseudo-labelling strategy, but other more complex strategies could be used. Therefore, we propose to consider as positives, instances from different domains that share the same label/pseudo-label, while instances with different labels/pseudo-labels and different domains are regarded as negatives. In this way, feature alignment among different domains is realised, while also taking into account semantic information. Formally, our proposed inter-domain contrastive loss is defined as:

$$\mathcal{L}^i_{\mathcal{IDC}} = -\frac{1}{\gamma_i} \sum_{j=1}^{3M} \mathbb{1}_{\rho_{i,j}} \cdot \mathbb{1}_{\tilde{Y}_i = \tilde{Y}_j} \cdot \log \frac{s^{\boldsymbol{z_i^v}, \boldsymbol{z_j^v}}}{\sum_{p=1}^{3M} \mathbb{1}_{\rho_{i,p}} \cdot s^{\boldsymbol{z_i^v}, \boldsymbol{z_p^v}}}, \tag{4.4}$$

where $\mathbb{1}_{\rho_{a,b}} = \mathbb{1}_{a \notin \Omega} \mathbb{1}_{b \notin \Omega} \mathbb{1}_{D(a) \neq D(b)}$, $\gamma_i$ is the number of positives for instance $i$ and $D(\cdot)$ is a function that returns the domain of an instance. $\Omega$ denotes the set of target instances for which pseudo-labels are not considered reliable. Note that alternative losses such as those based on domain discrepancy minimisation and adversarial learning are designed to encourage the network to produce domain-agnostic features, whereas our proposed inter-domain contrastive loss encourages the network to produce tight representations and exploits label/pseudo-label information to push together features belonging to the same class and pull apart those belonging to different classes. To limit the effect of noise that is typically present in pseudo-labels, we propose to employ a simple sample filtering procedure: target instances are added to $\Omega$ when $H(\sigma(\boldsymbol{l}_i^T)) > log(n_{classes})/\eta$, where $H$ is an entropy function and $\eta$ a user-defined parameter that represents the percentage of the maximum allowed entropy.

**Two-headed network.** As discussed above, we design an architecture with two different heads. Each head is supervised with different losses. The first head, $h_{\mathcal{CE}}$, is mainly trained with a cross-entropy loss on source instances, *i.e.*:

$$\mathcal{L}^i_{\mathcal{CE}} = -\sum Y^S_i \log \sigma(\boldsymbol{l_i}). \tag{4.5}$$

Differently, the second head, $h_{\mathcal{CT}}$, is trained using the contrastive losses. Besides the previously described $\mathcal{L}^i_{\mathcal{C}_v}$ and $\mathcal{L}^i_{\mathcal{IDC}}$, we also introduce a supervised contrastive loss [86]:

$$\mathcal{L}^i_{\mathcal{SC}} = -\frac{1}{\gamma_i}\sum_{j=1}^{M} \mathbb{1}_{i\neq j} \cdot \mathbb{1}_{\tilde{Y}_i=\tilde{Y}_j} \cdot \log \frac{s^{\boldsymbol{z^v}_i,\boldsymbol{z^v}_j}}{\sum_{p=1}^{M^S} \mathbb{1}_{i\neq p} \cdot s^{\boldsymbol{z^v}_i,\boldsymbol{z^v}_p}}. \tag{4.6}$$

By leveraging from source supervision in a different way, the two heads promote the learning of different feature representations. To maximally benefit from this dual head structure, we introduce an additional loss term that enforces coherence between the predictions of the two heads. Besides stabilising the training of both heads, it makes additional information flow directly from the $h_{\mathcal{CT}}$ to $h_{\mathcal{CE}}$ and vice versa. As the contrastive head operates on instance pairs, we propose to define this coherence loss considering pairwise predictions associated to pairs of source and target videos in the minibatch, as follows:

$$\mathcal{L}^{i,j}_{\mathcal{ST}} = q_{i,j}\log(p_{i,j}) + (1-q_{i,j})\log(1-p_{i,j}) \tag{4.7}$$

Here $p_{i,j} = \boldsymbol{l_i}\boldsymbol{l_j}^\top$ denotes the pairwise predictions computed on source video $i$ and target video $j$ using of the logits produced by $h_{\mathcal{CE}}$. Similarly, $q_{i,j}$ indicates the binary prediction label which is computed though $h_{\mathcal{CT}}$ as:

$$q_{i,j} = \begin{cases} 1, & \text{if } \cos(\overline{\boldsymbol{v}}_i, \overline{\boldsymbol{v}}_j) > \theta \\ 0, & \text{otherwise} \end{cases} \tag{4.8}$$

where $\cos(\cdot,\cdot)$ indicates the cosine similarity between two vectors and $\theta$ is a threshold that we set equal to 0.5.

**Overall Loss.** The whole model is trained by combining the losses and weighting them accordingly as follows:

$$\mathcal{L} = \frac{1}{M}\sum_{i=1}^{M} w_{ce}\mathcal{L}^i_{\mathcal{CE}} + w_{sc}\mathcal{L}^i_{\mathcal{SC}} + \frac{1}{\mu}\sum_{i=1}^{3M} w_{idc}\mathcal{L}^i_{\mathcal{IDC}} + \frac{1}{2M}\sum_{i=1}^{2M} w_c\mathcal{L}^i_{\mathcal{C}} + \frac{1}{2M^2}\sum_{i=1}^{2M}\sum_{j=1}^{M} w_{st}\mathcal{L}^{i,j}_{\mathcal{ST}}, \tag{4.9}$$

where $\mu$ is the number of instances with at least one positive.

**Inference.** At inference time, the projection heads $\omega_c$ and $\omega_v$, and $h_{\mathcal{CT}}$ are discarded. Data is only forwarded through the shared backbone and $h_{\mathcal{CE}}$, producing the logits $\boldsymbol{l}$ that are further normalised by a softmax function to generate the classes probabilities. It is worth noting that while at training time the addition of the double head implies an increase in terms of parameters, during inference, since the $\omega_c$, $\omega_v$ and $h_{\mathcal{CT}}$ are discarded, the number of parameters is the same as if we used a single head architecture.

## 4.5 Experimental Results

### 4.5.1 Setup

**Datasets.** We conduct experiments on three standard UDA benchmarks for action recognition: *UCF* ↔ *HMDB* [27], *UCF* ↔ *Olympic Sports* [27], *Kinetics* → *NEC-Drone* [36], and on our newly proposed *Mixamo* → *Kinetics*. In *UCF* ↔ *HMDB* and *UCF* ↔ *Olympic Sports*, the domain shift is caused by varying visual appearance, lighting, camera viewpoint, etc. However, source and target domains are both associated with videos depicting real scenes. In the *Kinetics* → *NEC-Drone*, the domain shift is large as data of the source domain consist of Youtube videos, while the target domain comprises videos taken from a camera installed on a drone.

Lastly, the *Mixamo* $\rightarrow$ *Kinetics* dataset presents the most severe domain shift, comprising synthetically generated video sequences in the source domain and Youtube videos in the target.

**Baselines.** We compare with three state-of-the-art UDA methods for action recognition: (i) TA$^3$N [27], considering both the original implementation of the 2D encoder (from [198]) and the adapted 3D version (similar to [37]) using the I3D [22] backbone; (ii) TCoN [122], considering the Resnet101 architecture as backbone, and (iii) SAVA [37], which employs I3D as clip feature extractor. We did not compare with [149] because their approach combines RGB with optical flow information, whereas ours solely uses RGB. Results are also reported for each backbone considering the following settings: (i) *supervised source only*, when the network is trained only with supervised source data, and (ii) *supervised target only*, when the network is trained (fine-tuned) with supervised target data. These settings correspond respectively to a lower and an upper bound for UDA methods. Note that while code for TA$^3$N [27] is publicly available, we did not find implementations for TCoN [122] and SAVA [37]. The associated results are taken from the original papers. Methods are compared in terms of *Top-1 Accuracy*.

**Implementation details.** We employ an I3D architecture as backbone network to be comparable with our closest competitor [37]. $\phi(\cdot)$ since . The I3D is pretrained on Kinetics for all datasets, except on the *Mixamo* $\rightarrow$ *Kinetics* where it is initialised by inflating the weights from an Imagenet-pretrained Inception-v1 network, as in [22], and fine-tuned on Mixamo labelled data. We implemented $\kappa$ as MLP with architecture *Linear/ReLU/Linear/Sigmoid* that receives as input $K = 4$ clips, following [37]. The two heads $h_{\mathcal{CE}}(\cdot)$ and $h_{\mathcal{CT}}(\cdot)$ are both implemented as 2-layers MLPs with ReLU activation and without BatchNorm layers, with the only difference that a linear classifier is appended to $h_{\mathcal{CE}}(\cdot)$. Both take as input

the video-level features, but $h_{\mathcal{CE}}(\cdot)$ outputs a vector of size equal to the number of classes and $h_{\mathcal{CT}}(\cdot)$ a vector of size 256. The projection heads $\omega_v$ and $\omega_c$ are both implemented as *Linear/ReLU/Linear* with output 128. The input space of $\omega_v$ is 256, whereas in $\omega_c$ it is 1024.

We perform video-based data augmentations on target data. Following [131], given a video, the same transformation is applied to all frames coherently. Colour, spatial and random horizontal flip augmentations are considered. Additional details about augmentations are reported in the supplementary material.

Hyper-parameters selection is performed following a common protocol in UDA literature [145], *i.e.* by selecting a subset (here, 5 annotated videos per class) in the target training set and by using them as validation set. On *HMDB* $\leftrightarrow$ *UCF* and *UCF* $\leftrightarrow$ *Olimpic*, the losses weights are $w_{ce} = 1$, $w_{sc} = 1$, $w_{idc} = 1.5$, $w_c = 0.2$ and $w_{st} = 0.02$; for *Kinetics* $\rightarrow$ *NEC-Drone* we set $w_{idc} = 0.2$, $w_c = 1.2$ and $w_{st} = 0.01$ and for *Mixamo* $\rightarrow$ *Kinetics* we set $w_{idc} = 2$, $w_c = 0.2$ and $w_{st} = 0.02$. All values were found using grid-search. We trained our network with SGD with learning rate 0.02, momentum of 0.9 and weight decay of $1^{-9}$. We set $\eta = 6$ for *Mixamo* $\rightarrow$ *Kinetics* and $\eta = 4$ for the other experiments. Experiments were carried out on 4 Nvidia RTX 5000 GPUs for around 1 hour *HMDB* $\leftrightarrow$ *UCF* and *UCF* $\leftrightarrow$ *Olimpic*, 3 hours for *Kinetics* $\rightarrow$ *NEC-Drone* and 4 hours for *Mixamo* $\rightarrow$ *Kinetics*.

### 4.5.2 Results

**Comparison with state of the art.** We first report the results obtained by comparing our approach with state-of-the-art methods. Tables 4.2, 4.3 and 4.4 show the results of our experiments on *HMDB* $\leftrightarrow$ *UCF*, *UCF* $\leftrightarrow$ *Olympic Sports* and *Kinetics* $\rightarrow$ *NEC-Drone*, respectively. In all tables,

Table 4.2: Results on *UCF* ↔ *HMDB*

| **Method** | **Encoder** | **U → H** | **H → U** |
|---|---|---|---|
| Supervised source only [27] | Resnet101-TRN | 71.7 | 73.9 |
| DANN [61] | | 75.2 | 76.3 |
| JAN [105] | | 74.7 | 79.6 |
| AdaBN [98] | | 72.2 | 77.4 |
| MCD [140] | | 73.8 | 79.3 |
| TA$^3$N [27] | | 78.3 | 81.8 |
| Supervised target only [27] | | 82.8 | 94.9 |
| Supervised source only [33] | I3D-based TRN | 80.6 | 88.8 |
| TA$^3$N [27] | | 81.4 | 90.5 |
| Supervised target only [27] | | 93.1 | 97.0 |
| Supervised source only [37] | I3D | 80.3 | 88.8 |
| SAVA [37] | | 82.2 | *91.2* |
| Supervised target only [37] | | 95.0 | 96.8 |
| TCoN [122] | 2D/3D CNN | *87.2* | 89.1 |
| CO$^2$A | I3D | **87.8** | **95.8** |

the best results are indicated in bold and the second-best in italic.

As shown in Table 4.2, our approach outperforms all previous methods for the *HMDB* ↔ *UCF* setting. In particular, it achieves an accuracy of 87.8% for U → H and 95.8% for H → U, outperforming its best competitor, SAVA [37] with the same I3D backbone, by 5% and almost 4%, respectively. Notably, all recent UDA methods specifically designed for action recognition, *i.e.* TA$^3$N [27], TCoN [122], SAVA [37] and our method, significantly outperform traditional image-based UDA approaches, *i.e.* DANN [61], JAN [105], AdaBN [98] and MCD [140].

Similar observations can be made by looking at results in Table 4.3. Our

Table 4.3: Results on *UCF* ↔ *Olympic Sports*

| **Method** | **U → OS** | **OS → U** |
|---|---|---|
| W. Sultani *et al.* [152] | 33.3 | 47.9 |
| T. Xu *et al.* [185] | 87.0 | 75.0 |
| AMLS (GFK) [82] | 84.6 | 86.4 |
| AMLS (SA) [82] | 83.9 | 86.0 |
| DAAA [82] | 91.6 | 89.9 |
| TA$^3$N [27] (Resnet101-TRN) | *98.2* | 92.9 |
| TCoN [122] (Resnet101-TRN) | 96.8 | *96.7* |
| SAVA [37] (I3D) | 98.1 | *96.7* |
| CO$^2$A (I3D) | **100** | **97.5** |

method outperforms the best competing methods, *i.e.* TA$^3$N [27] on *UCF* → *Olympic Sports* and TCoN [122] and SAVA [37] on *Olympic Sports* → *UCF*. Again, modern UDA methods for action recognition, *i.e.* TA$^3$N [27], TCoN [122], SAVA [37] and CO$^2$A, are significantly more accurate than traditional techniques [82, 152, 185].

Finally, Table 4.4 reports the results obtained in the more challenging *Kinetics* → *NEC-Drone* setting. The gap in performance between *supervised source only* (lower bound) and *supervised target only* (upper bound) indicates a domain shift that is significantly more pronounced than that observed in the *HMDB* ↔ *UCF* and the *UCF* ↔ *Olympic Sports* datasets. Even in this challenging setting, our approach outperforms state-of-the-art methods. In particular, the accuracy of CO$^2$A is 1.6% higher than its best competitor SAVA [37].

**Results on the *Mixamo* → *Kinetics* dataset.** Table 4.5 shows the results obtained on our newly proposed *Mixamo* → *Kinetics* dataset. This setting is much more challenging than previous ones, not only due to the

large domain gap but also because it contains more action categories than *Kinetics* $\rightarrow$ *NEC-Drone* (14 classes versus 7) and previous benchmarks. For this dataset, we only consider baseline methods for which the code is publicly available, *i.e.* TA$^3$N [27]. Additionally, we run a previous image-based UDA approach, *i.e.* ADDA [167]. Due to the intrinsic difficulty of the *Mixamo* $\rightarrow$ *Kinetics* dataset, it is not surprising that all the methods achieve lower performance than in other settings. Still, our approach sets the state-of-the-art, outperforming its best competitor TA$^3$N [27].

The table also reports the performance of CO$^2$A and baselines considering a weakly supervised setting, *i.e.* assuming that annotations are available for 5 randomly selected target instances per class. As shown in the table, our method again outperforms the competitors. The table additionally reports, as an upper bound, the score of the *supervised target only* method, which considers annotations available on the entire target training set. The large gap between the performance of UDA methods and the upper bound encourages further research on this challenging synthetic-to-real UDA setting that we introduced with this paper.

**Ablation Study.** We also perform an ablation study to assess and empirically demonstrate the importance of our technical contributions. Table 4.6 reports the results of a set of experiments conducted to analyze the role of the distinct losses employed in our framework. The ablation experiments consider the *HMDB* $\leftrightarrow$ *UCF* and *Kinetics* $\rightarrow$ *NEC-Drone* datasets and report results obtained by disabling one loss at the time. Looking at Table 4.6 the following observations can be made: (i) The scores obtained with the full model (last line of the table) show that the different losses are complementary and the model achieves the best results when combining all of them; (ii) the inter-domain loss $\mathcal{L}_{\mathcal{IDC}}$ is beneficial in all the settings, enabling to reduce the domain shift by promoting domain distribution alignment; (iii) the heads consistency loss $\mathcal{L}_{\mathcal{ST}}$ provides a significant

Table 4.4: Results on *Kinetics* → *NEC-Drone*.

| **Method** | **Encoder** | **Top-1 acc** |
|---|---|---|
| Supervised source only [27] | ResNet-101-based TRN | 15.8 |
| TA$^3$N [27] | ResNet-101-based TRN | 25.0 |
| Supervised source only [27] | I3D-based TRN | 15.8 |
| TA$^3$N [27] | I3D-based TRN | 28.1 |
| Supervised source only [37] | I3D | 17.2 |
| DANN [61] | I3D | 22.3 |
| ADDA [167] | I3D | 23.7 |
| Choi et al. [36] (on val set) | I3D | 15.1 |
| SAVA [37] | I3D | *31.6* |
| Supervised target only | I3D | 81.7 |
| CO$^2$A | I3D | **33.2** |

benefit in term of performance in the most challenging setting, *i.e.* in the *Kinetics* → *NEC-Drone* setting: the accuracy drops by 6% when this loss is disabled; (iv) disabling the clip-level and video-level losses is also detrimental for performance, with different performance among datasets. This suggests that both video-level and clip-level information are important to describe action videos. Lastly, (vi) disabling the supervised contrastive loss greatly reduces the performance on *HMDB* →*UCF* and *Kinetics* → *NEC-Drone*, which is related to the fact that the second head is not performing on par on source data.

Figure 4.3 shows the results of a sensitivity analysis of our model concerning the weights associated to $\mathcal{L}_{\mathcal{IDC}}$ and $\mathcal{L}_{\mathcal{SC}}$.The sensitivity analysis for the weights of $\mathcal{L}_{\mathcal{C}_c}$, $\mathcal{L}_{\mathcal{C}_v}$ and $\mathcal{L}_{\mathcal{ST}}$ are provided in the supplementary material due to lack of space. We considered the *HMDB* ↔ *UCF* setting and we show that both losses are beneficial for the final score when the optimal

Table 4.5: Results on *Mixamo* → *Kinetics*.

| Method | Weak Supervision | Encoder | Top-1 acc |
|---|---|---|---|
| Supervised source only | | I3D | 11.2 |
| ADDA [167] | | I3D | 11.0 |
| TA$^3$N [27] | | Resnet101-TRN | 7.0 |
| TA$^3$N [27] | | I3D-based TRN | 10.0 |
| ADDA [167] | ✓ | I3D | 17.0 |
| TA$^3$N [27] | ✓ | Resnet101-TRN | 13.0 |
| TA$^3$N [27] | ✓ | I3D-based TRN | 19.1 |
| Supervised target only | ✓ | I3D | 79.3 |
| CO$^2$A | | I3D | **16.4** |
| CO$^2$A | ✓ | I3D | **20.1** |

values of their weights are set. Figure 4.3 (left) shows that a value of $w_{idc}$ equal to zero corresponds to the worst performance since no domain distribution alignment takes place. Figure 4.3 (right) shows that a value of $w_{sc}$ equal to zero, corresponding to no supervision provided on the contrastive head, is suboptimal and performance can be improved if supervision on both network heads is provided. Similarly, using a high weight for $\mathcal{L}_{\mathcal{SC}}$ implies relying too strongly on source data and losing the benefit of the

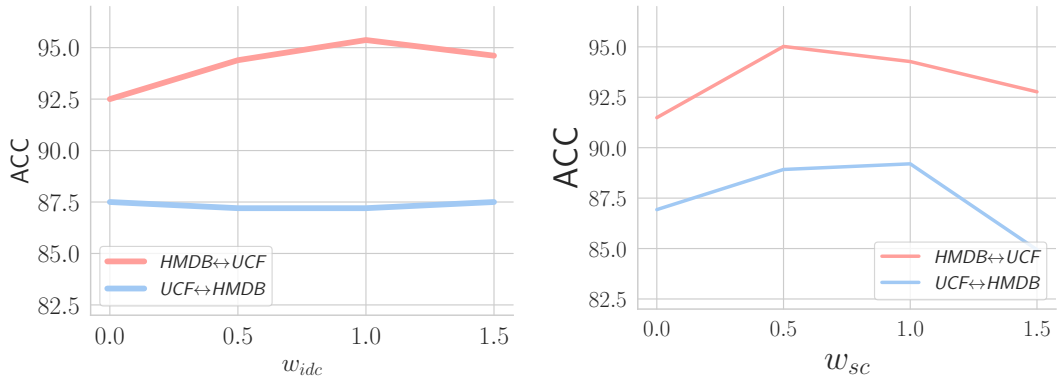



Figure 4.3: Sensitivity analysis of the weights of the losses $\mathcal{L}_{\mathcal{IDC}}$ and $\mathcal{L}_{\mathcal{SC}}$.

Table 4.6: Ablation study on *HMDB* $\leftrightarrow$ *UCF* and *Kinetics* $\rightarrow$ *NEC-Drone*: importance of different losses.

| **Method** | **H $\rightarrow$ U** | **U $\rightarrow$ H** | **K $\rightarrow$ N-D.** |
|---|---|---|---|
| CO$^2$A w/o $\mathcal{L}_{\mathcal{IDC}}$ | 92.5 | 87.5 | 30.9 |
| CO$^2$A w/o $\mathcal{L}_{\mathcal{ST}}$ | 94.4 | 82.4 | 27.0 |
| CO$^2$A w/o $\mathcal{L}_{\mathcal{C}_c}$ | 91.9 | 85.5 | 29.6 |
| CO$^2$A w/o $\mathcal{L}_{\mathcal{C}_v}$ | **95.8** | 81.5 | 24.8 |
| CO$^2$A w/o $\mathcal{L}_{\mathcal{SC}}$ | 91.5 | 86.9 | 28.1 |
| CO$^2$A (full) | **95.8** | **87.8** | **33.2** |

losses applied to the target data. Additional results are provided in the supplementary material, due to lack of space.

## 4.6 Conclusions

We presented CO$^2$A, a novel UDA approach for video action recognition that explores conditional feature alignment across domains within a contrastive learning framework. Our approach achieved state-of-the-art performance on three publicly available UDA benchmarks. Moreover, we introduced the novel large-scale dataset *Mixamo* $\rightarrow$ *Kinetics*. This new benchmark will foster future research in domain adaptation from synthetic to real video sequences. In the future, we plan to improve our approach by integrating more sophisticated methods for obtaining reliable pseudo-labels. Additionally, we plan to extend *Mixamo* to include videos generated with random light source position and strength and with moving cameras.

# Chapter 5

# Unsupervised Domain Adaptation for Video Transformers in Action Recognition

## 5.1 Introduction

The standard setting for visual learning tasks relies on the assumption that training and testing data belong to the same domain, i.e., they are drawn from the same distribution. However, in practice, this often does not hold, as many real-world scenarios require models to be tested on very different, yet related, domains. This often leads to poor performance in the domain of interest, especially when the domain shift is significant. To address this issue, Unsupervised Domain Adaptation (UDA) techniques have been developed with several approaches achieving remarkable results on image-related tasks. However, much less attention has been devoted to videos, which poses a greater challenge given the significant increase in complexity that arises when the temporal aspects come into play.

Action recognition [56, 196, 166, 22] is one of the most popular tasks when it comes to video analysis. This task is particularly challenging due

to some inherent characteristics of sequential visual data, mostly related to the significant variability that arises due to the many different ways in which a certain human activity can be carried out, with variations in terms of speed, duration, relative movement between camera and actor(s), occlusion, etc. These issues, along with many others, have been tackled in several previous works [56, 196, 166, 22] that demonstrated remarkable performance exploiting different deep architectures. Nevertheless, many open problems still exist when domain shift arises in video analysis.

In this work, we tackle the problem of UDA in video action recognition by proposing a novel approach that exploits a new information theoretic-based domain alignment loss on top of a transformer-based feature extractor. In particular, our architecture is built upon a shared encoder derived from the STAM [143] visual transformer coupled with a domain alignment loss inspired by the Information Bottleneck (IB) principle [163, 164]. The encoder is composed of a spatial transformer, e.g. ViT [50], that processes individual frames and a temporal transformer, e.g. a simple multi-layer transformer as in [173], that aggregates spatial features to process a whole video. In this paper, we propose a two-phase process for training, where we first fine-tune the whole transformer using only source data as in [107], and then we freeze the spatial transformer and fine-tune only the temporal one with the novel IB loss, which promotes domain distribution alignment. The seamless integration of the transformer with our adaptation strategy leads to improved recognition accuracy (see Sec. 5.4) that stems from a better spatio-temporal action localisation. As an example, Fig. 5.1 depicts a 16-frame sequence and the associated frame-level attention values for a validation video, comparing the model trained only on source data (a) and the same model trained with our UDA approach (b). From the picture, it is possible to see that our method enables the network to better focus on the frames where the handshake action takes place. We called

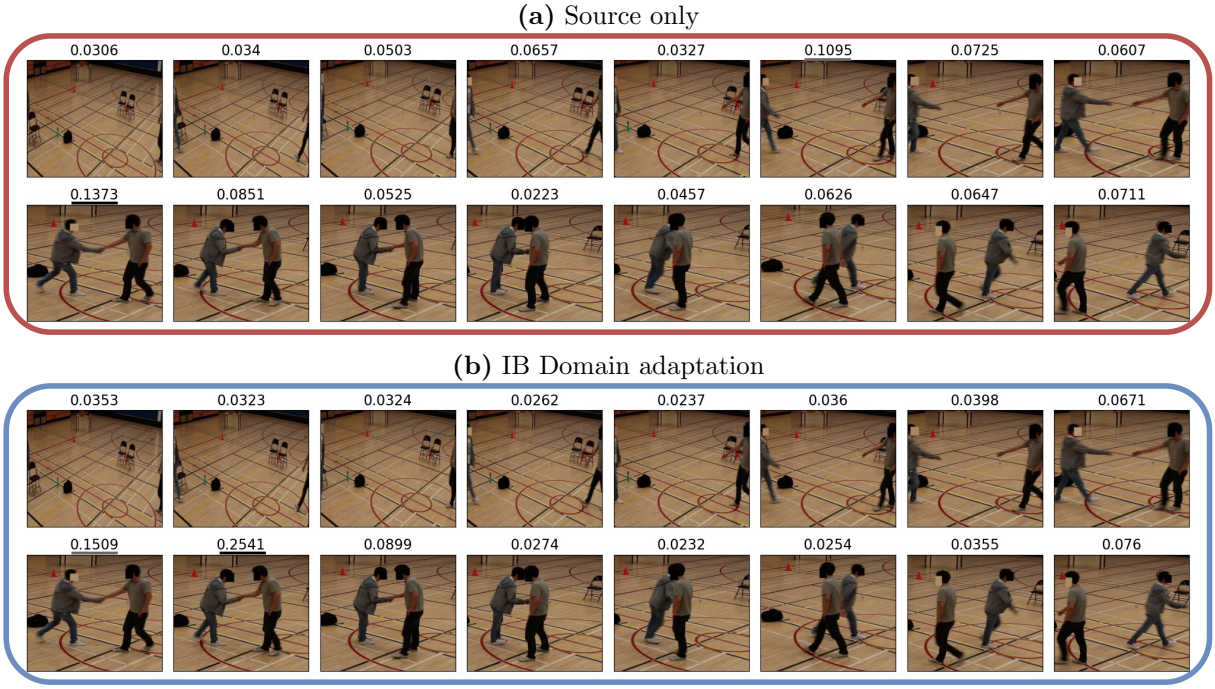



Figure 5.1: Temporal attention visualisation: comparison between a source-only model (a) and UDAVT (b). Most and second most attended frames are highlighted.

our method UDAVT (*Unsupervised Domain Adaptation for Video Transformers in Action Recognition*). We evaluate our approach on two popular UDA benchmarks for action recognition, namely *HMDB* $\leftrightarrow$ *UCF* [27] and *Kinetics* $\rightarrow$ *NEC-Drone* [36], and show that UDAVT outperforms previous UDA methods.

**Contributions.** Our contributions are as follows: (i) we propose a novel UDA approach for video action recognition that exploits class label information within a novel IB domain alignment loss; (ii) we show, for the first time, that a transformer-based feature extractor can be successfully employed in UDA for building a robust source model that is more resilient to domain shift than traditional architectures for video analysis; (iii) we provide an extensive evaluation of our method and show that UDAVT outperforms state-of-the-art UDA approaches on all considered datasets.

## 5.2 Related Work

**Action recognition.** In the last few years, several approaches and architectures have been proposed for video action recognition. For instance, two-stream networks were proposed in [56], jointly exploiting RGB and optical flow sequences. The Temporal Relation Network was proposed by Zhou *et al.* [196] to model temporal relations across the sequence by employing a specific pooling layer. Other works opted for 3D CNNs to learn spatio-temporal features [166, 22]. Recently, some works proposed contrastive-learning-based methods in order to extract efficient motion representations for action recognition [188, 131, 177, 126]. In order to model the temporal aspects, fusion methods have been proposed [114, 3], as well as, combinations of recurrent and convolutional networks [151]. Bai *et al.* [6] introduced an approach based on multi-range feature interchange to capture short-range motion features and long-range dependencies. Finally, advanced transformer-based approaches have been proposed for action recognition [64, 143, 11, 80, 111, 118], often exploiting skeleton points [128, 97, 153, 34, 35]. Different from our work, these approaches consider a standard supervised setting, where all data is labelled and there is no domain shift between training and evaluation datasets.

**UDA for images.** Existing UDA methods differ from the strategy adopted to address the domain shift. One option consists of matching statistical moments of the source and target data distributions [104, 17, 138, 110, 169], eventually integrating label information in the alignment loss [84]. Another common approach is based on adversarial learning [103, 78, 63, 167] and aims at learning discriminative and domain-agnostic feature representations by combining implementing a domain discriminator. Another possibility is to use Generative Adversarial Networks [76, 175, 141] to generate target-like instances from source data that are then employed

to train the model. Furthermore, self-supervised learning has been recently exploited for DA, in particular when devising auxiliary tasks, such as predicting rotations [154] or image patches permutations [14], to learn domain-invariant feature representations. Finally, recent works proposed to employ transformer-based architectures in the scope of domain adaptation [187, 186]. However, all these methods were proposed for images.

**UDA for action recognition** The challenging problem of domain shift for video action recognition has been addressed by a limited number of works [117, 122, 27, 36, 178], despite the several possible applications it finds in real-world scenarios. Chen *et al.* [27] proposed TA$^3$N (*Temporal Attentive Adversarial Adaptation Network*), which employs a temporal relation module to jointly perform alignment between the source and target domains and learn the temporal relation across the video sequences. Pan *et al.* [122] proposed TCoN (*Temporal Co-attention Network*), a deep architecture integrating a cross-domain attention module in order to match the distributions of source and target domain between temporally aligned feature representations. Choi *et al.* proposed two methods based on adversarial learning [36] and on an attention mechanism [178], respectively. Kim *et al.* [89] proposed an UDA approach based on learning cross-modal contrastive features, while in [117] an approach was introduced based on self-supervised and multimodal learning (RGB + optical flow). Song *et al.* [149] proposed a method for spatio-temporal contrastive domain adaptation. Lastly, Turrisi da Costa *et al.* [42] proposed CO$^2$A, an approach for UDA that relies on a contrastive loss to perform domain alignment, in addition to stabilising training by making a classification and a contrastive head agree. Our proposed approach presents similarities with previous self-supervised based frameworks [123, 88], but introduces novelty in employing an Information Bottleneck loss to align domains rather than to learn better feature representations. Additionally, our work is also the first to exploit

transformers for UDA in the context of action recognition.

## 5.3 Proposed method

**Problem setting and notation.** The paper tackles the problem of UDA for action recognition. Given a source dataset $\mathcal{S} = \{X_i^S, y_i^S\}_{i=1}^{N_S}$ of videos and associated annotations, and an unlabelled target dataset $\mathcal{T} = \{X_i^T\}_{i=1}^{N_T}$, where $X_i \in \mathcal{X}$ and $y \in \mathcal{Y}$, $\mathcal{Y} = \{1, 2, \ldots, K\}$ ($K$ denotes the number of action categories), we aim to learn a function $F_\theta : \mathcal{X} \to \mathcal{Y}$ with parameters $\theta$ that maps an input video $X$ to a class label $y$ and perform well on target data. Note that this is not a trivial task, since source and target data are sampled from two different distributions, $P^S(X) \neq P^T(X)$. To tackle this problem, we propose a novel approach called UDAVT that combines two main components: a spatio-temporal transformer architecture and a novel distribution alignment scheme derived from the IB principle [163].

**Overview.** An overview of UDAVT is shown in Fig.5.2. We propose a two-phase training pipeline where the model is first trained with source data and subsequently adapted using source and target data. Our model is defined as $F_\theta = C \circ H$, where $H$ represents a video transformer encoder [143] and $C$ represents a linear classifier. $H$ is composed by two main parts, a spatial transformer $H_s$ that extracts frame-level feature representations and a temporal transformer $H_t$ that aggregates the frame-level features to produce video-level representations. In particular, $H_s$ is the vision transformer ViT [50], whereas $H_t$ is a simple multi-layer transformer as in [173]. An auxiliary MLP projection head $P$ is also used in the second phase. Finally, the complete model also has a queue $Q$ that is responsible for keeping the most recent feature representations of source data. The two main phases of our approach are described as follows.

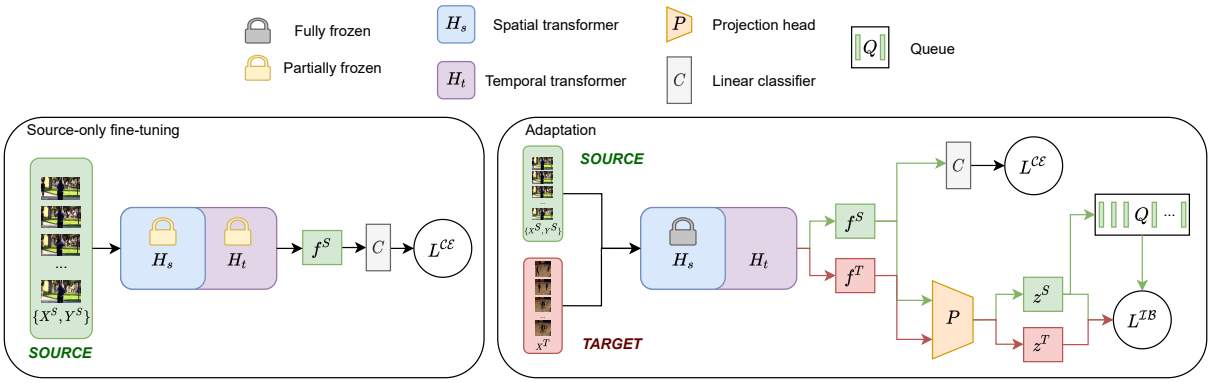



Figure 5.2: **Overview of UDAVT.** Our approach is articulated in two steps. In phase 1 (left), source data are fed to a video transformer $H$ (composed of a spatial transformer $H_s$ and a temporal transformer $H_t$) followed by a classifier $C$. The overall model is finetuned with a supervised cross-entropy loss $L^{\mathcal{CE}}$ similarly to [107]. In phase 2 (right), the weights of $H_s$ are frozen, while $H_t$ is fine-tuned. Source and target data are fed to the backbone and the proposed IB-based loss $L^{\mathcal{IB}}$ performs domain alignment, while $L^{\mathcal{CE}}$ further trains the action classifier. A queue $Q$ is added in order to increase the number of source instances considered while computing $L^{\mathcal{IB}}$.

**Phase I: Source-only fine-tuning.** The training process of this phase starts from a model $H$ pretrained on the Kinetics dataset [85] and consists of fine-tuning the entire model $F_\theta$ using only the source data $\mathcal{S}$. As in [143], we consider 16 frames uniformly sampled to represent a source video $X^S$ as input to $F_\theta$. The first part of the model $H_s$ divides each frame into 16x16 patches that are then projected into feature vectors. $H_s$ consists of ViT, which receives as input the projected patches together with a classification token $[CLS]^S$ as in [48]. Each frame is processed individually, extracting feature representations $f^S_{H_s} = [CLS]^S$ that consists of the classification token linked to that specific frame. During the forward pass, $[CLS]^S$ will collect all important information from the image patches. The frame-level features $f^S_{H_s}$ are then forwarded through $H_t$ together with a new classification token $[CLS]^T$ which, after processing, produces the video-level feature representations $f^S = [CLS]^T$. In this step, $H_s$ and $H_t$ are fine-tuned following the strategy proposed in [107], which consists of

freezing all parameters except the positional encoding, the input embeddings, the classification tokens and the affine transformations inside the layer normalisations [5]. While [107] studied the problem of partially fine-tuning a transformer for handling different modalities, in this work, we show that this strategy can be successfully applied to the problem of domain adaptation. Finally, the video level features are then fed to a linear classifier $C$. The entire model is trained with a supervised cross-entropy loss $L^{\mathcal{CE}}$, defined as:

$$L^{\mathcal{CE}} = -\mathbb{E}_{(X,y)\in\mathcal{S}} \sum y_k \log \sigma(F_\theta(X)), \tag{5.1}$$

where $\sigma$ is the softmax operation. Due to lack of space, the reader is referred to [143] and [50] for more details about the transformer architecture.

**Phase II: Target Adaptation.** In this phase, the spatial transformer $H_s$ is frozen, while the parameters of $H_t$ are trained to exploit both labelled source and unlabelled target data. Note that both $H_s$ and $H_t$ are initialised with the weights after Phase I. This choice is motivated by the need of reducing computational resources, while still performing adaptation at the temporal level. Freezing part of the model enables us to increase the batch size, which is fundamental for the proposed domain alignment strategy (Eqn. 5.3). To train our model, 16 frames are sampled from both source $X^S$ and target $X^T$ videos, as in the previous phase. Video-level feature representations $f^S$ and $f^T$ are then produced for videos of both domains. Subsequently, the temporal features of source videos $f^S$ are provided as input to the linear classifier $C$. To perform adaptation, we rely on the Information Bottleneck (IB) principle [163, 164]. Fig. 5.3 shows how the IB principle is applied to our problem. First, we assume that there exists a domain transformation $g \sim \mathcal{G}$ that maps a target instance $X^T$ to a source instance $\overline{X}^S$ that has the same label. Unlike [191], which considers that one instance is mapped to a perturbed version of the same instance via some

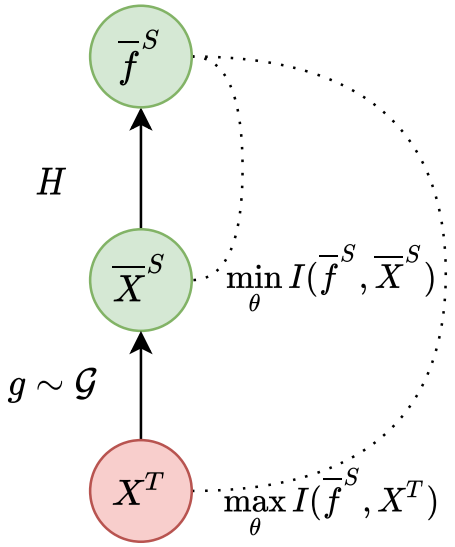



Figure 5.3: Information Bottleneck diagram showing the proposed flow of information to perform adaptation.

type of data augmentation, we map a single target instance $X^T$ to multiple different $\overline{X}^S$ in the same iteration. We experimentally show that this is indeed beneficial since by increasing the number of source instances via the usage of a queue, and consequently the number of pairs, we observed a large boost in performance. As annotations are not provided for the target domain, we resort to pseudo-labels for matching source and target instances. The model $H$ maps $\overline{X}^S$ to the feature representation $\overline{f}^S$. According to the IB principle, we want the model $H$ to learn a representation $\overline{f}^S$ which encodes as much information as possible about the original instance $X^T$. This objective is carried out by maximising the Mutual Information $I(\overline{f}^S, X^T)$. Then, the second objective consists of minimising $I(\overline{f}^S, \overline{X}^S)$ to make the model $H$ invariant to the transformation of the sample $X^T$ into a different domain. The overall loss function can be written as:

$$L^{\mathcal{IB}} = I(\overline{f}^S, \overline{X}^S) - \beta\, I(\overline{f}^S, X^T) \tag{5.2}$$

Since optimising for mutual information in a high dimensional space

is difficult, previous works have proposed different ways to approximate Eqn. 5.2. In this work, we derive a loss function similar to that used in the Barlow Twins method [191], where it was proved that, under certain conditions, Eqn. 5.2 can be approximated as:

$$L^{\mathcal{IB}} = \sum_{i}^{d}(1 - C_{ii})^2 + \lambda \sum_{i}^{d}\sum_{j \neq i}^{d}(C_{ij})^2, \tag{5.3}$$

where $\boldsymbol{C}$ is a cross-correlation matrix computed over a batch of $B$ data obtained through a feature extractor $\{z_1, \ldots, z_B\}$ and their corresponding transformed version $\{z'_1, \ldots, z'_B\}$, where $i$ is the feature index and $d$ is the total number of features. Each element of $\boldsymbol{C}$ is defined as $C_{ij} = \frac{\sum_b z_{i,b} z'_{j,b}}{\sqrt{\sum_b (z_{i,b})^2}\sqrt{\sum_b (z'_{i,b})^2}}$, where $z_i$ and $z'_i$ are mean centred. While in [191] the cross-covariance matrix is computed considering the original images and their augmented versions, in this paper we propose to re-purpose it for domain alignment using corresponding samples across the two domains. The loss in Eq.5.3 is a trade-off between two objectives, the first term that pushes the learned representation to be domain invariant and a second term that decorrelates the different components of the embedding. To build $\boldsymbol{C}$, we introduce a projection head $P$, similar to the one in [191], mapping $f^S$ and $f^T$ to $z^S$ and $z^T$. Then, each source instance representation $z_i^S$ is paired with all target instance representations $z_j^T$ where the label of instance $i$ and the pseudo-label of instance $j$ are equal. Note that the same instance $i$ or $j$ can appear in more than one pair. We also introduced a queue $Q$ to keep recent $z^S$, effectively increasing the number of possible instances that are paired with $z^T$ in the minibatch. After forming this list of pairs, the cross-correlation matrix can be computed between the source instances and the target instances of all pairs. This process makes the model invariant to instances of different domains and enables tackling the domain adaptation setting. Our final loss, introducing a weighting factor

$\alpha$, is then defined as follows:

$$\mathcal{L} = L^{\mathcal{CE}} + \alpha \mathcal{L}^{\mathcal{IB}}. \tag{5.4}$$

## 5.4 Experiments

**Datasets.** We conduct an extensive evaluation of UDAVT on two benchmarks for UDA in action recognition, namely *HMDB* $\leftrightarrow$ *UCF* [27] and *Kinetics* $\rightarrow$ *NEC-Drone* [36]. The former setting comprises videos from the *HMDB51* [93] and *UCF101* [150] action recognition datasets, which both contain real videos downloaded from Youtube. In this case, the domain shift is therefore present, but limited. *Kinetics* $\rightarrow$ *NEC-Drone*, consists of videos from the large scale *Kinetics* dataset [147], that contains sequences from Youtube, and the *NEC-Drone* [36] dataset, which consists of video sequences taken from moving drones in an indoor environment. Furthermore, the video sequences of *NEC-Drone* comprise high-resolution frames (1920x1080), and the action is often relegated to the corner of the frame and in many cases, the view is extremely slanted. Understandably, this setting is characterised by a significantly more challenging domain shift that consequently induces all the tested state-of-the-art methods to perform poorly on the original data. To alleviate this problem, we employed a pre-processing step exploiting a pretrained YOLO-based [135] human detection model using AlphaPose [55] to identify and locate the human actor(s) and then crop around the humans with a minimal resolution of 224x224. Table 5.1 reports an overview of the boost observed when applying different models to the cropped version of the dataset. As it can be seen, our pre-processing enables a consistent and significant improvement in performance for all methods, with the gain ranging from 14% to 50%. The Table also includes the performance of SAVA [178], which proposed by

Table 5.1: Effect of pre-processing on accuracy for *Kinetics* → *NEC-Drone*

| **Method** | **Original data** | **Cropped data** |
|---|---|---|
| Source only | 15.0 | 29.4 (+14.9) |
| SAVA [178] | 31.6 | 42.5 (+10.9) |
| $CO^2A$ [42] | 33.2 | 47.9 (+14.7) |
| UDAVT | 16.0 | **65.3 (+49.3)** |
| UDAVT - *supervised* | *30.0* | *78.9 (+48.9)* |

the researchers who originally released the dataset[1], and our best competitor $CO^2A$ [42]. In the following experiments, we used the cropped version of the *NEC-Drone* dataset.

**Implementation details.** Our method was implemented using Pytorch [125]. The model was fine-tuned in phase 1 starting from a STAM model pretrained on Kinetics-400. Subsequently, the partial fine-tune was carried out for 20 epochs with SGD, cosine learning rate decay, learning rate 0.001, weight decay of $1e^{-9}$ and batch size 8. In the second phase, the model was trained for an additional 20 epochs using the same optimiser, learning rate scheduler and weight decay, however, with a learning rate of 0.005 and a batch size of 64 instances per domain. Additionally, the projection head $P$ was not trained, as we found that using a fixed random non-linear projection made the model more resilient to bad pseudo-labels. Since this results in fewer parameters to optimise, we hypothesise that this makes training more consistent. For HMDB ↔ UCF, we set $\alpha = 0.01$ and the queue size to 1024. For Kinetics → NEC-Drone, we used $\alpha = 0.025$ and a queue of size 2048. First, $\alpha$ needs to be set to a small value due to the sheer difference in magnitude between $\mathcal{L}^{\mathcal{CE}}$ and $\mathcal{L}^{\mathcal{IB}}$. Second, $\alpha$ controls the strength

[1]The original code for SAVA was not available so the result for the cropped version of *Kinetics* → *NEC-Drone* was obtained with our implementation.

of decorrelating feature representations of two different instances across domains that share the same label (pseudo-label for the target domain). Intuitively, it needs to be set higher the larger the domain gap is.

**Baselines.** We compare our results with those obtained by state-of-the-art methods for UDA in video action recognition, namely TA$^3$N [27], TCoN [122], SAVA [178] and CO$^2$A [42]. For a fair comparison, the results of TA$^3$N are also reported after replacing their ResNet backbone with I3D. Also, as a transformer-based baseline, we report the results obtained by replacing our proposed UDAVT loss with three different domain alignment strategies, namely: a Maximum Mean Discrepancy (MMD) domain alignment component [104], an adversarial approach relying on a domain classifier as in [62] and a Maximum Classifier Discrepancy (MCD) based component [140]. MCD aligns domains by employing task-specific decision boundaries that maximise the discrepancy between the output of two distinct classifiers to detect target samples lying far from source support and minimise the discrepancy of the transformer, so it learns how to produce target features closer to source support. The adversarial-based approach [62] consists of adding an MLP-based domain classifier that is responsible for predicting the domains of the instances given their video-level feature representations. We added a target cross entropy based on the pseudo-labels to all baselines was we found that this improved performance.

### 5.4.1 Results

Table 5.2 presents the results on *HMDB* $\leftrightarrow$ *UCF*. Along with the scores achieved with our proposed method, we report the ones obtained by previous approaches in the same settings. As it can be observed, all transformer-based models significantly outperform previous methods (except for CO$^2$A [42]) in both directions, suggesting that transformer-based methods are

more robust to domain shift even without any domain adaptation strategy. In particular, we achieve an accuracy of 96.8% and 92.3% in the two directions, outperforming the current best competitor ($CO^2A$ [42]) by 1% and 4.5%, respectively. Results also show that our method outperforms MMD with a transformer-based architecture. Also, the proposed MCD and adversarial-based baselines are outperformed (in just one of the two directions for the case of MCD). Finally, we report, as upper bounds, the scores obtained with the supervised version of UDAVT, i.e., the case where ground truth target labels are used instead of pseudo-labels to compute the cross-correlation matrix. Table 5.3 reports the scores obtained on the *Kinetics* $\rightarrow$ *NEC-Drone* benchmark. This setting corresponds to a more significant domain shift since the target video sequences are shot by drones in a specific indoor environment. For this reason, it is easy to observe that the absolute value of all the reported scores is significantly lower when compared to the accuracy obtained in the previous benchmarks. However, the results clearly show how the transformer-based approaches strongly outperform the baselines achieving a score of 65.3%, which is about 17 points more than the best competitor. In addition, the proposed loss achieves more than 10 points when compared to the MMD-based transformer. The gap is wider when it comes to the MCD and adversarial-based baselines, which are outperformed by 27 and 25 points. These experiments show that (i) the transformer-based backbone proves effective when applied to cases where a higher domain shift is present and (ii) the proposed UDAVT alignment method addresses the domain gap more efficiently leading to a significant increase in accuracy on the target domain.

One question that might arise is: what if the pseudo-labels have poor quality? We want to show that the domains are aligned even if the pseudo-labels are initially causing clusters of different classes to form. Since UDAVT is strongly inspired by the IB principle, commonly employed

Table 5.2: Results on *HMDB* ↔ *UCF*.

| **Method** | **Encoder** | **H → U** | **U → H** |
|---|---|---|---|
| **Baselines** | | | |
| Source only [27] | ResNet | 71.7 | 73.9 |
| DANN [61] | | 76.3 | 75.2 |
| JAN [105] | | 74.7 | 79.6 |
| AdaBN [98] | | 72.2 | 77.4 |
| MCD [140] | | 73.8 | 79.3 |
| TA$^3$N [27] | | 81.8 | 78.3 |
| Target only [27] | | *82.8* | *94.9* |
| TCoN [122] | 2D/3D CNN | 89.1 | 87.2 |
| Source only [178] | I3D | 88.8 | 80.3 |
| TA$^3$N [27] | | 90.5 | 81.4 |
| SAVA [178] | | 91.2 | 82.2 |
| CO$^2$A [42] | | 95.8 | 87.8 |
| Target only [178] | | *95.0* | *96.8* |
| **Transformer-based** | | | |
| Source only | Transformer | 93.7 | 86.9 |
| MMD [104] | | 96.5 | 87.9 |
| MCD [140] | | **97.2** | 87.9 |
| Adversarial [62] | | 96.6 | 87.6 |
| UDAVT (ours) | | 96.8 | **92.3** |
| UDAVT (ours) - *supervised* | | *97.2* | *94.4* |
| Target only | | *97.9* | *95.8* |

for self-supervised learning, we also perform an additional evaluation repurposing other methods, i.e., SimCLR [28] and VICReg [9], for domain adaptation. Both methods have been proposed for unsupervised representation learning and, to adapt them for UDA, we used the same procedure employed to construct pairs in UDAVT, i.e., we replaced distorted versions of the same instances with pairs of source and target instances with corresponding label/pseudo-label. VICReg is similar to Barlow Twins but explicitly avoids the collapsing problem. It employs different terms to

Table 5.3: Results on *Kinetics* → *NEC-Drone*.

| **Method** | **Encoder** | **Top-1 Acc** |
|---|---|---|
| **Baselines** | | |
| Source only | Resnet | 15.8 |
| TA$^3$N [27] | | 28.0 |
| Source only | I3D | 32.0 |
| TA$^3$N [27] | | 44.7 |
| SAVA [178] | | 42.5 |
| CO$^2$A [42] | | 45.8 |
| **Transformer-based** | | |
| Source only | Transformer | 29.4 |
| MMD [104] | | 54.4 |
| MCD [140] | | 38.1 |
| Adversarial [62] | | 40.8 |
| UDAVT (ours) | | **65.3** |
| UDAVT (ours) - *supervised* | | *78.1* |
| *Target only* | | *82.9* |

maintain invariance to data augmentation, diversifies feature representations for different data, and enforces different features to encode different information. SimCLR relies on the concept of positive and negative pairs and, by using the InfoNCE loss [170], it performs an optimisation procedure that consists of pulling the feature representations of positives closer together whereas those of the negatives are pushed farther away. We adapted this loss for DA using a similar formulation to [87], in which multiple positives can be considered for the same instance. In the proposed model, positive instances are those from different domains that share the same label or pseudo-label and negatives are instances from a different do-

Table 5.4: Application of different self-supervised methods for domain alignment.

| **Method** | **Encoder** | **H → U** | **U → H** | **K → N-D** |
|---|---|---|---|---|
| Source only | Transformer | 93.7 | 86.9 | 29.4 |
| SimCLR [28] | | 95.4 | 85.8 | 38.6 |
| VICReg [9] | | 95.1 | 86.4 | 46.5 |
| UDAVT (ours) | | **96.8** | **92.3** | **65.3** |

main, but without matching labels. To avoid specialising within a domain, we did not consider inter-domain pairs as neither negatives nor positives. Results for these losses, compared to our UDAVT, are reported in Table 5.4. These results show that although re-purposing the considered self-supervised methods to DA produces competitive performance in the target domain, our proposed UDAVT loss outperforms both strategies. In particular, in the challenging *Kinetics* → *NEC-Drone* setting the performance boost is more than 16% higher than the second-best method.

Lastly, we report in Fig. 5.4, the ablation study that we carry out to study the performance of the model concerning (i) the usage of the queue $Q$ and (ii) the usage of target pseudo-labels for the domain alignment component. For the first, we removed $Q$, while for the latter we used random labels for target data. We can observe that the removal of $Q$ is detrimental to the overall performance of the model in the supervised and unsupervised cases, respectively, except for the supervised setting of *Kinetics* → *NEC-Drone*, where the model behaves very similarly. Regarding the labels, results show that the model always benefits from being provided with better target labels.

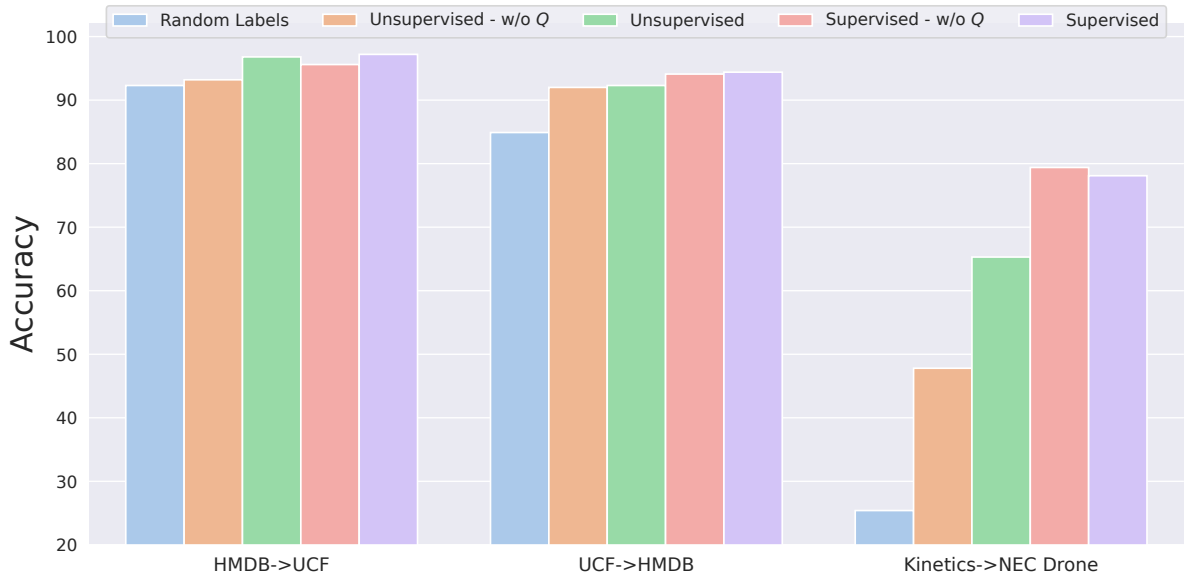



Figure 5.4: Ablation on the usage of the queue $Q$ and random labels for target data instead of pseudo-labels for domain alignment.

## 5.5 Conclusions

We presented UDAVT, a simple and novel framework for unsupervised domain adaptation in video action recognition, which couples a powerful transformer-based feature extractor with a domain alignment component that exploits the Information Bottleneck principle to perform domain alignment. We reported results on two popular benchmarks for domain adaptation in action recognition, proving the effectiveness of UDAVT by outperforming previous state-of-the-art models in all settings, and further provided insight by exploring self-supervised contrastive variants for domain alignment, all of which proved effective, yet inferior to our proposed IB loss. In future work, we plan on improving our approach by further leveraging the capabilities of visual transformers for video action recognition while devising more sophisticated domain alignment strategies accordingly. Another research direction would be adapting the model to open set domain adaptation.

# Chapter 6

# Self-supervised Models are Continual Learners

## 6.1 Introduction

During the last few years, self-supervised learning (SSL) has become the most popular paradigm for unsupervised visual representation learning [19, 21, 28, 72, 69, 191, 9, 29]. Indeed, under certain assumptions (*e.g.*, offline training with large amounts of data and resources), SSL methods are able to extract representations that match the quality of representations obtained with supervised learning, without requiring annotations. However, these assumptions do not always hold in real-world scenarios, *e.g.*, when new unlabeled data are made available progressively over time. In fact, in order to integrate new knowledge into the model, training needs to be repeated on the whole dataset, which is impractical, expensive, and sometimes even impossible when old data is not available. This issue is exacerbated by the fact that SSL models are notoriously computationally expensive to train.

Continual learning (CL) studies the ability of neural networks to learn tasks sequentially. Prior art in the field focuses on mitigating catastrophic

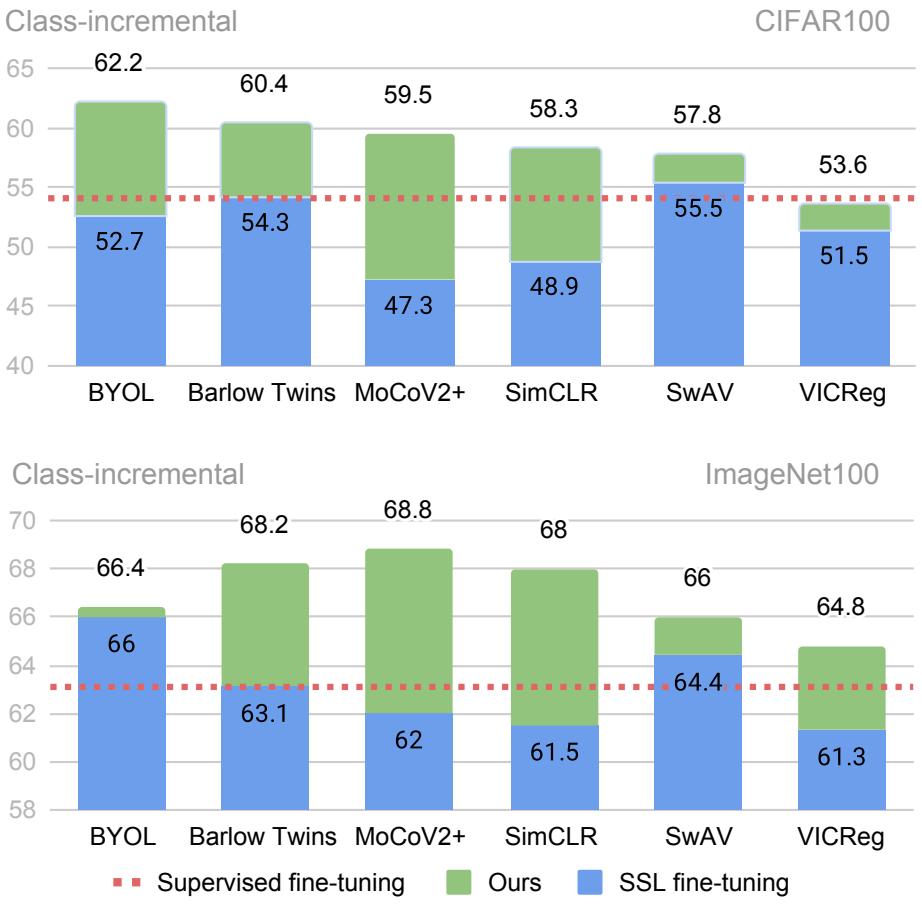



Figure 6.1: Linear evaluation accuracy of representations learned with different self-supervised methods on class-incremental CIFAR100 and ImageNet100. In blue the accuracy of SSL fine-tuning, in green the improvement brought by CaSSLe. The red dashed line is the accuracy attained by supervised fine-tuning.

forgetting [112, 59, 65, 45]. Common benchmarks in the CL literature evaluate the discriminative performance of classifiers learned *with supervision* from non-stationary distributions. In this paper, we tackle the same forgetting phenomenon in the context of SSL. Unsupervised representation learning is indeed appealing for sequential learning since it does not require human annotations, which are particularly hard to obtain when new data is generated on-the-fly. This setup, called Continual Self-Supervised Learning (CSSL), is surprisingly under-investigated in the literature.

In this work, we propose CaSSLe, a simple and effective framework for CSSL of visual representations based on the intuition that SSL models are intrinsically capable of learning continually, and that SSL losses can be seamlessly converted into distillation losses. Our key idea is to train the current model to predict past representations with a prediction head, thus encouraging it to remember past knowledge. CaSSLe has several favourable features: (i) it is compatible with popular state-of-the-art SSL loss functions and architectures, (ii) it is simple to implement, and (iii) it does not require any additional hyperparameter tuning with respect to the original SSL method. Our experiments demonstrate that SSL methods trained continually with CaSSLe significantly outperform all the related methods (CSSL baselines and several methods adapted from supervised CL).

We also perform a comprehensive analysis of the behavior of six popular SSL methods in diverse CL settings (*i.e.*, class, data, and domain incremental). We provide empirical results on small (CIFAR100), medium (ImageNet100), and large (DomainNet) scale datasets. Our study sheds new light on interesting properties of SSL methods that emerge when learning continually. Among other findings, we discover that, in the class-incremental setting, SSL methods typically approach or outperform supervised learning (see Fig.6.1), while this is not generally true for other settings (data-incremental and domain-incremental) where supervised learning still shows a sizeable advantage.

## 6.2 Related Work

**Self-Supervised Learning.** Recent SSL approaches have shown performance comparable to their supervised learning equivalents [19, 21, 28, 72,

69, 191, 9, 29]. In a nutshell, most of these methods use image augmentation techniques to generate correlated views (positives) from a sample, and then learn a model that is invariant to these augmentations by enforcing the network to output similar representations for the positives. Initially, contrastive learning, based on instance discrimination [181] using noise-contrastive estimation [70, 119], was a popular strategy [28, 72]. However, this learning paradigm requires large batch sizes or memory banks. A few methods that use a negative-free cosine similarity loss [69, 30] have addressed such issues.

Concurrently, clustering-based methods (SwAV [19], DeepCluster v2 [19, 18] and DINO [21]) have also been proposed. They do not operate on the features directly, and instead compare positives through a cross-entropy loss using cluster prototypes as a proxy. Redundancy reduction-based methods have also been popular[53, 191, 9]. Among them, BarlowTwins [191] considers an objective function measuring the cross-correlation matrix between the features, and VicReg[9] uses a mix of variance, invariance and covariance regularizations. Methods such as [52] have explored the use of nearest-neighbour retrieval and divide and conquer [160]. However, none of these works studied the ability of SSL methods to learn continually and adaptively.

**Continual Learning.** A plethora of methods have been developed to counteract catastrophic forgetting [91, 139, 144, 106, 26, 142, 25, 4, 192, 15, 58, 51, 179, 23, 134, 79, 129, 99, 121, 24, 136]. Following [45], these works can be organized into three macro-categories: replay-based [121, 136, 134, 15, 26, 106], regularization-based [58, 99, 144, 91, 192, 4, 23, 51, 79, 25, 179, 24], and parameter isolation methods [139, 142]. All these works evaluate the effectiveness of CL methods using a linear classifier learned sequentially over time. However, this evaluation does not reflect an important aspect, *i.e.*, the internal dynamics of the hidden representations. Moreover, most

CL methods tend to rely on supervision in order to mitigate catastrophic forgetting. A few of them can be adapted for the unsupervised setting, although their effectiveness is greatly reduced (see discussion in Sec. 6.5, Sec. 6.6 and the supplementary material).

Works such as [133, 1, 148] laid the foundations of unsupervised CL, but their studies are severely limited to digit-like datasets, *e.g.*, MNIST and Omniglot, and the proposed methods are unfit for large-scale scenarios. Recently, [60, 16] explored self-supervised pretraining for supervised continual learning with online and few-shot tasks, and [24] presented a supervised contrastive CL approach. Two concurrent works [101, 108] have also attempted to address CSSL recently. The former [101] extends [24] to the unsupervised setting, but is specifically designed for contrastive SSL, such as [28, 72], and lacks generalizability to other popular SSL paradigms. The latter [108] is also limited as it only shows small-scale experiments in the class-incremental setting and considers just two SSL methods. In contrast, we present a general framework for CSSL with superior performance, conduct large-scale experiments on three challenging settings, thereby presenting a deeper analysis of CSSL.

## 6.3 Preliminaries

**Self-Supervised Learning.** The training procedure of several state-of-the-art SSL methods [191, 28, 72, 19, 21, 9, 52, 69] can be summarized as follows. Given an image $\boldsymbol{x}$ in a batch sampled from a distribution $\mathcal{D}$, two correlated views $\boldsymbol{x}^A$ and $\boldsymbol{x}^B$ are extracted by applying stochastic image augmentations, such as random cropping, color jittering and horizontal flipping. View $\boldsymbol{x}^A$ is fed to an encoder $f_\theta = f_p \circ f_b$, which is parametrized by $\theta$ and has a backbone $f_b$ and a projection head $f_p$, that extracts feature representations $\boldsymbol{z}^A = f_\theta(\boldsymbol{x}^A)$. Similarly, $\boldsymbol{x}^B$ is forwarded into the same

networks, or possibly copies thereof, updated with exponential moving average (EMA), to obtain the representation $\boldsymbol{z}^B$. A loss function $\mathcal{L}_{SSL}$ is applied to these representations to learn the parameters $\theta$ as follows:

$$\underset{\theta}{\operatorname{argmin}}\ \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}\left[\mathcal{L}_{SSL}\left(\boldsymbol{z}^A, \boldsymbol{z}^B\right)\right]. \tag{6.1}$$

More details on the implementation of $\mathcal{L}_{SSL}$ are provided in Sec. 6.5.1 and Tab. 6.1. This procedure turns out to be extremely powerful at extracting visual representations from large unlabeled datasets. The intuition behind the success of these models is that they learn to be invariant to augmentations. Importantly, augmentations are hand-crafted in a way that the two views $\boldsymbol{x}^A$ and $\boldsymbol{x}^B$ contain roughly the same semantics as $\boldsymbol{x}$, but their overall appearance (geometry, colors, resolution, etc.) is different. This forces the model map images with the same semantics to similar regions of the feature space. Interestingly, these augmentations are much stronger, *i.e.*, they distort the image more, than augmentations commonly used to train supervised models.

**Continual Learning.** The CL problem focuses on training models such as deep neural networks from non-stationary data distributions. More formally, this involves a network $f'_{\theta'} = f'_c \circ f'_b$ with parameters $\theta'$, backbone $f'_b$ and a classifier $f'_c$, that learns from an ordered set of tasks $\{1, \ldots, T\}$, each exhibiting a different data distribution $\mathcal{D}_t$. Usually, an image $\boldsymbol{x}$ sampled i.i.d. from $\mathcal{D}_t$ is processed by $f'$ that predicts a probability distribution $\boldsymbol{p}$ over the set of classes $\mathcal{Y}_t$. The objective is to find parameters $\theta'$ such as:

$$\underset{\theta'}{\operatorname{argmin}} \sum_{t=1}^{T} \mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})\sim\mathcal{D}_t}\left[\mathcal{L}_{CL}\left(\boldsymbol{p}, \boldsymbol{y}\right)\right], \tag{6.2}$$

where, in most cases, $\mathcal{L}_{CL}$ is the cross-entropy loss. However, during task $t$, the previous data distribution $\mathcal{D}_{t-1}$ is not available and therefore Eq. (6.2) cannot be minimized directly. Current research focuses on approximating

$\theta'$ using indirect approaches. Some of them [99, 51] are based on knowledge distillation [75], *i.e.*, transferring knowledge from one network to another by forcing them to produce the same outputs. We will discuss the applicability of distillation methods in CSSL in Sec. 6.5.

## 6.4 Continual Self-Supervised Learning

In this paper, we tackle the problem of Continual Self-Supervised Learning as an extension of both SSL and CL. In practice, a CSSL experiment starts with the first task, where the model is trained as per the specific self-supervised method that it implements, with no difference from offline training. Subsequent tasks are then presented to the model sequentially, and the data from the previous tasks are discarded. No labels are provided during this training phase. For the sake of simplicity and since we are exploring a new, challenging setting, we assume task boundaries to be provided to the model. More formally, the CSSL objective is to learn a strong feature extractor that is invariant to augmentations on all tasks. Following the notation introduced in Sec. 6.3, we define:

$$\underset{\theta}{\operatorname{argmin}} \sum_{t=1}^{T} \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}_t} \left[ \mathcal{L}_{SSL} \left( \boldsymbol{z}^A, \boldsymbol{z}^B \right) \right]. \tag{6.3}$$

Note the absence of labels $\boldsymbol{y}$ when sampling from $\mathcal{D}_t$, the summation over the set of tasks inherited from Eq. (6.2) and the SSL loss function in Eq. (6.1). The expectation is approximated using stochastic gradient descent on minibatches.

**Evaluation.** After each task, it is possible (for evaluation purposes) to train a linear classifier on top of the obtained backbone $f_b$. With this linear classifier we report accuracy on the test set. This protocol is compatible with standard CL metrics, as shown in Sec. 6.6.1. We explore three CSSL

settings in our work.

▸ **Class-incremental**: each task $t$ is represented by a dataset $D_t \sim \mathcal{D}_t$ containing images that belong to a set of classes $\mathcal{Y}_t$ such that $\mathcal{Y}_t \cap \mathcal{Y}_s = \varnothing$ for each other task $s \neq t$. Note that the class labels are only used for splitting the dataset and they are unknown to the model. In practice, the set of classes in the dataset are shuffled and then partitioned into $T$ tasks. Each task contains the same number of classes.

▸ **Data-incremental**: each task $t$ contains a set of images $D_t$ such that $D_t \cap D_s = \varnothing$ for each other task $s \neq t$. No additional constraints are imposed on the classes. In practice, the whole dataset is shuffled and then partitioned into $T$ tasks. Each task can potentially contain all the classes.

▸ **Domain-incremental**: each task $t$ contains a set of images $D_t$ drawn from a different domain. We assume that the set of classes $\mathcal{Y}_t$ in each dataset remains the same for all tasks but the data distribution changes, as if the data were collected from different sources.

## 6.5 The CaSSLe Framework

We now introduce "CaSSLe", our framework for continual self-supervised learning of visual representations and detail its compatibility with several SSL methods.

**Distillation in CSSL.** From a supervised CL perspective, the concept of invariance is interesting. Here, we would like to learn representations of previously-learned semantic concepts that are invariant to the state of the model's parameters. Indeed, this idea was investigated in prior works [51, 79] that leverage knowledge distillation for CL. However, such approaches are only mildly effective in a CSSL scenario, as we show in Sec. 6.6. We believe this is due to CSSL being fundamentally different from supervised CL. In CSSL, we aim to extract the best possible representations

that can be subsequently reused in a variety of tasks, and maximize the linear separability of features at the end of the CL phase. Thus, the linear classifier does not benefit much from the stability of the representations. Also, forcing the representations not to change may prevent the model from learning new concepts. This is especially critical for SSL methods for two reasons: (i) the performance of the models improve substantially with longer training, implying that the representations continue to get refined, and (ii) they exhibit different losses and feature normalizations that might interfere with distillation and vice-versa (*e.g.*, BarlowTwins uses standardization while [51, 79] use $l2$-normalization). Nonetheless, the features still need to be informative of previous tasks to maximize the separability of the old distribution but the current state might be too different from the previous one making comparing representations complicated.

**Distillation through SSL losses.** Our framework, shown in Fig. 6.2, is based on the following ideas: (i) a predictor network that maps the current state of the representations to their past state, by leveraging a distillation through time strategy that satisfies both stability and plasticity principles, and (ii) a family of adaptable distillation losses inherited from the SSL literature that solves the issue of having different objectives interfering with each other.

When a new task is received, we start by making a copy of the current model. This copy does not require gradient computation and will not be updated. We call this the *frozen encoder* $f^{t-1}$. As soon as an image $\boldsymbol{x} \in D_t$ is available we apply our stochastic image augmentations and extract its features $\boldsymbol{z} = f^t(\boldsymbol{x})$. In addition, we also use the frozen encoder to extract another feature vector $\bar{\boldsymbol{z}} = f^{t-1}(\boldsymbol{x})$. Now, our goal is to ensure that $\boldsymbol{z}$ contains at least as much information as (and ideally more than) $\bar{\boldsymbol{z}}$. Instead of enforcing the two feature vectors to be similar, and hence discouraging the new model from learning new concepts, we propose to use a predictor

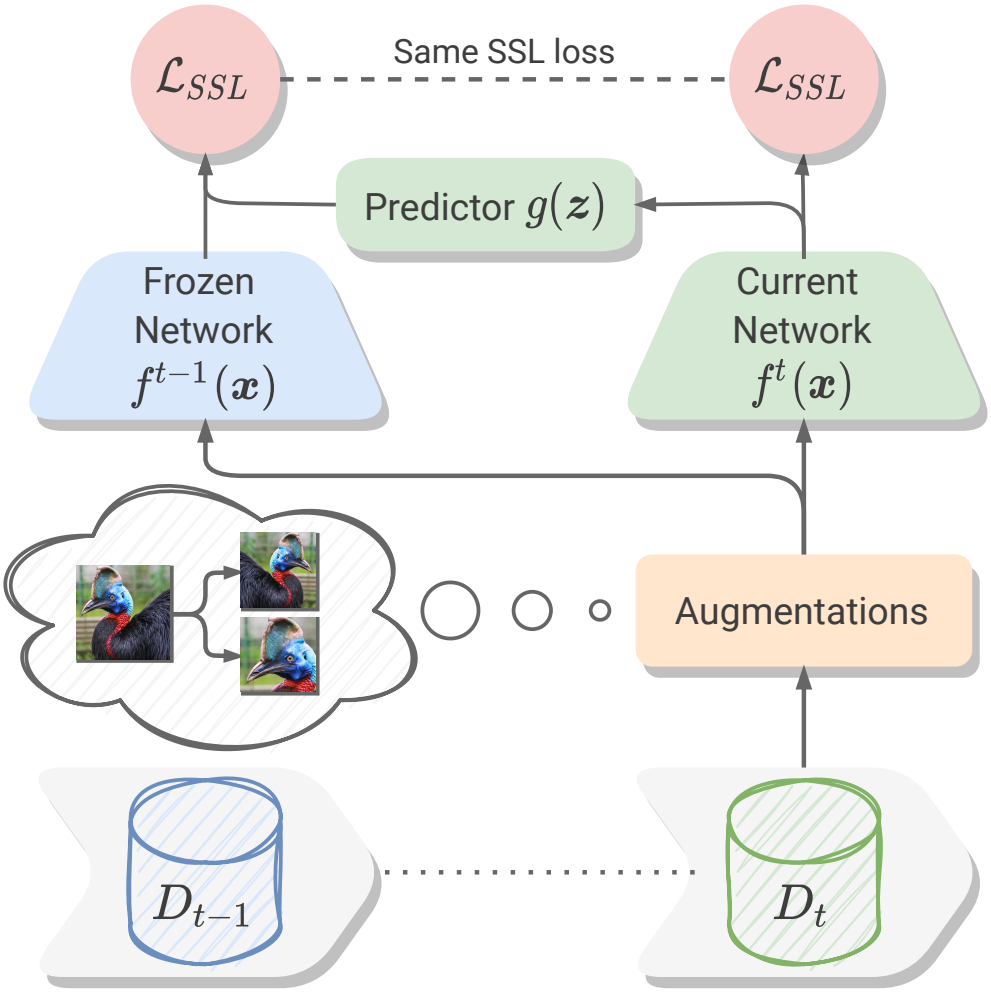



Figure 6.2: Overview of the CaSSLe framework.

network $g$ to project the representations from the new feature space to the old one. If the predictor is able to perfectly map from one space to the other, then it implies that $\boldsymbol{z}$ is at least as powerful as $\bar{\boldsymbol{z}}$.

We are now ready to perform distillation, but which is the most appropriate distillation loss? Since we want the representations produced by $g$ to be invariant to the state of the model, we propose to use the same SSL loss used to simulate invariance to augmentations. Empirically, we verify that this choice reduces interference and minimizes the need for hyperparameter tuning. We can hence write a generic distillation loss by reusing

the definition of $\mathcal{L}_{SSL}$:

$$\mathcal{L}_D(\boldsymbol{z}, \bar{\boldsymbol{z}}) = \mathcal{L}_{SSL}(g(\boldsymbol{z}), \bar{\boldsymbol{z}}). \tag{6.4}$$

Note that $\bar{\boldsymbol{z}}$ is always detached from the computational graph, such that the frozen encoder does not receive any gradient, and the gradient only flows through the predictor $g$, as prescribed in [30]. On the one hand, if training converges and $\mathcal{L}_D$ is minimized, the features predicted by $g$ will likely be quasi-invariant to the state of the model, which satisfies the stability principle. On the other hand, the current encoder is less bound to its previous state, hence representations $\boldsymbol{z}$ can be more plastic. The loss can be extended to multiple views by applying it to both representations, *i.e.*, $\mathcal{L}_D(\boldsymbol{z}^A, \bar{\boldsymbol{z}}^A) + \mathcal{L}_D(\boldsymbol{z}^B, \bar{\boldsymbol{z}}^B)$, and also swapped distillation, *e.g.*, $\mathcal{L}_D(\boldsymbol{z}^A, \bar{\boldsymbol{z}}^B)$ and vice-versa (see ablation in Tab. 6.6).

The final loss of an SSL method trained continually with the CaSSLe framework is given by:

$$\begin{aligned} \mathcal{L} &= \mathcal{L}_{SSL}(\boldsymbol{z}^A, \boldsymbol{z}^B) + \mathcal{L}_D(\boldsymbol{z}^A, \bar{\boldsymbol{z}}^A) \\ &= \mathcal{L}_{SSL}(\boldsymbol{z}^A, \boldsymbol{z}^B) + \mathcal{L}_{SSL}(g(\boldsymbol{z}^A), \bar{\boldsymbol{z}}^A). \end{aligned} \tag{6.5}$$

This loss can be made symmetric by applying it to both the views (swapping $A$ and $B$ in Eq. (6.5)) and it can also be easily adapted for multi-crop [19]. Note that we do not use any hyperparameter to weight the importance of the distillation loss with respect to the SSL loss.

### 6.5.1 Compatibility of SSL methods with CaSSLe

The main difference among SSL methods is the loss function that they use. Following the notation defined in Sec. 6.3, and the loss functions in Tab. 6.1, we now detail if and how SSL losses can be used in our CaSSLe framework. Full derivation of distillation losses is deferred to the supplementary material.

**InfoNCE-based** methods [28, 72] perform instance discrimination, where positive samples help to build invariance to augmentations. The negatives prevent the model from falling into degenerate solutions. The InfoNCE (*a.k.a.* contrastive) loss can be written as in Eq. (6.6), where subscript $i$ is the index of a generic sample in the batch, sim is the cosine similarity and $\eta(i)$ is the set of negatives for sample $i$ in the current batch. Distilling knowledge with this loss is equivalent to performing instance discrimination of current task samples but in the feature space learnt in the past. Thus, the predictor $g$ learns to project samples from the present to the past space to maximize the distance with the negative samples, and the similarity with itself in the past.

**MSE-based** approaches [69, 30] enforce consistency among positive samples and ignore the negatives. BYOL [69] uses a momentum encoder and SimSiam [30] performs a stop gradient operation to avoid degenerate solutions. Since the representations are $l2$-normalized, their loss (Eq. 6.7) can be rewritten as the negative cosine similarity: $-\text{sim}(\boldsymbol{q}^A, \boldsymbol{z}^B) = -\frac{\boldsymbol{q}^A}{||\boldsymbol{q}^A||_2} \cdot \frac{\boldsymbol{z}^B}{||\boldsymbol{z}^B||_2}$, where $\boldsymbol{q}^A = h(\boldsymbol{z}^A)$ and $h$ is a prediction head. The gradient is back-propagated only through the representations of the first argumentation. A special case of this family of methods is VICReg [9], which uses a combination of multiple losses, where MSE acts as invariance term. Features are not $l2$-normalized in VICReg and its predictor is the identity function. In our framework, this loss encourages the model to predict the past state of representations without additional regularization.

**Cross-entropy-based.** Instead of simply enforcing invariance of the representations to augmentations, cluster prototypes $\mathbf{C} = \{\mathbf{c}_1, \ldots, \mathbf{c}_K\}$ are used as a proxy in these approaches, so that the model learns to predict invariant cluster assignments. Slight variations of this idea result in different methods: SwAV [19], DeepClusterV2 [19] and DINO [21]. Once a probability distribution over the prototypes is predicted, the cross-entropy

Table 6.1: Overview of state-of-the-art SSL methods and losses. In all tables, highlight colors are coded according to the type of loss.

| Methods | Loss | Equation |
|---|---|---|
| SimCLR [28<br>MoCo [72]<br>NNCLR [52 | InfoNCE | $-\log \frac{\exp\left(\text{sim}\left(\boldsymbol{z}_i^A, \boldsymbol{z}_i^B\right)/\tau\right)}{\sum_{\boldsymbol{z}_j \in \eta(i)} \exp\left(\text{sim}\left(\boldsymbol{z}_i^A, \boldsymbol{z}_j\right)/\tau\right)}$ (6.6) |
| BYOL [69]<br>SimSiam [30<br>VICReg [9] | MSE | $-\|\|\boldsymbol{q}^A - \boldsymbol{z}^B\|\|_2^2$ (6.7) |
| SwAV [19]<br>DCV2 [19]<br>DINO [21] | Cross-entropy | $-\sum_d \boldsymbol{a}_d^B \log \frac{\exp\left(\text{sim}\left(\boldsymbol{z}^A, \boldsymbol{c}_d\right)/\tau\right)}{\sum_k \exp(\text{sim}(\boldsymbol{z}^A, \boldsymbol{c}_k)/\tau)}$ (6.8) |
| Barlow<br>Twins [191]<br>VICReg [9] | Cross-correlation | $\sum_u \left(1 - \mathcal{C}_{uv}\right)^2 + \lambda \sum_u \sum_{v \neq u} \mathcal{C}_{uv}^2$ (6.9) |

loss (Eq. 6.8) is used to compare the two views. Features and cluster prototypes $\boldsymbol{c}$ are $l2$-normalized. The assignments $\boldsymbol{a}^B$ can be calculated in several ways, *e.g.*, k-means in DeepCluster, Sinkhorn-Knopp in SwAV and EMA in DINO. When employed as a distillation loss, cross-entropy encourages $g$ to predict the assignments generated by the frozen encoder with a set of frozen prototypes: $\boldsymbol{a}^B = \frac{\exp\left(\text{sim}\left(\bar{\boldsymbol{z}}^B, \boldsymbol{c}_d^{t-1}\right)/\tau\right)}{\sum_k \exp\left(\text{sim}\left(\bar{\boldsymbol{z}}^B, \boldsymbol{c}_k^{t-1}\right)/\tau\right)}$, where $\mathbf{C}^{t-1} = \left\{\mathbf{c}_1^{t-1}, \ldots, \mathbf{c}_K^{t-1}\right\}$.

**Cross-correlation-based**. These methods use a different approach based on decorrelating the components of the feature space, *e.g.*, Barlow Twins [191], VICReg [9] and W-MSE [53]. For our analysis, we will mainly focus on Barlow Twins' implementation of this objective. Extensions to VICReg are left for future work. The cross-correlation based objective function is shown in Eq. 6.9, where $\lambda$ is an hyperparameter to control the importance of the

first and the second terms of the loss, and $\mathcal{C}_{uv} = \frac{\sum_i \boldsymbol{z}^A_{i,u} \boldsymbol{z}^B_{i,v}}{\sqrt{\sum_i \left(\boldsymbol{z}^A_{i,u}\right)^2} \cdot \sqrt{\sum_i \left(\boldsymbol{z}^B_{i,v}\right)^2}}$ is the value of position $(u, v)$ of the cross-correlation matrix computed between the representations of the views along the batch dimension. Note that the representations here are mean centered along the batch dimension, such that each unit has mean output zero over the batch. Performing distillation with this loss has the additional effect of decorrelating the dimensions of the predicted features $g(\boldsymbol{z}^A)$.

## 6.6 Experiments

### 6.6.1 Experimental Protocol

**Evaluation Metrics.** Following previous work [106], we propose the following metrics to evaluate the quality of the representations extracted by our CSSL model:

▸ **Linear Evaluation Accuracy**: accuracy of a classifier trained on top of the backbone $f_b$ on all tasks (or a subset, *e.g.*, 10% of the data) or a downstream task. For class-incremental and data-incremental, we use the task-agnostic setting, meaning that at evaluation time we do not assume to know the task ID. For the domain-incremental setting, we perform both task-aware and task-agnostic evaluations (the latter is discussed in the supplementary material). To calculate the average accuracy we compute $A = \frac{1}{T} \sum_{i=1}^{T} A_{T,i}$, where $A_{j,k}$ is the linear evaluation accuracy of the model on task $k$ after observing the last sample from task $j$.

▸ **Forgetting**: a common metric in the CL literature, it quantifies how much information the model has forgotten about previous tasks. It is formally defined as: $F = \frac{1}{T-1} \sum_{i=1}^{T-1} \max_{t \in \{1,\dots,T\}} (A_{t,i} - A_{T,i})$.▸ **Forward Transfer**: measures how much the representations that we learned so far are helpful in learning new tasks, namely:

$FT = \frac{1}{T-1}\sum_{i=2}^{T} A_{i-1,i} - R_i$ where $R_i$ is the linear evaluation accuracy of a random network on task $i$.

**Datasets.** We perform experiments on 3 datasets: CIFAR100 [92] (class-incremental), a 100-class dataset with 60k 32x32 colour images; ImageNet100 [161] (class- and data-incremental), 100-class subset of the ILSVRC2012 dataset with ≈130k images in high resolution (resized to 224x224); DomainNet [127] (domain-incremental), a 345-class dataset containing roughly 600k high-resolution images (resized to 224x224) divided into 6 domains. We experiment with 5 tasks for the class- and data-incremental settings and with 6 tasks (one for each domain in DomainNet) in the case of domain-incremental. The supplementary material presents additional results with different numbers of tasks. For the domain-incremental setting, we order the domains in decreasing number of images.

**Implementation details.** The SSL methods are adapted from `solo-learn` [40], an established SSL library, which is the main code base for all our experiments. The number of epochs per task is as follows: 500 for CIFAR100, 400 for ImageNet100, 200 for DomainNet. The backbone $f_b$ is a ResNet18 [74], with batch size 256. We use LARS [189] for all our experiments. The offline version of each method, which serves as an upper bound, is trained for the same number of epochs as the continual counterpart for a fair comparison. All the results for offline upper bounds are obtained using the checkpoints provided in [40]. For some SSL methods, it was necessary to slightly increase the learning rate over the values provided by [40] in order for the methods to fully converge in the CSSL setting. Although tuning the hyperparameters might be beneficial in some settings, we do **not** perform any hyperparameter tuning for CaSSLe. We also neither change the parameters of the SSL methods, nor use a weight for the distillation loss (as per Eq. (6.5)).

**Baselines.** Most of the CL methods require labels which makes them unsuitable for CSSL. However, a few works can be adapted for our setting with minimal changes. We choose baselines from three categories [45]: prior-focused regularization (EWC [91]), data-focused regularization (POD [51], Less-Forget [79]), and rehearsal-based replay (ER [136], DER [15]) methods. We also compare with two concurrent works that propose approaches for CSSL (LUMP [108], Lin et al. [101]). Finally, we do not consider methods based on VAEs [133, 1], since they have been shown to yield poor performance on large scale. Details on how the baselines are selected, implemented and tuned for CSSL can be found in the supplementary material.

Table 6.2: Comparison with state-of-the-art CL methods on CIFAR100 (5 tasks, class-incremental) using linear evaluation top-1 accuracy, forgetting and forward transfer.

| **Strategy** | SimCLR | | | Barlow Twins | | | BYOL | | |
|---|---|---|---|---|---|---|---|---|---|
| | **A** (↑) | **F** (↓) | **T** (↑) | **A** (↑) | **F** (↓) | **T** (↑) | **A** (↑) | **F** (↓) | **T** (↑) |
| Fine-tuning | 48.9 | 1.0 | 33.5 | 54.3 | 0.4 | 39.2 | 52.7 | 0.1 | 35.9 |
| EWC [91] | 53.6 | **0.0** | 33.3 | 56.7 | **0.2** | 39.1 | 56.4 | **0.0** | 39.9 |
| ER [136] | 50.3 | 0.1 | 32.7 | 54.6 | 3.0 | 39.4 | 54.7 | 0.4 | 36.3 |
| DER [15] | 50.7 | 0.4 | 33.2 | 55.3 | 2.5 | 39.6 | 54.8 | 1.1 | 36.7 |
| LUMP [108] | 52.3 | 0.3 | 34.5 | 57.8 | 0.3 | 41.0 | 56.4 | 0.2 | 37.9 |
| Less-Forget[79] | 52.5 | 0.2 | 33.8 | 56.4 | **0.2** | 40.1 | 58.6 | 0.2 | 41.1 |
| POD[51] | 51.3 | 0.1 | 33.8 | 55.9 | 0.3 | 40.3 | 57.9 | **0.0** | 41.1 |
| CaSSLe | **58.3** | 0.2 | **36.4** | **60.4** | 0.4 | **42.2** | **62.2** | **0.0** | **43.6** |
| Offline | 65.8 | - | - | 70.9 | - | - | 70.5 | - | - |

### 6.6.2 Results

**Comparison with the state of the art.** In Tab. 6.2 we report comparison with CL baselines and fine-tuning in composition with three SSL methods: SimCLR, Barlow Twins and BYOL. We select these three meth-

Table 6.3: Comparison with Lin et al. [101] on CIFAR100 (2 and 5 tasks, class-incremental setting). MoCoV2+ is an updated version of MoCoV2 that uses a symmetric loss. The difference between the two is ≈1% at convergence [30].

| Strategy | Method | 2 Tasks | 5 Tasks |
|---|---|---|---|
| Lin et al.[101] | SimCLR | 55.7 | - |
| | MoCoV2 | 56.1 | 53.8 |
| CaSSLe | SimCLR | **61.8** | **58.3** |
| | MoCoV2+ | **63.3** | **59.5** |

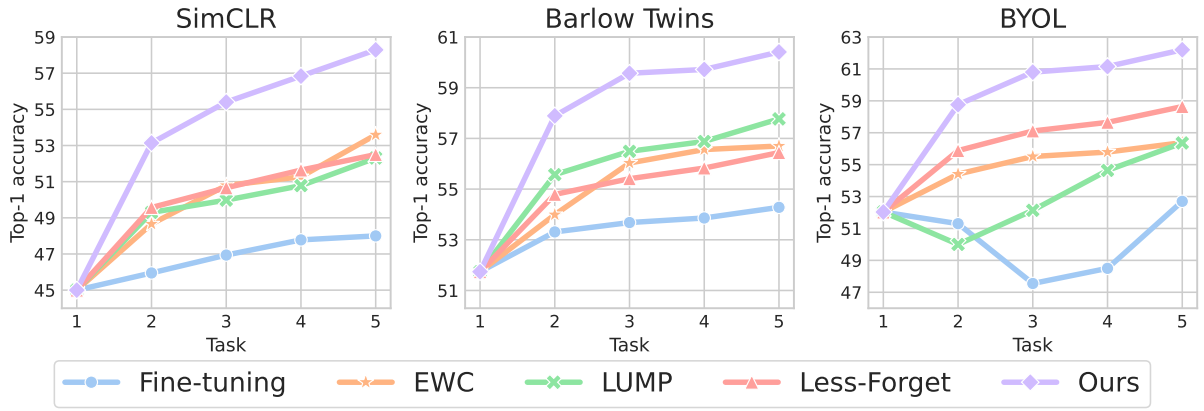



Figure 6.3: Evolution of top-1 linear evaluation accuracy over tasks on CIFAR100 (5 tasks, class-incremental).

ods for the following reasons: (i) they feature different losses (InfoNCE, Cross-correlation and MSE), (ii) they exhibit different feature normalizations ($l2$, standardization and mean centering), and (iii) they use different techniques to avoid collapse (negatives, redundancy reduction, momentum encoder). The comparison is performed on class-incremental CIFAR100 with 5 tasks. Offline learning results are reported as upper bound.

First, we notice that CaSSLe produces better representations than all the other strategies, outperforming them by large margins with all SSL methods in terms of top-1 accuracy. Moreover, our framework also shows better forward transfer, meaning that its features are easier to generalize

to other tasks (also evident in Tab. 6.8). CaSSLe appears to reduce catastrophic forgetting with respect to fine-tuning, and is comparable to other methods. In general, SSL methods already have low forgetting with respect to supervised learning on CIFAR100 (see Tab. 6.4) and therefore there is little margin for improvement. However, on higher resolution images (ImageNet100) CaSSLe actually achieves remarkable results in the mitigation of catastrophic forgetting.

Replay-based methods (ER, DER) clearly do not help against forgetting in CSSL. We found two reasons for this failure. First, in supervised CL, replay-based methods benefit from storing labels, which contain a lot of information about previous tasks and enable the retraining of the linear classifier on old classes. This is not the case in CSSL, where labels are unavailable. Second, SSL models need more training epochs to converge, which means that samples in the buffer are also replayed many more times. This causes severe overfitting on these exemplars, defeating the purpose of the replay buffer. LUMP mitigates this effect by augmenting the buffer using mixup but does not reach too far, surpassing other baselines only with Barlow Twins. EWC holds up surprisingly well, outperforming more recent methods, meaning that the importance of the weights can be calculated accurately with the self-supervised loss. Distillation methods (POD, LessForget) show good performance. However, they use $l2$-normalization in their loss, causing loss of information when coupled with Barlow Twins, which decreases accuracy.

Fig. 6.3 shows the evolution of top-1 linear evaluation accuracy over the whole training trajectory on class-incremental CIFAR100 with 5 tasks. CaSSLe outperforms the other methods, and keeps improving throughout the sequence. We found BYOL to be unstable when simply fine-tuning the model. CaSSLe, EWC and Less-Forget mitigate this instability completely. On the other hand, LUMP first drops slightly and then recovers. We believe

Table 6.4: Linear evaluation top-1 accuracy on class-incremental CIFAR100 and ImageNet100 with 5 tasks. CaSSLe is compared to fine-tuning, offline and supervised learning.

| **Method** | **Strategy** | **CIFAR100** | | | **ImageNet100** | | |
|---|---|---|---|---|---|---|---|
| | | **A** (↑) | **F** (↓) | **T** (↑) | **A** (↑) | **F** (↓) | **T** (↑) |
| Barlow Twins | Fine-tuning | 54.3 | **0.4** | 39.2 | 63.1 | 10.7 | 44.4 |
| | CaSSLe | **60.4** | **0.4** | **42.2** | **68.2** | **1.3** | **47.9** |
| | Offline | 70.9 | - | - | 80.4 | - | - |
| SwAV | Fine-tuning | 55.5 | **0.0** | 32.8 | 64.4 | 4.3 | 42.8 |
| | CaSSLe | **57.8** | **0.0** | **34.5** | **66.0** | **0.2** | **43.6** |
| | Offline | 64.9 | - | - | 74.3 | - | - |
| BYOL | Fine-tuning | 52.7 | 0.1 | 35.9 | 66.0 | 2.9 | 43.2 |
| | CaSSLe | **62.2** | **0.0** | **42.2** | **66.4** | **1.1** | **46.6** |
| | Offline | 70.5 | - | - | 80.3 | - | - |
| VICReg | Fine-tuning | 51.5 | 0.9 | 36.4 | 61.3 | 7.9 | 42.0 |
| | CaSSLe | **53.6** | **0.2** | **41.1** | **64.8** | **4.3** | **45.3** |
| | Offline | 68.5 | - | - | 79.4 | - | - |
| MoCoV2+ | Fine-tuning | 47.3 | 0.2 | 33.4 | 62.0 | 8.4 | 41.6 |
| | CaSSLe | **59.5** | **0.0** | **39.6** | **68.8** | **1.5** | **46.8** |
| | Offline | 69.9 | - | - | 79.3 | - | - |
| SimCLR | Fine-tuning | 48.9 | 1.0 | 33.5 | 61.5 | 8.1 | 40.3 |
| | CaSSLe | **58.3** | **0.2** | **36.4** | **68.0** | **2.2** | **45.8** |
| | Offline | 65.8 | - | - | 77.5 | - | - |
| Supervised | Fine-tuning | 54.1 | 6.8 | 36.5 | 63.1 | 5.6 | 42.5 |
| | Offline | 75.6 | - | - | 81.9 | - | - |

this is due to some instability introduced by the mixup regularization, to which the model takes time to adapt.

In Tab. 6.3 we also compare with Lin et al. [101] on class-incremental CIFAR100. Although our method is not specifically designed for contrastive learning, it substantially outperforms Lin et al. with 2 and 5 tasks. It is worth nothing that MoCoV2+ is slightly better than MoCoV2 ($\approx$1% difference), whereas our gains are much larger ($\approx$7%).

**Ablation study.** We ablate the most critical design choices we adopt in CaSSLe: (i) distillation without swapped views, and (ii) the presence of a prediction head $g$. These results are reported in Tab. 6.6. Our full framework clearly outperforms its variants with swapped views and without predictor. This validates our hypothesis that a predictor to map new features to the old feature space is crucial. The result that swapping views does not help is likely due to the frozen encoder not being invariant to the current task.

**Class-incremental.** In Tab. 6.4 we report a study of CSSL with 6 SSL methods in composition with the CaSSLe framework on class-incremental CIFAR100 and ImageNet100. Fine-tuning and Offline SSL results are reported as lower and upper bounds. The accuracy of supervised learning is also reported. CaSSLe always improves with respect to fine-tuning. In particular, our framework produces higher forward transfer and lower forgetting, especially on ImageNet100, where methods tend to forget more. Notably, CaSSLe outperforms supervised fine-tuning, except when coupled with VICReg on CIFAR100. On average, SSL methods trained continually with CaSSLe improve by 6.8% on CIFAR100 and 4% on ImageNet100.

**Data-incremental.** Tab. 6.7 presents results for linear evaluation top-1 accuracy on ImageNet100 with 5 tasks in a data-incremental scenario. While no SSL method is better than supervised fine-tuning, Barlow Twins

Table 6.5: Training 5 times longer on 1/5 of the data vs. training continually w/ and w/o CaSSLe on ImageNet100 (5 tasks, class- and data-incremental). **Bold** is best, underlined is second best.

| Setting | Method | Fine-tune | Offline 1/5 | CaSSLe |
|---|---|---|---|---|
| Class-inc. | SimCLR | 61.5 | 63.1 | **68.0** |
| | Barlow Twins | 63.1 | 63.5 | **68.2** |
| | BYOL | 66.0 | 60.6 | **66.4** |
| Data-inc. | SimCLR | 68.9 | 67.2 | **72.1** |
| | Barlow Twins | 71.3 | 70.2 | **74.9** |
| | BYOL | **74.0** | 66.7 | 73.3 |

Table 6.6: Ablation study of design choices in CaSSLe.

| Strategy | Method | Swap | No pred. | Ours |
|---|---|---|---|---|
| CaSSLe | SimCLR | 49.3 | 52.6 | **58.3** |
| | Barlow Twins | 57.4 | 57.3 | **60.4** |
| | BYOL | 52.0 | 58.6 | **62.2** |

coupled with CaSSLe is competitive. CaSSLe improves performance in all cases by 2% on average, except for BYOL. This is likely due to the fact that in the data-incremental scenario remembering past knowledge is less important than in other scenarios, and BYOL already has a momentum encoder that provides some information about the past. This hypothesis is validated by the fact that MoCoV2+ (that uses a momentum encoder) improves less than SimCLR when coupled with CaSSLe. We believe that, by tuning the EMA schedule, improvement could also be achieved for BYOL. In addition, BYOL already shows impressive performance with fine-tuning, outperforming all the other methods by more than 2%. Interestingly, SwAV comes closest to its offline upper bound, with only a 3% decrease in performance when coupled with CaSSLe.

**Domain-incremental.** We also examine the capability of CaSSLe to learn continually when the domain from which the data is drawn changes. Tab. 6.7 shows the average top-1 accuracy of a linear classifier trained on top of the frozen feature extractor on all domains separately (domain-aware). Domain-agnostic evaluation and results for each domain are presented in the supplementary material. Again, CaSSLe improves every method by 4.4% on average, showing that our distillation strategy is robust to domain shift, and although the data distribution is really different, information transfer is still performed. Interestingly, most of the methods, when trained with CaSSLe get very close to their offline accuracy.

**Long training vs continual training.** We also analyze the following question: is it worth training continually or is it better to train for longer on a small dataset? This depends on two factors: (i) the SSL method, and (ii) the CSSL setting. For SimCLR and Barlow Twins in the class-incremental setting it seems to be better to train offline on 1/5 of the classes instead of training continually with 5 tasks. In this setting, offline BYOL seems to suffer from instability, ending up lower than fine-tuning. On the other hand, on the data-incremental setting, fine-tuning outperforms longer training, especially for BYOL, which also outperforms CaSSLe (as explained previously). Apart from this exception, CaSSLe always produces better representations than other strategies, making it the go-to option.

**Downstream and semi-supervised.** In Tab. 6.8, we present the downstream performance of CaSSLe compared with fine-tuning when trained on ImageNet100 and evaluated on DomainNet (Real). Barlow Twins, SwAV and BYOL show higher performance than the supervised model, even when considering a fine-tuning strategy. This is probably due to the fact that SSL methods tend to learn more general features than their supervised counterparts. CaSSLe improves performance on all the SSL methods, making them surpass the supervised baseline. Lastly, when compared with fine-tuning,

CaSSLe improves the performance of SSL methods by 3.4% on average. Tab. 6.9 contains the top-1 accuracy on ImageNet100 when training a linear classifier on a frozen backbone with limited amount of labels (10% and 1%). First, we can observe that no SSL method with fine-tune surpasses the performance of supervised learning. When using CaSSLe, MoCoV2+ outperforms supervised with 10% labels and, in general, Barlow Twins and MoCoV2+ work best in both semi-supervised settings. CaSSLe improves all SSL methods when compared with fine-tuning.

## 6.7 Conclusion

In this work, we study Continual Self-Supervised Learning (CSSL), the problem of learning a set of tasks without labels continually. We make two important contributions for the SSL and CL communities: (i) we present CaSSLe, a simple and effective framework for CSSL that shows how SSL methods and losses can be seamlessly reused to learn continually, and (ii) we perform a comprehensive analysis of CSSL, leading to the emergence of interesting properties of SSL methods.

**Limitations.** Although CaSSLe shows exciting performance, it has some limitations. First, it is applicable in settings where task boundaries are provided. Second, our framework increases the amount of computational resources needed for training by roughly 30%, both in terms of memory and time. Finally, CaSSLe does not perform clustering, meaning that it is unable to directly learn a mapping from data to latent classes, and thus needs either a linear classifier trained with supervision, or some clustering algorithm.

**Broader impact.** The capabilities of supervised CL agents are bounded by the need for human-produced annotations. CSSL models can poten-

tially improve without the need for human supervision. This facilitates the creation of powerful AIs that may be used for malicious purposes such as discrimination and surveillance. Also, since in CSSL the data is supposed to come from a non-curated stream, the model may be affected by biases in the data. This is problematic because biases are then be transferred to downstream tasks.

Table 6.7: Linear evaluation accuracy on ImageNet100 (5 tasks, data-incremental) and DomainNet (6 tasks, domain-incremental).

| Method | Strategy | ImageNet100 (Data-inc.) | DomainNet (Domain-inc.) |
|---|---|---|---|
| Barlow Twins | Fine-tuning | 71.3 | 50.3 |
| | CaSSLe | **74.9** | **55.5** |
| | Offline | 80.4 | 57.2 |
| SwAV | Fine-tuning | 70.8 | 49.6 |
| | Knowledge | **71.3** | **54.3** |
| | Offline | 74.3 | 54.6 |
| BYOL | Fine-tuning | **74.0** | 50.6 |
| | CaSSLe | 73.3 | **55.1** |
| | Offline | 80.3 | 56.6 |
| VICReg | Fine-tuning | 70.2 | 49.3 |
| | CaSSLe | **72.3** | **52.9** |
| | Offline | 79.4 | 56.7 |
| MoCoV2+ | Fine-tuning | 69.5 | 43.2 |
| | CaSSLe | **71.9** | **46.7** |
| | Offline | 78.2 | 53.7 |
| SimCLR | Fine-tuning | 68.9 | 45.1 |
| | CaSSLe | **72.1** | **50.0** |
| | Offline | 77.5 | 52.6 |
| Supervised | Fine-tuning | 75.9 | 55.9 |
| | Offline | 81.9 | 66.4 |

Table 6.8: Downstream performance with different SSL methods trained on Imagenet-100 and evaluated on DomainNet (Real).

| Strategy | Barlow Twins | SwAV | BYOL | VICReg | MoCoV2+ | SimCLR | Supervised |
|---|---|---|---|---|---|---|---|
| Fine-tune | 56.2 | 55.9 | 55.0 | 54.0 | 52.4 | 51.6 | 54.3 |
| CaSSLe | **60.3** | **56.9** | **56.9** | **56.3** | **58.7** | **56.5** | |

Table 6.9: Top-1 linear accuracy on Imagenet-100 with different SSL methods, semi-supervised setting with 10% and 1% of labels.

| Percentage | Strategy | Barlow Twins | SwAV | BYOL | VICReg | MoCoV2+ | SimCLR | Supervised |
|---|---|---|---|---|---|---|---|---|
| 10% | Fine-tune | 56.6 | 57.6 | 55.7 | 53.6 | 54.9 | 52.5 | 60.8 |
| | CaSSLe | **60.3** | **58.2** | **56.5** | **56.5** | **61.7** | **58.9** | |
| 1% | Fine-tune | 42.6 | 42.5 | 42.3 | 40.4 | 40.9 | 39.7 | 48.1 |
| | CaSSLe | **47.0** | **43.1** | **43.4** | **43.2** | **47.8** | **46.8** | |

# Chapter 7

# Conclusions and Future Research Directions

This thesis explored and proposed novel methods in self-supervised learning applied to unsupervised domain adaptation and continual learning.

First, we proposed an open-source library to unify different methods and training strategies in a way to facilitates further research in the area of self-supervised learning.

Second, we proposed two novel methods for unsupervised domain adaptation that leveraged the recent advances in self-supervised learning. In the first method, we leveraged variations of the contrastive loss that were adapted specifically for performing unsupervised domain adaptation for videos. In the second, we worked on transformer-based architectures and proposed an information bottleneck loss to efficiently learn from both the labeled and the unlabeled data that was available.

Lastly, we also tackled the problem of continual learning. For that, we posed the continual learning problem as an unsupervised task and proposed a simple and effective method that was based on self-supervised losses and a distillation loss derived from the self-supervised losses. By doing so, we show that self-supervised learning is not only a strong pre-training method

by itself, allowing us to train models that are able to produce powerful feature representations of the data, but also serves as an auxiliary method to further improve models in other domains.

## 7.1 Future research directions

Although the current self-supervised methods are powerful, they still require a lot of domain expertise to work and a lot of computational resources. Because of that, the first interesting research direction would be on how to efficiently learn in a self-supervised way within a resource or data-constrained environment.

Additionally, the methods currently explored here are all within the vision domain. Multi-modal methods, such as CLIP [132], are also an interesting research direction. These methods can leverage much more data, not only providing stronger feature representations of visual data but also allowing prompting. Prompting methods allow one to interact with a model via text to change its behavior during test time. Furthermore, multi-modal methods are also currently employed in the few-shot learning scenario. There, instead of having access to a large amount of labeled data from the desired classes, only a few, around 8 or 16, are available. One current promising research direction is in using both parameter-efficient fine-tuning (PEFT) methods and synthetic data, as explored in [39]. PEFT methods will allow us to leverage large pretrained multi-modal models for multiple tasks without the need for expensive fine-tuning schedules.

Another research direction is to use only synthetic data for training DL models. For instance, in [159], the authors train a contrastive learning method on only synthetic data, showing that it can rival training with real data. The synthetic data is generated using a diffusion model that is

guided by image descriptions provided in an open-source large-scale image-text dataset. In [158], they further expand on this idea, first generating a synthetic caption that is then used to generate the synthetic images. However, using synthetic data for training different types of models, *e.g.*, segmentation or object detection methods, is still an unexplored research direction. Furthermore, the current generation process cannot be done in real-time, so it is done as an initial step, carried out before training a model. Improving the speeds of the generation process would also allow one to connect the generation and the training process, creating synthetic data that is specifically tailored to that moment in the training process.

# Bibliography

[1] Alessandro Achille, Tom Eccles, Loic Matthey, Christopher P Burgess, Nick Watters, Alexander Lerchner, and Irina Higgins. Life-long disentangled representation learning with cross-domain latent homologies. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

[2] Idan Achituve, Haggai Maron, and Gal Chechik. Self-supervised learning for domain adaptation on point clouds. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2021.

[3] Farhat Afza, Muhammad Khan, Muhammad Sharif, Seifedine Kadry, Gunasekaran Manogaran, Tanzila Saba, Imran Ashraf, and Robertas Damasevicius. A framework of human action recognition using length control features fusion and weighted entropy-variances based feature selection. *Image and Vision Computing*, 2021.

[4] Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. Memory aware synapses: Learning what (not) to forget. In *Proceedings of the IEEE/CVF European Conference on Computer vision (ECCV)*, 2018.

[5] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[6] Sikai Bai, Qi Wang, and Xuelong Li. Mfi: Multi-range feature interchange for video action recognition. In *2022 26th International Conference on Pattern Recognition (ICPR)*, 2021.

[7] Randall Balestriero, Mark Ibrahim, Vlad Sobal, Ari Morcos, Shashank Shekhar, Tom Goldstein, Florian Bordes, Adrien Bardes, Gregoire Mialon, Yuandong Tian, Avi Schwarzschild, Andrew Gordon Wilson, Jonas Geiping, Quentin Garrido, Pierre Fernandez, Amir Bar, Hamed Pirsiavash, Yann LeCun, and Micah Goldblum. A cookbook of self-supervised learning. *arXiv preprint arXiv:2304.12210*, 2023.

[8] Hangbo Bao, Li Dong, Songhao Piao, and Furu Wei. Beit: Bert pre-training of image transformers. *arXiv preprint arXiv:2106.08254*, 2022.

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

[10] Loïc Barrault, Yu-An Chung, Mariano Coria Meglioli, David Dale, Ning Dong, Mark Duppenthaler, Paul-Ambroise Duquenne, Brian Ellis, Hady Elsahar, Justin Haaheim, et al. Seamless: Multilingual expressive and streaming speech translation. *arXiv preprint arXiv:2312.05187*, 2023.

[11] Gedas Bertasius, Heng Wang, and Lorenzo Torresani. Is space-time attention all you need for video understanding? *arXiv preprint arxiv:2102.05095*, 2021.

[12] C.M. Bishop. *Pattern Recognition and Machine Learning*. Information Science and Statistics. Springer New York, 2016.

[13] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[14] Silvia Bucci, Antonio D'Innocente, and Tatiana Tommasi. Tackling partial domain adaptation with self-supervision. In *International Conference on Image Analysis and Processing*, 2019.

[15] Pietro Buzzega, Matteo Boschini, Angelo Porrello, Davide Abati, and Simone Calderara. Dark experience for general continual learning: a strong, simple baseline. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[16] Lucas Caccia and Joelle Pineau. Special: Self-supervised pretraining for continual learning. *arXiv preprint arXiv:2106.09065*, 2021.

[17] Fabio Maria Carlucci, Lorenzo Porzi, Barbara Caputo, Elisa Ricci, and Samuel Rota Bulò. AutoDIAL: Automatic Domain Alignment Layers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2017.

[18] Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep clustering for unsupervised learning of visual features. In *Proceedings of the IEEE/CVF European Conference on Computer vision (ECCV)*, 2018.

[19] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[20] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[21] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. *arXiv preprint arXiv:2104.14294*, 2021.

[22] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[23] Francisco M Castro, Manuel J Marin-Jimenez, Nicolas Guil, Cordelia Schmid, and Karteek Alahari. End-to-end incremental learning. In *Proceedings of the IEEE/CVF European Conference on Computer vision (ECCV)*, 2018.

[24] Hyuntak Cha, Jaeho Lee, and Jinwoo Shin. Co2l: Contrastive continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[25] Arslan Chaudhry, Puneet K Dokania, Thalaiyasingam Ajanthan, and Philip HS Torr. Riemannian walk for incremental learning: Understanding forgetting and intransigence. In *Proceedings of the IEEE/CVF European Conference on Computer vision (ECCV)*, 2018.

[26] Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with a-gem. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019.

[27] Min-Hung Chen, Zsolt Kira, Ghassan AlRegib, Jaekwon Yoo, Ruxin Chen, and Jian Zheng. Temporal attentive alignment for large-scale video domain adaptation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.

[28] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2020.

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

[30] Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[31] Xinlei Chen, Saining Xie, and Kaiming He. An empirical study of training self-supervised vision transformers. *arXiv preprint arXiv:2104.02057*, 2021.

[32] Yi-Hsin Chen, Wei-Yu Chen, Yu-Ting Chen, Bo-Cheng Tsai, Yu-Chiang Frank Wang, and Min Sun. No More Discrimination: Cross City Adaptation of Road Scene Segmenters. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2017.

[33] Yu Chen, Chunhua Shen, Hao Chen, Xiu-Shen Wei, Lingqiao Liu, and Jian Yang. Adversarial Learning of Structure-Aware Fully Convolutional Networks for Landmark Localization. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2019.

[34] Yi-Bin Cheng, Xipeng Chen, Junhong Chen, Pengxu Wei, Dongyu Zhang, and Liang Lin. Hierarchical transformer: Unsupervised representation learning for skeleton-based human action recognition. In *Proceedings of the IEEE International Conference on Multimedia and Expo*, 2021.

[35] Yi-Bin Cheng, Xipeng Chen, Dongyu Zhang, and Liang Lin. Motion-transformer: Self-supervised pre-training for skeleton-based action recognition. In *Proceedings of the ACM International Conference on Multimedia in Asia*, 2021.

[36] Jinwoo Choi, Gaurav Sharma, Manmohan Chandraker, and Jia-Bin Huang. Unsupervised and semi-supervised domain adaptation for action recognition from drones. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2020.

[37] Jinwoo Choi, Gaurav Sharma, S. Schulter, and J. Huang. Shuffle and attend: Video domain adaptation. In *Proceedings of the IEEE/CVF European Conference on Computer vision (ECCV)*, 2020.

[38] Gabriela Csurka. *Domain adaptation in computer vision applications*. Springer, 2017.

[39] Victor G. Turrisi da Costa, Nicola Dall'Asen, Yiming Wang, Nicu Sebe, and Elisa Ricci. Diversified in-domain synthesis with efficient fine-tuning for few-shot classification. *arXiv preprint arXiv:2312.03046*, 2023.

[40] Victor G. Turrisi da Costa, Enrico Fini, Moin Nabi, Nicu Sebe, and Elisa Ricci. Solo-learn: A library of self-supervised methods for visual representation learning. *arXiv preprint arXiv:2108.01775*, 2021.

[41] Victor G Turrisi da Costa, Giacomo Zara, Paolo Rota, Thiago Oliveira-Santos, Nicu Sebe, Vittorio Murino, and Elisa Ricci. Dual-head contrastive domain adaptation for video action recognition. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2022.

[42] Victor G. Turrisi da Costa, Giacomo Zara, Paolo Rota, Thiago Oliveira-Santos, Nicu Sebe, Vittorio Murino, and Elisa Ricci. Dual-head contrastive domain adaptation for video action recognition. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2022.

[43] Victor G Turrisi da Costa, Giacomo Zara, Paolo Rota, Thiago Oliveira-Santos, Nicu Sebe, Vittorio Murino, and Elisa Ricci. Unsupervised domain adaptation for video transformers in action recognition. In *2022 26th International Conference on Pattern Recognition (ICPR)*, 2022.

[44] Dima Damen, Hazel Doughty, Giovanni Maria Farinella, Sanja Fidler, Antonino Furnari, Evangelos Kazakos, Davide Moltisanti, Jonathan Munro, Toby Perrett, Will Price, and Michael Wray. Scaling egocentric vision: The epic-kitchens dataset. In *Proceedings of the IEEE/CVF European Conference on Computer vision (ECCV)*, 2018.

[45] Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Ales Leonardis, Gregory Slabaugh, and Tinne Tuytelaars. Continual learning: A comparative study on how to defy forgetting in classification tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2019.

[46] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proceed-*

*ings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

[47] Mohammad Mahdi Derakhshani, Enrique Sanchez, Adrian Bulat, Victor G Turrisi da Costa, Cees GM Snoek, Georgios Tzimiropoulos, and Brais Martinez. Bayesian prompt learning for image-language model generalization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.

[48] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2019.

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

[50] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2021.

[51] Arthur Douillard, Matthieu Cord, Charles Ollion, Thomas Robert, and Eduardo Valle. Podnet: Pooled outputs distillation for small-tasks incremental learning. In *Proceedings of the IEEE/CVF European Conference on Computer vision (ECCV)*, 2020.

[52] Debidatta Dwibedi, Yusuf Aytar, Jonathan Tompson, Pierre Sermanet, and Andrew Zisserman. With a little help from my friends:

Nearest-neighbor contrastive learning of visual representations. *arXiv preprint arXiv:2104.14548*, 2021.

[53] Aleksandr Ermolov, Aliaksandr Siarohin, Enver Sangineto, and Nicu Sebe. Whitening for self-supervised representation learning. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2021.

[54] William Falcon, Ananya Harsh Jha, Teddy Koker, and Kyunghyun Cho. Aavae: Augmentation-augmented variational autoencoders. *arXiv preprint arXiv:2107.12329*, 2021.

[55] Hao-Shu Fang, Shuqin Xie, Yu-Wing Tai, and Cewu Lu. RMPE: Regional multi-person pose estimation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2017.

[56] Christoph Feichtenhofer, Axel Pinz, and Andrew Zisserman. Convolutional two-stream network fusion for video action recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

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

[58] Enrico Fini, Stèphane Lathuilière, Enver Sangineto, Moin Nabi, and Elisa Ricci. Online continual learning under extreme memory constraints. In *Proceedings of the IEEE/CVF European Conference on Computer vision (ECCV)*, 2020.

[59] Robert M. French. Catastrophic forgetting in connectionist networks. *Trends in Cognitive Sciences*, 1999.

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

[61] Yaroslav Ganin and Victor Lempitsky. Unsupervised Domain Adaptation by Backpropagation. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2015.

[62] Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2015.

[63] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *Journal of Machine Learning Research*, 2016.

[64] Rohit Girdhar, João Carreira, Carl Doersch, and Andrew Zisserman. Video action transformer network. *arXiv preprint arXiv:1812.02707*, 2019.

[65] 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.

[66] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[67] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio.

Generative adversarial nets. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, 2014.

[68] Priya Goyal, Quentin Duval, Jeremy Reizenstein, Matthew Leavitt, Min Xu, Benjamin Lefaudeux, Mannat Singh, Vinicius Reis, Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Ishan Misra. Vissl. *GitHub. Note: https://github.com/facebookresearch/vissl*, 2021.

[69] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, Bilal Piot, koray kavukcuoglu, Remi Munos, and Michal Valko. Bootstrap your own latent - a new approach to self-supervised learning. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[70] Michael Gutmann and Aapo Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the AISTATS*, 2010.

[71] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. *arXiv preprint arXiv:2111.06377*, 2021.

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

[73] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015.

[74] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[75] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[76] Judy Hoffman, Eric Tzeng, Taesung Park, Jun-Yan Zhu, Phillip Isola, Kate Saenko, Alexei A Efros, and Trevor Darrell. CyCADA: Cycle-Consistent Adversarial Domain Adaptation. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2018.

[77] Judy Hoffman, Dequan Wang, Fisher Yu, and Trevor Darrell. FCNs in the Wild: Pixel-level Adversarial and Constraint-based Adaptation. *arXiv preprint arXiv:1612.02649*, 2016.

[78] Weixiang Hong, Zhenzhen Wang, Ming Yang, and Junsong Yuan. Conditional Generative Adversarial Network for Structured Domain Adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[79] Saihui Hou, Xinyu Pan, Chen Change Loy, Zilei Wang, and Dahua Lin. Learning a unified classifier incrementally via rebalancing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[80] Ziyuan Huang, Zhiwu Qing, Xiang Wang, Yutong Feng, Shiwei Zhang, Jianwen Jiang, Zhurong Xia, Mingqian Tang, Nong Sang, and Marcelo H. Ang Jr au2. Towards training stronger video vision transformers for epic-kitchens-100 action recognition. *arXiv preprint arXiv:2106.05058*, 2021.

[81] Ahmet Iscen, Jeffrey Zhang, Svetlana Lazebnik, and Cordelia Schmid. Memory-efficient incremental learning through feature adaptation. In *Proceedings of the IEEE/CVF European Conference on Computer vision (ECCV)*, 2020.

[82] Arshad Jamal, Vinay P Namboodiri, Dipti Deodhare, and KS Venkatesh. Deep domain adaptation in action space. In *Proceedings of the British Machine Vision Conference (BMVC)*, 2018.

[83] Heinrich Jiang and Ofir Nachum. Identifying and correcting label bias in machine learning. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, 2020.

[84] Guoliang Kang, Lu Jiang, Yi Yang, and Alexander G Hauptmann. Contrastive adaptation network for unsupervised domain adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[85] Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, et al. The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950*, 2017.

[86] Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. Supervised contrastive learning. *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[87] Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. Supervised contrastive learning. *arXiv preprint arXiv:2004.11362*, 2021.

[88] Donghyun Kim, Kuniaki Saito, Tae-Hyun Oh, Bryan A Plummer, Stan Sclaroff, and Kate Saenko. Cross-domain self-supervised learning for domain adaptation with few source labels. *arXiv preprint arXiv:2003.08264*, 2020.

[89] Donghyun Kim, Yi-Hsuan Tsai, Bingbing Zhuang, Xiang Yu, Stan Sclaroff, Kate Saenko, and Manmohan Chandraker. Learning cross-modal contrastive features for video domain adaptation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

[90] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross Girshick. Segment anything. *arXiv preprint arXiv:2304.02643*, 2023.

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

[92] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, Univ. Toronto, 2009.

[93] Hildegard Kuehne, Hueihan Jhuang, Estíbaliz Garrote, Tomaso Poggio, and Thomas Serre. Hmdb: a large video database for human motion recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2011.

[94] Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. Learning representations for automatic colorization. In *Proceedings of the IEEE/CVF European Conference on Computer vision (ECCV)*, 2016.

[95] Junnan Li, Dongxu Li, Caiming Xiong, and Steven Hoi. Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation. *arXiv preprint arXiv:2201.12086*, 2022.

[96] W. Li, Fuyu Li, Yongkang Luo, and Peng Wang. Deep domain adaptive object detection: a survey. In *Proceedings of the IEEE Symposium Series on Computational Intelligence (SSCI)*, 2020.

[97] Xiangyu Li, Yonghong Hou, Pichao Wang, Zhimin Gao, Mingliang Xu, and Wanqing Li. Trear: Transformer-based rgb-d egocentric action recognition. *IEEE Transactions on Cognitive and Developmental Systems*, 2020.

[98] Yanghao Li, Naiyan Wang, Jianping Shi, Jiaying Liu, and Xiaodi Hou. Revisiting batch normalization for practical domain adaptation. In *Proceedings of the International Conference on Learning Representations (ICLR) Workshop*, 2017.

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

[100] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. *arXiv preprint arXiv:1405.0312*, 2014.

[101] Zhiwei Lin, Yongtao Wang, and Hongxiang Lin. Continual contrastive self-supervised learning for image classification. *arXiv preprint arXiv:2107.01776*, 2021.

[102] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vi-

sion transformer using shifted windows. *International Conference on Computer Vision (ICCV)*, 2021.

[103] Mingsheng Long, Zhangjie Cao, Jianmin Wang, and Michael I Jordan. Conditional Adversarial Domain Adaptation. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

[104] Mingsheng Long and Jianmin Wang. Learning Transferable Features with Deep Adaptation Networks. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2015.

[105] Mingsheng Long, Han Zhu, Jianmin Wang, and Michael I Jordan. Deep transfer learning with joint adaptation networks. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2017.

[106] David Lopez-Paz and Marc-Aurelio Ranzato. Gradient episodic memory for continual learning. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[107] Kevin Lu, Aditya Grover, Pieter Abbeel, and Igor Mordatch. Pretrained transformers as universal computation engines. *arXiv preprint arXiv:2103.05247*, 2021.

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

[109] Massimiliano Mancini, Samuel Rota Bulo, Barbara Caputo, and Elisa Ricci. Adagraph: Unifying predictive and continuous domain adaptation through graphs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[110] Massimiliano Mancini, Lorenzo Porzi, Samuel Rota Bulò, Barbara Caputo, and Elisa Ricci. Boosting Domain Adaptation by Discovering Latent Domains. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[111] Vittorio Mazzia, Simone Angarano, Francesco Salvetti, Federico Angelini, and Marcello Chiaberge. Action transformer: A self-attention model for short-time pose-based human action recognition. *Pattern Recognition*, Apr 2022.

[112] Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, 1989.

[113] Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2020.

[114] Yue Meng, Rameswar Panda, Chung-Ching Lin, Prasanna Sattigeri, Leonid Karlinsky, Kate Saenko, Aude Oliva, and Rogerio Feris. Adafuse: Adaptive temporal fusion network for efficient action recognition. *arXiv preprint arXiv:2102.05775*, 2021.

[115] Mathew Monfort, Bolei Zhou, Sarah Adel Bargal, Alex Andonian, Tom Yan, Kandan Ramakrishnan, Lisa M. Brown, Quanfu Fan, Dan Gutfreund, Carl Vondrick, and Aude Oliva. Moments in time dataset: one million videos for event understanding. *arXiv preprint arXiv:1801.03150*, 2018.

[116] Saeid Motiian, Marco Piccirilli, Donald A Adjeroh, and Gianfranco Doretto. Unified deep supervised domain adaptation and generalization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2017.

[117] Jonathan Munro and Dima Damen. Multi-modal domain adaptation for fine-grained action recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[118] Daniel Neimark, Omri Bar, Maya Zohar, and Dotan Asselmann. Video transformer network. *arXiv preprint arXiv:2102.00719*, 2021.

[119] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

[120] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mahmoud Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Hervé Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

[121] Oleksiy Ostapenko, Mihai Puscas, Tassilo Klein, Patrick Jahnichen, and Moin Nabi. Learning to remember: A synaptic plasticity driven framework for continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[122] Boxiao Pan, Zhangjie Cao, Ehsan Adeli, and Juan Carlos Niebles. Adversarial cross-domain action recognition with co-attention. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020.

[123] Changhwa Park, Jonghyun Lee, Jaeyoon Yoo, Minhoe Hur, and Sungroh Yoon. Joint contrastive learning for unsupervised domain adaptation. *arXiv preprint arXiv:2006.10297*, 2020.

[124] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[125] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[126] Mandela Patrick, Yuki M Asano, Polina Kuznetsova, Ruth Fong, João F Henriques, Geoffrey Zweig, and Andrea Vedaldi. Multi-modal self-supervision from generalized data transformations. *arXiv preprint arXiv:2003.04298*, 2020.

[127] Xingchao Peng, Qinxun Bai, Xide Xia, Zijun Huang, Kate Saenko, and Bo Wang. Moment matching for multi-source domain adaptation. In *International Conference on Computer Vision (ICCV)*, 2019.

[128] Chiara Plizzari, Marco Cannici, and Matteo Matteucci. Spatial temporal transformer network for skeleton-based action recognition. In Alberto Del Bimbo, Rita Cucchiara, Stan Sclaroff, Giovanni Maria Farinella, Tao Mei, Marco Bertini, Hugo Jair Escalante, and Roberto Vezzani, editors, *In Proceedings of International Conference on Learning Representations (ICPR) Workshops*. Springer International Publishing, 2021.

[129] Ameya Prabhu, Philip HS Torr, and Puneet K Dokania. Gdumb: A simple approach that questions our progress in continual learning. In *Proceedings of the IEEE/CVF European Conference on Computer vision (ECCV)*. Springer, 2020.

[130] Albert Pumarola, Jordi Sanchez, Gary Choi, Alberto Sanfeliu, and Francesc Moreno-Noguer. 3DPeople: Modeling the Geometry of Dressed Humans. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.

[131] Rui Qian, Tianjian Meng, Boqing Gong, Ming-Hsuan Yang, Huisheng Wang, Serge Belongie, and Yin Cui. Spatiotemporal contrastive video representation learning. *arXiv preprint arXiv:2008.03800*, 2020.

[132] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. *arXiv preprint arXiv:2103.00020*, 2021.

[133] Dushyant Rao, Francesco Visin, Andrei A Rusu, Yee Whye Teh, Razvan Pascanu, and Raia Hadsell. Continual unsupervised representation learning. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[134] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[135] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[136] Anthony Robins. Catastrophic forgetting, rehearsal and pseudorehearsal. *Connection Science*, 1995.

[137] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. *arXiv preprint arXiv:2112.10752*, 2022.

[138] Subhankar Roy, Aliaksandr Siarohin, Enver Sangineto, Samuel Rota Bulò, Nicu Sebe, and Elisa Ricci. Unsupervised Domain Adaptation using Feature-Whitening and Consensus Loss. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[139] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, and R. Hadsell. Progressive neural networks. *arXiv 1606.04671*, 2016.

[140] Kuniaki Saito, Kohei Watanabe, Yoshitaka Ushiku, and Tatsuya Harada. Maximum classifier discrepancy for unsupervised domain adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[141] Swami Sankaranarayanan, Yogesh Balaji, Carlos D Castillo, and Rama Chellappa. Generate to adapt: Aligning domains using gen-

erative adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[142] Joan Serra, Didac Suris, Marius Miron, and Alexandros Karatzoglou. Overcoming catastrophic forgetting with hard attention to the task. *Proceedings of the International Conference on Machine Learning (ICML)*, 2018.

[143] Gilad Sharir, Asaf Noy, and Lihi Zelnik-Manor. An image is worth 16x16 words, what is a video worth? *arXiv preprint arXiv:2103.13915*, 2021.

[144] Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual learning with deep generative replay. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[145] Rui Shu, Hung H Bui, Hirokazu Narui, and Stefano Ermon. A dirt-t approach to unsupervised domain adaptation. *arXiv preprint arXiv:1802.08735*, 2018.

[146] Gunnar A Sigurdsson, Abhinav Gupta, Cordelia Schmid, Ali Farhadi, and Karteek Alahari. Actor and observer: Joint modeling of first and third-person videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[147] Lucas Smaira, João Carreira, Eric Noland, Ellen Clancy, Amy Wu, and Andrew Zisserman. A short note on the kinetics-700-2020 human action dataset. *arXiv preprint arXiv:2010.10864*, 2020.

[148] James Smith, Cameron Taylor, Seth Baer, and Constantine Dovrolis. Unsupervised progressive learning and the stam architecture. *arXiv preprint arXiv:1904.02021*, 2019.

[149] Xiaolin Song, Sicheng Zhao, Jingyu Yang, Huanjing Yue, Pengfei Xu, Runbo Hu, and Hua Chai. Spatio-temporal contrastive domain adaptation for action recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2021.

[150] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012.

[151] Alexandros Stergiou and Ronald Poppe. Learn to cycle: Time-consistent feature discovery for action recognition. *Pattern Recognition Letters*, 2021.

[152] Waqas Sultani and Imran Saleemi. Human action recognition across datasets by foreground-weighted histogram decomposition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[153] Yan Sun, Yixin Shen, and Liyan Ma. Msst-rt: Multi-stream spatial-temporal relative transformer for skeleton-based action recognition. *Sensors*, 2021.

[154] Yu Sun, Eric Tzeng, Trevor Darrell, and Alexei A Efros. Unsupervised domain adaptation through self-supervision. *arXiv preprint arXiv:1909.11825*, 2019.

[155] Igor Susmelj, Matthias Heller, Philipp Wirth, Jeremy Prescott, and Malte Ebner et al. Lightly. *GitHub. Note: https://github.com/lightly-ai/lightly*, 2020.

[156] OpenAI Team. Gpt-4 technical report, 2023.

[157] Pytorch Lightning Development Team. Pytorch lightning. *GitHub. Note: https://github.com/PyTorchLightning/pytorch-lightning*, 2019.

[158] Yonglong Tian, Lijie Fan, Kaifeng Chen, Dina Katabi, Dilip Krishnan, and Phillip Isola. Learning vision from models rivals learning vision from data. *arXiv preprint arXiv:2312.17742*, 2023.

[159] Yonglong Tian, Lijie Fan, Phillip Isola, Huiwen Chang, and Dilip Krishnan. Stablerep: Synthetic images from text-to-image models make strong visual representation learners. *arXiv preprint arXiv:2306.00984*, 2023.

[160] Yonglong Tian, Olivier J Henaff, and Aaron van den Oord. Divide and contrast: Self-supervised learning from uncurated data. *arXiv preprint arXiv:2105.08054*, 2021.

[161] Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive multiview coding. In *Proceedings of the IEEE/CVF European Conference on Computer vision (ECCV)*. Springer, 2020.

[162] Yuandong Tian, Xinlei Chen, and Surya Ganguli. Understanding self-supervised learning dynamics without contrastive pairs. In *International Conference on Machine Learning (ICML)*. PMLR, 2021.

[163] Naftali Tishby, Fernando C Pereira, and William Bialek. The information bottleneck method. *arXiv preprint physics/0004057*, 2000.

[164] Naftali Tishby and Noga Zaslavsky. Deep learning and the information bottleneck principle. *arXiv preprint arXiv:1503.02406*, 2015.

[165] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu,

Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

[166] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3d convolutional networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2015.

[167] Eric Tzeng, Judy Hoffman, Trevor Darrell, and Kate Saenko. Adversarial discriminative domain adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[168] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. Adversarial Discriminative Domain Adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[169] Eric Tzeng, Judy Hoffman, Ning Zhang, Kate Saenko, and Trevor Darrell. Deep domain confusion: Maximizing for domain invariance. *arXiv preprint arXiv:1412.3474*, 2014.

[170] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2019.

[171] Gül Varol, Javier Romero, Xavier Martin, Naureen Mahmood, Michael J. Black, Ivan Laptev, and Cordelia Schmid. Learning from synthetic humans. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[172] Gül Varol, Ivan Laptev, Cordelia Schmid, and Andrew Zisserman. Synthetic humans for action recognition from unseen viewpoints. *International Journal of Computer Vision*, 2021.

[173] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *arXiv preprint arXiv:1706.03762*, 2017.

[174] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *arXiv preprint arXiv:1706.03762*, 2023.

[175] Riccardo Volpi, Pietro Morerio, Silvio Savarese, and Vittorio Murino. Adversarial feature augmentation for unsupervised domain adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[176] Apoorv Vyas, Bowen Shi, Matthew Le, Andros Tjandra, Yi-Chiao Wu, Baishan Guo, Jiemin Zhang, Xinyue Zhang, Robert Adkins,

William Ngan, et al. Audiobox: Unified audio generation with natural language prompts. *arXiv preprint arXiv:2312.15821*, 2023.

[177] Jiangliu Wang, Jianbo Jiao, and Yun-Hui Liu. Self-supervised video representation learning by pace prediction. In *Proceedings of the IEEE/CVF European Conference on Computer vision (ECCV)*, 2020.

[178] Jin woo Choi, Gaurav Sharma, S. Schulter, and Jia-Bin Huang. Shuffle and attend: Video domain adaptation. In *Proceedings of the IEEE/CVF European Conference on Computer vision (ECCV)*, 2020.

[179] Yue Wu, Yinpeng Chen, Lijuan Wang, Yuancheng Ye, Zicheng Liu, Yandong Guo, and Yun Fu. Large scale incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[180] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. `https://github.com/facebookresearch/detectron2`, 2019.

[181] Zhirong Wu, Yuanjun Xiong, Stella X Yu, and Dahua Lin. Unsupervised feature learning via non-parametric instance discrimination. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[182] Liang Xiao, Jiaolong Xu, Dawei Zhao, Zhiyu Wang, Li Wang, Yiming Nie, and Bin Dai. Self-supervised domain adaptation with consistency training. In *2020 25th International Conference on Pattern Recognition (ICPR)*. IEEE, 2021.

[183] Zhenda Xie, Zheng Zhang, Yue Cao, Yutong Lin, Jianmin Bao, Zhuliang Yao, Qi Dai, and Han Hu. Simmim: A simple framework for masked image modeling. *arXiv preprint arXiv:2111.0988*, 2022.

[184] Jiaolong Xu, Liang Xiao, and Antonio M López. Self-supervised domain adaptation for computer vision tasks. *IEEE Access*, 2019.

[185] Tiantian Xu, Fan Zhu, Edward K Wong, and Yi Fang. Dual many-to-one-encoder-based transfer learning for cross-dataset human action recognition. *Image and Vision Computing*, 2016.

[186] Tongkun Xu, Weihua Chen, Pichao Wang, Fan Wang, Hao Li, and Rong Jin. Cdtrans: Cross-domain transformer for unsupervised domain adaptation. *arXiv preprint arXiv:2105.14138*, 2021.

[187] Guanglei Yang, Hao Tang, Zhun Zhong, Mingli Ding, Ling Shao, Nicu Sebe, and Elisa Ricci. Transformer-based source-free domain adaptation. *arXiv preprint arXiv:2105.14138*, 2021.

[188] Xitong Yang, Xiaodong Yang, Sifei Liu, Deqing Sun, Larry Davis, and Jan Kautz. Hierarchical contrastive motion learning for video action recognition. *arXiv preprint arXiv:2007.10321*, 2020.

[189] Yang You, Igor Gitman, and Boris Ginsburg. Large batch training of convolutional networks. *arXiv preprint arXiv:1708.03888*, 2017.

[190] Giacomo Zara, Victor Guilherme Turrisi da Costa, Subhankar Roy, Paolo Rota, and Elisa Ricci. Simplifying open-set video domain adaptation with contrastive learning. *arXiv preprint arXiv:2301.03322*, 2023.

[191] Jure Zbontar, Li Jing, Ishan Misra, Yann Lecun, and Stephane Deny. Barlow twins: Self-supervised learning via redundancy reduction. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2021.

[192] Friedeman Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2017.

[193] Peiyuan Zhang, Guangtao Zeng, Tianduo Wang, and Wei Lu. Tinyllama: An open-source small language model. *arXiv preprint arXiv:2401.02385*, 2024.

[194] Yang Zhang, Philip David, and Boqing Gong. Curriculum Domain Adaptation for Semantic Segmentation of Urban Scenes. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2017.

[195] Mingkai Zheng, Shan You, Fei Wang, Chen Qian, Changshui Zhang, Xiaogang Wang, and Chang Xu. Ressl: Relational self-supervised learning with weak augmentation. *arXiv preprint arXiv:2107.09282*, 2021.

[196] Bolei Zhou, Alex Andonian, Aude Oliva, and Antonio Torralba. Temporal relational reasoning in videos. In *Proceedings of the IEEE/CVF European Conference on Computer vision (ECCV)*, 2018.

[197] Jinghao Zhou, Chen Wei, Huiyu Wang, Wei Shen, Cihang Xie, Alan Yuille, and Tao Kong. ibot: Image bert pre-training with online tokenizer. *arXiv preprint arXiv:2111.07832*, 2022.

[198] Xingyi Zhou, Arjun Karpur, Chuang Gan, Linjie Luo, and Qixing Huang. Unsupervised Domain Adaptation for 3D Keypoint Estimation via View Consistency. In *Proceedings of the IEEE/CVF European Conference on Computer vision (ECCV)*, 2018.

# Appendix A

# Supplementary Material: Dual-head contrastive domain adaptation for video action recognition

## A.1 The Mixamo dataset

This Section describes additional details and statistics about our Mixamo dataset. In Table A.1, we report the number of videos and frames for each class in our dataset. For comparison purposes, we also report the same information for the corresponding Kinetics subset. Figures A.1 and A.2 provides a visual overview of the distribution of the number of frames and the number of videos across the two datasets.

Table A.1: Number of videos and frames in Mixamo and Kinetics

| **Class** | **# videos** | | **# frames** | |
|---|---|---|---|---|
| | **Mixamo** | **Kinetics** | **Mixamo** | **Kinetics** |
| backflip | 959 | 844 | 83,717 | 51,879 |
| breakdancing | 2304 | 829 | 238,464 | 63,613 |
| capoeira | 3456 | 940 | 326,304 | 75,102 |
| clapping | 1344 | 934 | 175,488 | 74,469 |
| golf putting | 1037 | 650 | 100,370 | 55,567 |
| jogging | 2304 | 719 | 143,424 | 60,808 |
| punching | 2016 | 577 | 108,784 | 52,114 |
| salsa dancing | 960 | 517 | 326,880 | 42,544 |
| shouting | 1248 | 680 | 148,224 | 52,407 |
| side kick | 2304 | 970 | 142,272 | 73,998 |
| squat | 2081 | 888 | 265,833 | 74,906 |
| swing dancing | 1304 | 750 | 599,055 | 64,723 |
| texting | 1296 | 548 | 432,000 | 48,690 |
| throwing | 1920 | 1816 | 221,760 | 155,869 |

Furthermore, the dataset presents a rich internal variability within each action class. That is, each class is divided into a number of sub-classes, each associated to a different way of performing the same action. For

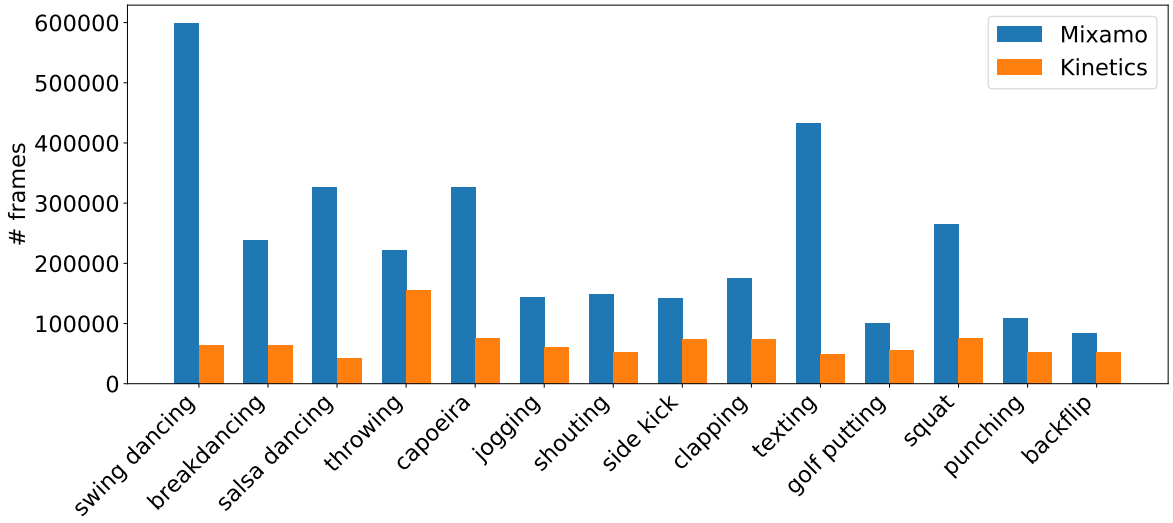



Figure A.1: Distribution of frames per class across Mixamo and Kinetics

instance, the *jogging* class includes 8 sub-actions, which consists of unique animations: *jog forward*, *jogging with box*, *jog forward diagonal*, *injured jog*, *slow jog*, *jogging*, *jog in circle*, *jogging stumble*. Figure A.3 shows the number of unique sub-classes for each one of the original 14 categories in our synthetic dataset.

## A.2 Single-head versus dual-head ablation

For completeness, we ablate the usage of the proposed dual-head architecture. Although it is not possible to apply $\mathcal{L}_{\mathcal{ST}}$ without the two heads, it is possible to use the other losses in a single-head architecture. By doing so, the contrastive losses can directly influence the classification head, which could lead to better feature representations. However, in Table A.2, we show that, by doing so, the contrastive losses are detrimental for the performance of the model. Although on *UCF→HMDB* the performance of a single-head model is slightly better than the dual-head model, in the other direction, *HMDB→UCF*, there is a large drop in accuracy. Lastly,

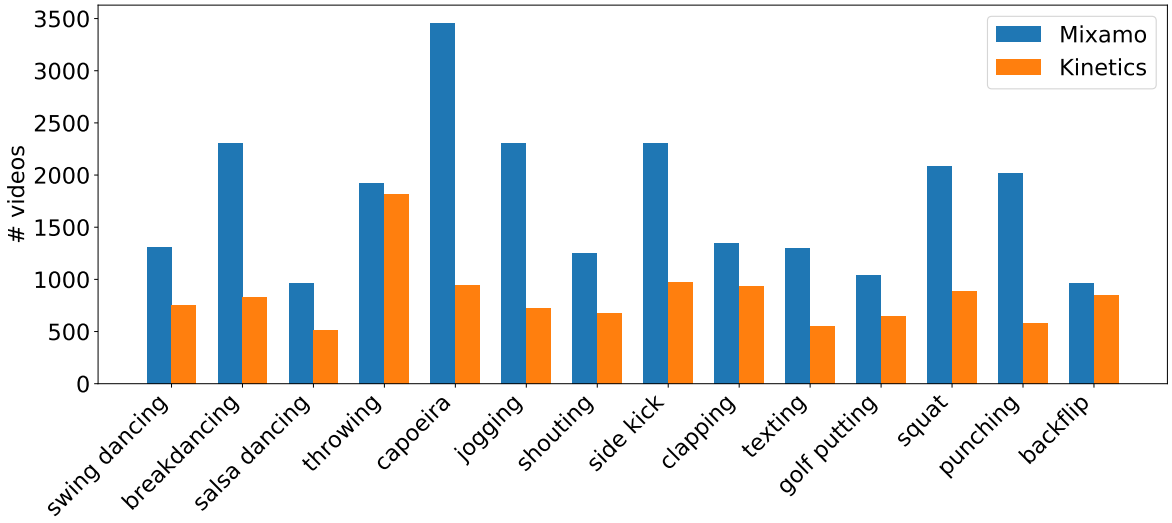



Figure A.2: Distribution of videos per class across Mixamo and Kinetics

Table A.2: Ablation of single-head versus multi-head on *HMDB↔UCF*

| **Method** | **H→U** | **U→H** |
|---|---|---|
| CO$^2$A dual-head w/o $\mathcal{L}_{\mathcal{ST}}$ | 94.4 | 82.4 |
| CO$^2$A single-head w/o $\mathcal{L}_{\mathcal{ST}}$ | 91.4 | 82.9 |
| CO$^2$A (full) | **95.8** | **87.8** |

the single-head approach is greatly outperformed by the full model.

## A.3 Additional sensitivity analysis

In Figure A.4, we also ablated the sensitivity of our method to different weights considering the losses $\mathcal{L}_{\mathcal{C}_c}$ and $\mathcal{L}_{\mathcal{C}_v}$. Note that we decoupled $w_c$ into $w_c^c$ for the clip-level loss and $w_c^v$ for the video-level loss. First, considering $\mathcal{L}_{\mathcal{C}_c}$, we can see that our method achieves the best performance for a value of the weight equal to 0.2. This indicates a trade-off between a condition in which the loss has enough weight to guide representation learning at

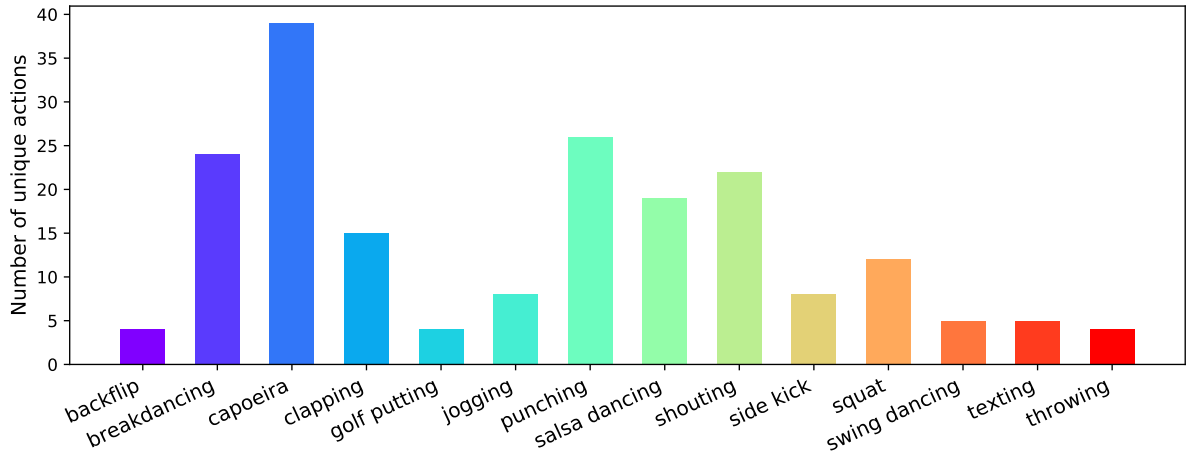



Figure A.3: Distribution of unique actions per class

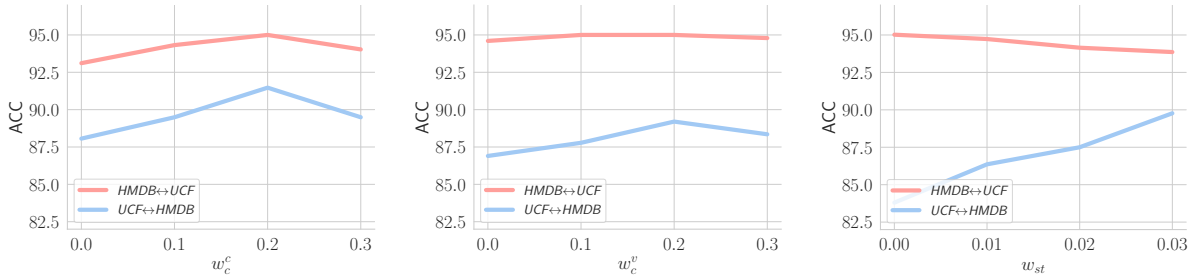



Figure A.4: Sensitivity analysis of the weights of the losses $\mathcal{L}_{\mathcal{C}_c}$, $\mathcal{L}_{\mathcal{C}_v}$ and $\mathcal{L}_{\mathcal{SC}}$.

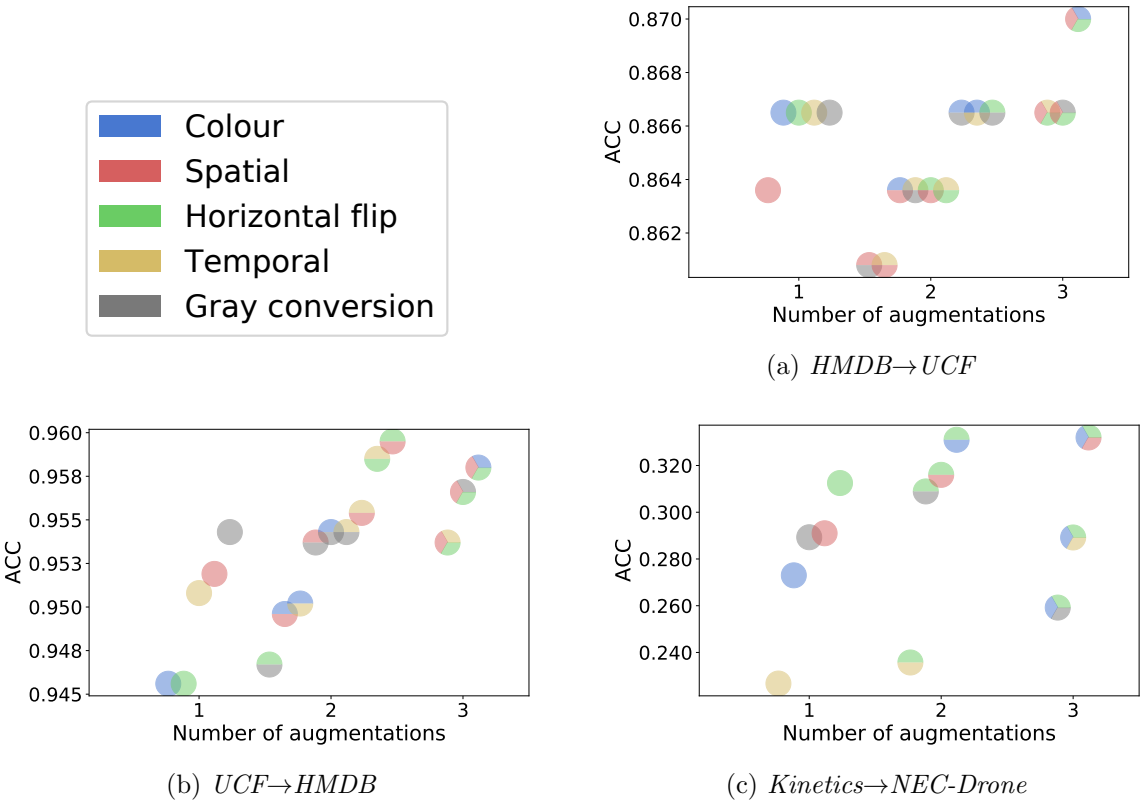
the clip-level and a condition where it dominates other losses. A very similar behaviour is observed for $\mathcal{L}_{\mathcal{C}_v}$, even if the impact of this loss on *HMDB→UCF* is less pronounced. Nonetheless, for the other direction, where the domains are quite different, using a good value for the weight of $\mathcal{L}_{\mathcal{C}_v}$ results in an accuracy of around 2% higher.

## A.4 Augmentation details

Video-based augmentations were applied similarly to [131]. More specifically, frame-wise augmentations are performed keeping time consistency, *i.e.*, for each video, the parameters for the augmentations are randomised

once, and then applied to all frames equally. Colour, spatial and random horizontal flip augmentations were applied only to the target data. The colour augmentation parameters for *torchvision* were 0.15 for the brightness, contrast and saturation, and 0.05 for hue. Spatial augmentation was performed by resizing the image to 256 by 256 and randomly cropping it to be of size 224 by 224 (using the default parameters). In *Mixamo→Kinetics* we also applied a temporal augmentation, which simply samples the total amount of frames, orders and then divides them into the $K$ clips. For source data, we applied the same augmentations only in *Mixamo→Kinetics*. In settings where no augmentations were applied, images are simply resized to 256 and are centrally cropped with a size of 224. Horizontal flip is applied with a 50% probability.

For completeness, we also ablated our method with different combinations of augmentations on *HMDB↔UCF* and *Kinetics→NEC-Drone* in Figure A.5. In Figure A.5 (a) we can observe that our method is not sensitive to the choice of the augmentations and different combinations of augmentations achieve very similar performance. However, In Figure A.5 (b), we can see that the combination of *colour + spatial + horizontal* outperforms other configurations. Lastly, in the more challenging setting of *Kinetics→NEC-Drone* (Figure A.5 (c)), we can see that the choice of augmentations is more important, with *colour + horizontal* and *colour + spatial + horizontal* performing very similarly. Because *colour + spatial + horizontal* performs best on *UCF→HMDB* and *Kinetics→NEC-Drone* and is only slightly inferior to the best combination on *HMDB→UCF* we selected it as a good default combination across these datasets.




(a) *HMDB→UCF*

(b) *UCF→HMDB*

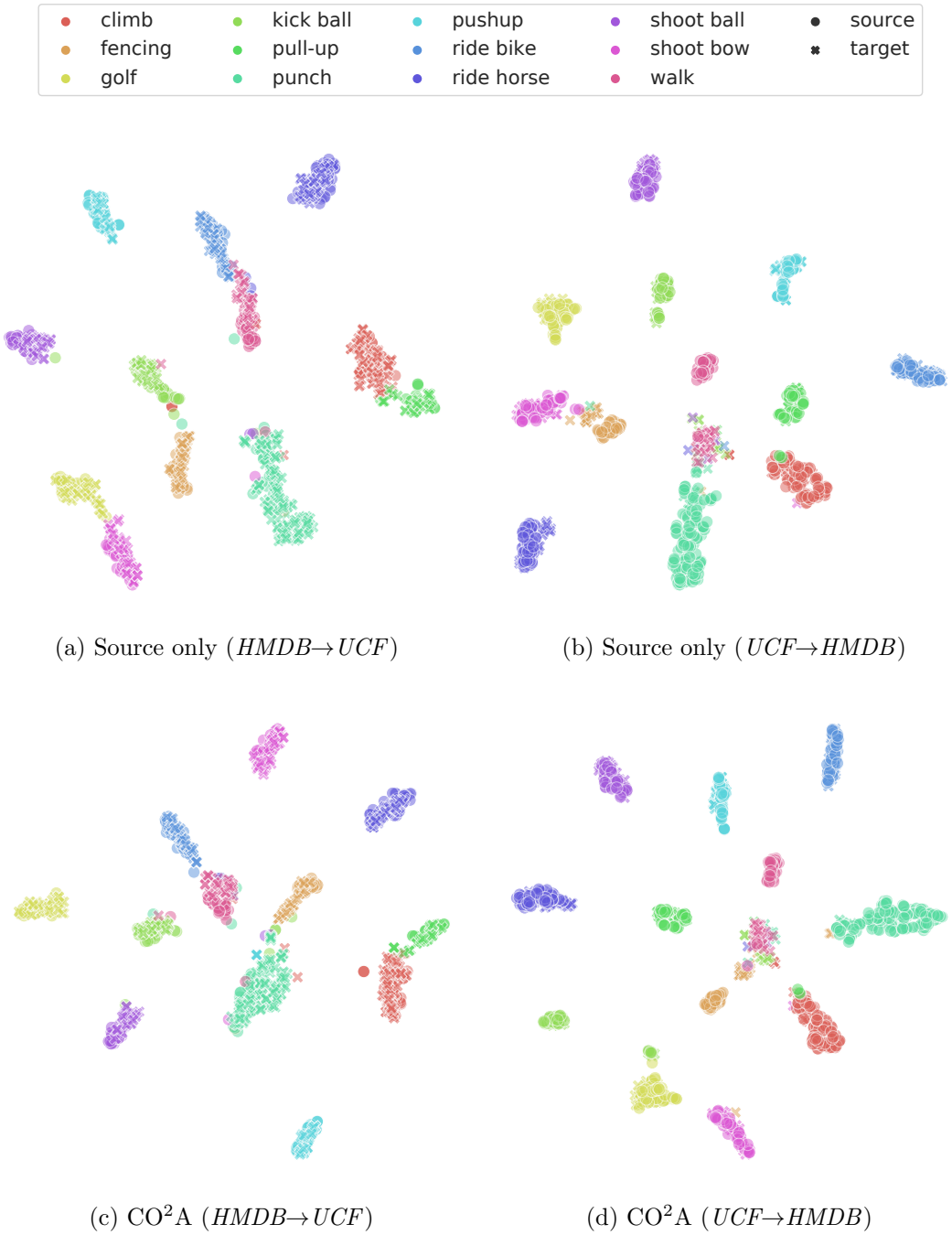
(c) *Kinetics→NEC-Drone*

Figure A.5: Ablation of different augmentations using CO$^2$A on *HMDB↔UCF* and *Kinetics→NEC-Drone*.

## A.5 Visualisation of the learned representations

In Figure A.6 we visualise the features before the linear classifier on the test data for *HMDB↔UCF* when considering a source only model and CO2A. First, considering *HMDB→UCF*, our model produces more compact clusters when considering the majority of classes. Also, it is able to better separate some classes, e.g., *golf* from *shoot bow* and *pull-up* from *fencing*. On *UCF→HMDB* the classes' clusters are more compact, but we can also observe that some clusters became even more compact, e.g., *punch* and *kick ball*. Likewise, it better-separated *fencing* and *shoot bow*. On both directions, we observe that the circles and crosses (source and target domains) have more overlap, indicating that the adaptation procedure better

aligns both domains.




(a) Source only (*HMDB→UCF*)

(b) Source only (*UCF→HMDB*)

(c) CO$^2$A (*HMDB→UCF*)

(d) CO$^2$A (*UCF→HMDB*)

Figure A.6: t-SNE plots of test data on *HMDB↔UCF* for a source only model versus CO2A.

# Appendix B

# Supplementary Material: Self-supervised Models are Continual Learners

## B.1 PyTorch-like pseudo-code

We provide a PyTorch-like pseudo-code of our method. As you can see, CaSSLe is simple to implement and does not add much complexity to the base SSL method. In this snippet, the losses are made symmetric by summing the two contributions. In some cases, the two losses are averaged instead. In CaSSLe, we symmetrize in the same way as the base SSL method we are considering.

**Algorithm 1** PyTorch-like pseudo-code for CaSSLe.

```
# aug: stochastic image augmentation
# f: backbone and projector
# frozen_f: frozen backbone and projector
# g: CaSSLe's predictor
# loss_fn: any SSL loss in Tab. 1 (main paper)

# PyTorchLightning handles loading and optimization
def training_step(x):

    # correlated views
    x1, x2 = aug(x), aug(x)

    # forward backbone and projector
    z1, z2 = f(x1), f(x2)

    # optionally forward predictor...

    # compute SSL loss (symmetric)
    ssl_loss = loss_fn(z1, z2) \\
             + loss_fn(z2, z1)

    # forward frozen backbone and projector
    z1_bar, z2_bar = frozen_f(x1), frozen_f(x2)

    # compute distillation loss (symmetric)
    distill_loss = loss_fn(g(z1), z1_bar) \\
                 + loss_fn(g(z2), z2_bar)

    # no hyperparameter for loss weighting
    return ssl_loss + distill_loss
```

## B.2 Derivation of distillation losses

In this section, we derive distillation losses from the SSL losses in Tab. 1 of the main paper, starting from the definition of our distillation loss:

$$\mathcal{L}_D(\boldsymbol{z}, \bar{\boldsymbol{z}}) = \mathcal{L}_{SSL}(g(\boldsymbol{z}), \bar{\boldsymbol{z}}), \tag{B.1}$$

where $z$ and $\bar{z}$ are the representations of the current and frozen encoder, and $g$ is CaSSLe's predictor network implemented as a two-layer MLP with 2048 hidden neurons and ReLU activation.

**Contrastive based.** Our distillation loss based on contrastive learning is implemented as follows:

$$\mathcal{L}(z_i, \bar{z}_i) = -\log \frac{\exp\left(\text{sim}\left(\boldsymbol{z}_i, \bar{\boldsymbol{z}}_i\right)/\tau\right)}{\sum_{\boldsymbol{z}_j \in \bar{\eta}(i)} \exp\left(\text{sim}\left(\boldsymbol{z}_i, \boldsymbol{z}_j\right)/\tau\right)}, \tag{B.2}$$

where $\bar{\eta}(i)$ is the set of negatives for the sample with index $i$ in the batch. Note that the negatives are drawn both from the predicted and frozen features.

**MSE based.** This distillation loss is simply the MSE between the predicted features and the frozen features:

$$\mathcal{L}(z, \bar{z}) = -||g(\boldsymbol{z}) - \bar{\boldsymbol{z}}||_2^2. \tag{B.3}$$

It can be implemented with the cosine similarity as stated in the main manuscript.

**Cross-entropy based.** The cross-entropy loss, when used for distillation in an unsupervised setting, makes sure that the current encoder is able to assign samples to the frozen centroids (or prototypes) consistently with the frozen encoder:

$$\mathcal{L}(z, \bar{z}) = -\sum_d \bar{\boldsymbol{a}}_d \log \frac{\exp\left(\text{sim}\left(g(\boldsymbol{z}), \boldsymbol{c}_d^{t-1}\right)/\tau\right)}{\sum_k \exp\left(\text{sim}\left(g(\boldsymbol{z}), \boldsymbol{c}_k^{t-1}\right)/\tau\right)} \tag{B.4}$$

where:

$$\bar{\boldsymbol{a}} = \frac{\exp\left(\text{sim}\left(\bar{\boldsymbol{z}}, \boldsymbol{c}_d^{t-1}\right)/\tau\right)}{\sum_k \exp\left(\text{sim}\left(\bar{\boldsymbol{z}}, \boldsymbol{c}_k^{t-1}\right)/\tau\right)}, \tag{B.5}$$

and the set of frozen prototypes is denoted as: $\mathbf{C}^{t-1} = \left\{\mathbf{c}_1^{t-1}, \ldots, \mathbf{c}_K^{t-1}\right\}$.

**Cross-correlation based.** We consider Barlow Twins' [191] implementation of this objective. For VICReg [9] we only consider the invariance term. As a distillation loss, the cross-correlation matrix is computed with the predicted and frozen features:

$$\mathcal{L}(z, \bar{z}) = \sum_u \left(1 - \bar{\mathcal{C}}_{uv}\right)^2 + \lambda \sum_u \sum_{v \neq u} \bar{\mathcal{C}}_{uv}^2, \tag{B.6}$$

where:

$$\bar{\mathcal{C}}_{uv} = \frac{\sum_i g(\boldsymbol{z}_{i,u}) \bar{\boldsymbol{z}}_{i,v}}{\sqrt{\sum_i g\left(\boldsymbol{z}_{i,u}\right)^2} \cdot \sqrt{\sum_i \left(\bar{\boldsymbol{z}}_{i,v}\right)^2}}. \tag{B.7}$$

## B.3 Further discussion and implementation details of the baselines

**Selection.** When evaluating our framework, we try to compare it with as many existing related methods as possible. However, given that SSL models are computationally intensive, it was not possible to run all baselines and methods in all the CL settings we considered. As mentioned in the main manuscript, we choose eight baselines (seven related methods + fine-tuning) belonging to three CL macro-categories and test them on CIFAR100 (class-incremental) in combination with three SSL methods. The selection was based on the ease of adaptation to CSSL and the similarity to our framework.

The most similar to CaSSLe are data-focused regularization methods. Among them, a large majority leverage knowledge distillation using the outputs of a classifier learned with supervision e.g. [99, 23, 58], while a few works employ feature distillation [79, 51] which is viable even without supervision. [81] is also related to CaSSLe, but it focuses on memory efficiency which is less interesting in our setting. Also, [81] explicitly uses the

classifier after feature adaptation, hence it is unclear how to adapt it for CSSL, especially since in SSL positives are generated using image augmentations, which are not applicable to a memory bank of features. On the contrary, augmentations can be used in replay methods, among which we select the most common (ER [136]) and one of the most recent (DER [15]). Regarding prior-focused regularization methods, we choose EWC [91] over others (SI [192], MAS [4], etc.) as it is considered the most influential and it works best with task boundaries. We also consider two CSSL baselines: LUMP [108] and Lin et. al [101]. Finally, we do not consider methods based on VAEs [133, 1], since they have been shown to yield poor performance on the large and medium scale. For instance, as found by [54], a VAE trained offline on CIFAR10 reaches an accuracy of 57.2%, which is lower than any method (except VICReg) trained continually on CIFAR100 with CaSSLe.

**Implementation.** For EWC, we use the SSL loss instead of the supervised loss to estimate importance weights. For POD and Less-Forget, we only re-implement the feature distillation without considering the parts of their methods that explicitly use the classifier. For DER, we replace the logits of the classifier with the projected features in the buffer. We re-implement all these baselines by adapting them from the official implementation (POD), or from the Mammoth framework provided with [15] (DER, ER, EWC), or from the paper (Less-Forget). We also compare with two concurrent works that propose approaches for CSSL (LUMP [108], Lin et al. [101]). LUMP uses k-NN evaluation, therefore we adapt the code provided by the authors to run in our code base. For Lin et al., we compare directly with their published results, since they use the same evaluation protocol. We perform hyperparameter tuning for all baselines, searching over 5 values for the distillation loss weights of POD and Less-Forget, 3 values for the

Table B.1: Linear evaluation top-1 accuracy on ImageNet100 (5 tasks, class- and data-incremental).

| **Method** | **Strategy** | **ImageNet100** | |
|---|---|---|---|
| | | Class-inc. | Data-inc. |
| Supervised | Fine-tuning | 61.6 | 74.3 |
| Contrastive | CaSSLe | **69.6** | **76.9** |

weight of the regularization in EWC and 3 replay batch sizes for replay methods. The size of the replay buffer is 500 samples for all replay-based methods.

## B.4 Additional results

**Continual supervised contrastive with CaSSLe.** After the popularization of contrastive learning [28, 72] for unsupervised learning of representations, [86] proposed a supervised version of the contrastive loss. Here, we show that CaSSLe is easily extendable to support supervised contrastive learning. The implementation is basically the same as for our vanilla contrastive-based distillation loss. In Tab. B.1, we show the improvement that CaSSLe brings with respect to fine-tuning, which is sizeable in the class-incremental setting. We also report the same comparison on DomainNet in Tab. B.2, showing interesting results in both task-aware and task-incremental evaluation.

**Task-agnostic evaluation and domain-wise accuracy on DomainNet.** In the main manuscript, we showed that CaSSLe significantly improved performance in the domain-incremental setting using task-aware evaluation. Here, "task-aware" refers to the fact that linear evaluation is performed on each

Table B.2: Linear evaluation top-1 accuracy on DomainNet (6 tasks, domain-incremental setting) w/ and w/o CaSSLe. The sequence of tasks is Real → Quickdraw → Painting → Sketch → Infograph → Clipart. "Aw." stands for task-aware, "Ag," for task-agnostic.

| Method | Strategy | Real | | Quickdraw | | Painting | | Sketch | | Infograph | | Clipart | | Avg. | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Aw. | Ag. | Aw. | Ag. | Aw. | Ag. | Aw. | Ag. | Aw. | Ag. | Aw. | Ag. | Aw. | Ag. |
| Barlow Twins | Finetuning | 56.3 | 50.9 | 54.1 | 45.8 | 42.7 | 35.9 | 49.0 | 41.9 | 22.0 | 17.4 | 59.0 | 52.5 | 50.3 | 43.7 |
| | CaSSLe | **62.7** | **57.1** | **59.1** | **50.6** | **49.2** | **42.1** | **53.8** | **47.7** | **25.5** | **20.6** | **61.9** | **55.6** | **55.5** | **48.9** |
| | Offline | 67.1 | 63.0 | 60.3 | 53.9 | 52.4 | 46.3 | 51.9 | 46.9 | 25.9 | 21.0 | 58.8 | 52.6 | 57.2 | 51.8 |
| SwAV | Finetuning | 57.7 | 52.3 | 53.2 | 43.5 | 43.0 | 35.9 | 46.1 | 39.0 | 21.6 | 16.5 | 53.4 | 46.6 | 49.6 | 42.5 |
| | CaSSLe | **62.8** | **57.8** | **59.5** | **50.2** | **47.5** | **41.2** | **49.5** | **42.5** | **22.5** | **17.9** | **56.5** | **49.6** | **54.3** | **47.5** |
| | Offline | 64.1 | 59.5 | 60.6 | 53.6 | 47.6 | 42.9 | 47.7 | 42.1 | 23.3 | 18.9 | 53.6 | 47.3 | 54.6 | 49.1 |
| BYOL | Finetuning | 58.7 | 53.2 | 51.7 | 41.6 | 44.0 | 37.4 | 49.6 | 43.9 | 23.5 | 19.0 | 58.6 | 53.5 | 50.6 | 43.8 |
| | CaSSLe | **63.7** | **60.5** | **59.3** | **50.9** | **48.6** | **44.1** | **50.4** | **45.2** | **24.1** | **19.4** | **59.0** | **54.4** | **55.1** | **49.7** |
| | Offline | 67.2 | 64.0 | 60.2 | 53.3 | 51.5 | 47.3 | 50.4 | 46.2 | 24.5 | 20.8 | 57.0 | 51.5 | 56.6 | 51.9 |
| VICReg | Finetuning | 54.7 | 49.6 | 53.0 | 44.9 | 42.1 | 34.7 | 49.0 | 41.9 | 21.1 | 16.4 | 58.5 | 52.6 | 49.3 | 42.8 |
| | CaSSLe | **59.0** | **53.2** | **56.4** | **47.8** | **46.0** | **38.9** | **52.3** | **45.6** | **23.9** | **18.5** | **60.9** | **55.3** | **52.9** | **46.1** |
| | Offline | 66.4 | 62.7 | 59.2 | 53.5 | 52.4 | 47.2 | 53.2 | 48.1 | 25.3 | 20.7 | 58.3 | 53.2 | 56.7 | 51.9 |
| SimCLR | Finetuning | 52.5 | 47.6 | 48.2 | 38.1 | 37.5 | 31.7 | 42.8 | 35.7 | 18.8 | 14.4 | 50.9 | 46.8 | 45.1 | 38.4 |
| | CaSSLe | **58.4** | **43.4** | **54.2** | **44.7** | **43.9** | **37.7** | **47.6** | **41.9** | **22.0** | **17.8** | **54.9** | **50.5** | **50.0** | **44.2** |
| | Offline | 62.1 | 59.5 | 58.3 | 52.9 | 46.1 | 42.5 | 45.6 | 41.3 | 22.1 | 18.8 | 51.0 | 45.9 | 52.6 | 48.6 |
| MoCoV2+ | Finetuning | 50.9 | 45.5 | 45.8 | 37.5 | 36.0 | 29.3 | 39.5 | 32.1 | 17.9 | 13.5 | 50.3 | **44.5** | 43.2 | 36.7 |
| | CaSSLe | **56.0** | **50.3** | **48.7** | **40.0** | **40.4** | **33.6** | **42.0** | **35.0** | **19.9** | **15.2** | **51.7** | **44.5** | **46.7** | **38.8** |
| | Offline | 65.2 | 61.3 | 57.9 | 51.3 | 48.7 | 43.1 | 44.7 | 39.1 | 23.4 | 19.0 | 51.3 | 44.8 | 53.7 | 48.4 |
| Supervised Contrastive | Finetuning | 57.7 | 52.6 | 55.3 | 45.5 | 44.9 | 38.0 | 51.7 | 45.0 | 22.6 | 18.3 | 64.0 | 60.0 | 52.1 | 45.4 |
| | CaSSLe | **63.4** | **58.8** | **59.7** | **51.3** | **50.1** | **44.7** | **55.9** | **50.3** | **26.9** | **22.4** | **65.0** | **61.3** | **56.7** | **50.9** |
| | Offline | 67.4 | 65.3 | 65.8 | 63.0 | 53.6 | 50.9 | 56.0 | 53.1 | 28.0 | 25.7 | 62.8 | 59.6 | 60.0 | 57.4 |
| Supervised | Finetuning | 63.0 | 58.2 | 56.9 | 47.6 | 49.1 | 44.0 | 55.7 | 50.3 | 27.7 | 23.3 | 68.6 | 63.5 | 55.9 | 49.8 |
| | Offline | 74.7 | 73.2 | 68.5 | 67.8 | 62.0 | 59.3 | 65.7 | 63.7 | 33.7 | 34.5 | 72.3 | 69.3 | 66.4 | 65.0 |

domain separately, i.e. a different linear classifier is learned for each domain. However, it might also be interesting to check the performance of the model when the domain is unknown at test time. For this reason, we report the performance of our model when evaluated in a task-agnostic fashion. In addition, we also show the accuracy of each task (i.e. domain). All this information is presented in Tab. B.2. CaSSLe **always** outperforms fine-tuning with both evaluation protocols. The accuracy of CaSSLe on "Clipart" is also higher than offline. This is probably due to a combination of factors: (i) Clipart is the last task, therefore it probably benefits in forward transfer and (ii) a similar effect to the one found in [160], where

Table B.3: k-NN evaluation on ImageNet100 (5 tasks, class-incremental) performed on backbone and projected features.

| Method | Strategy | k-NN accuracy (↑) | |
|---|---|---|---|
| | | Backbone ($f_b$) | Projector ($f_p$) |
| Barlow Twins | Fine-tuning | 59.1 | 34.4 |
| | CaSSLe | **63.4** | **53.2** |
| SwAV | Fine-tuning | **60.0** | 53.9 |
| | CaSSLe | 59.7 | **61.3** |
| BYOL | Fine-tuning | 57.1 | 33.0 |
| | CaSSLe | **61.2** | **60.8** |
| VICReg | Fine-tuning | 56.7 | 35.3 |
| | CaSSLe | **59.5** | **43.4** |
| MoCoV2+ | Fine-tuning | 54.5 | 39.0 |
| | CaSSLe | **61.5** | **53.1** |
| SimCLR | Fine-tuning | 54.8 | 40.1 |
| | CaSSLe | **61.7** | **53.2** |

dividing data in subgroups tends to enable the learning of better representations. Also, we notice that task-agnostic accuracy is lower than the task-aware counterpart. This is expected and means that the class conditional distributions are not perfectly aligned in different domains. As in the main paper, the colors are related to the type of SSL loss.

**Additional results with k-NN evaluation.** For completeness, in this supplementary material, we also show that CaSSLe yields superior performance when evaluated with a k-NN classifier instead of linear evaluation. We use weighted k-NN with l2-normalization (cosine similarity) and temperature

Table B.4: Linear evaluation top-1 accuracy on CIFAR100 (10 tasks, class-incremental).

| Method | Strategy | A (↑) |
|---|---|---|
| SimCLR | Fine-tuning | 39.3 |
| | CaSSLe | **52.7** |
| Barlow Twins | Fine-tuning | 49.9 |
| | CaSSLe | **53.7** |

scaling as in [21]. Since k-NN is much faster than linear evaluation we could also assess the quality of the projected representations, instead of just using the backbone. The results can be inspected in Tab. B.3. Three interesting phenomena arise: (i) CaSSLe always improves with respect to fine-tuning, (ii) the features of the backbone $f_b$ are usually better than the features of the projector $f_p$ and (iii) CaSSLe causes information retention in the projector, which significantly increases the performance of the projected features. An exception is represented by SwAV [19], which seems to behave differently to other methods. First, the accuracy of the projected features in SwAV is much higher than other methods. This might be due to the fact that it uses prototypes, which bring the representations 1 layer away from the loss, making them less specialized in the SSL task. Second, it seems that CaSSLe only improves the projected features when coupled with SwAV. However, this is probably an artifact of the evaluation procedure, as the l2-normalization probably causes loss of information. Indeed, although the overall performance is lower, SwAV + CaSSLe outperforms SwAV + fine-tuning (58.7% vs 56.9%) if the Euclidean distance is used in place of the cosine similarity for the backbone features. We leave a deeper investigation of this phenomenon for future work.

Table B.5: Linear evaluation top-1 accuracy on ImageNet100 (5 tasks, class- and data-incremental) with ResNet50 [74].

| **Method** | **Strategy** | **A (↑)** | |
|---|---|---|---|
| | | **Class-inc.** | **Data-inc.** |
| SimCLR | Fine-tuning | 70.7 | 75.6 |
| | CaSSLe | **74.0** | **77.2** |
| Barlow Twins | Fine-tuning | 71.2 | 75.8 |
| | CaSSLe | **74.8** | **78.1** |

**Different number of tasks.** The analysis of CSSL settings that we show in the main manuscript is limited to the 5 task scenario. However, it is interesting to run the same benchmarks with a longer task sequence. Nonetheless, one should also remember that SSL methods are data-hungry, hence the less data is available per task, the higher the instability of the SSL models. In Tab. B.4, we present additional results with 10 tasks on CIFAR100 (class-incremental). Barlow Twins seems to hold up surprisingly well, finishing up at roughly 50% accuracy, while SimCLR suffers in the low data regime. Nonetheless, CaSSLe outperforms fine-tuning with Barlow Twins, and to a very large extent with SimCLR.

**Deeper architectures.** The experiments we propose in the main manuscript feature a ResNet18 network. This is a common choice in CL. However, in SSL, it is more common to use ResNet50. For this reason, in Tab. B.5 we show that the same behavior observed with smaller networks is also obtained with deeper architectures. More specifically, CaSSLe outperforms fine-tuning in both class- and data-incremental settings by large margins.

Table B.6: Combinations of SSL methods and distillation losses on CIFAR100 (class-incremental, 2 tasks).

| Distillation Loss | SimCLR | Barlow Twins | BYOL |
|---|---|---|---|
| InfoNCE | **61.8** | 64.5 | 64.8 |
| Cross-correlation | 60.1 | **67.2** | 65.8 |
| MSE | 61.3 | 64.6 | **66.7** |

**The role of the predictor.** In the main manuscript, we provided an intuitive explanation of the role of the predictor network that maps the current feature space to the frozen feature space. This intuition is corroborated by extensive experimentation and ablation studies. However, one more thing that is worth mentioning is that the success of the predictor network might also be related to the findings in SimSiam [30], BYOL [69] and DirectPred [162]. Moreover, we perform additional ablations on the design of CaSSLe's predictor for SimCLR on CIFAR100 (5 tasks): adding BatchNorm after the hidden layer does not make any difference in terms of performance, and removing the non-linearity only causes a 0.3% drop in accuracy.

**Combinations of SSL methods and distillation losses.** For computational reasons, it was not feasible to perform experiments combining all SSL methods with all possible distillation losses. However, in Tab. B.6 we provide a subset of the possible combinations to validate our strategy that uses the same SSL loss for distillation.