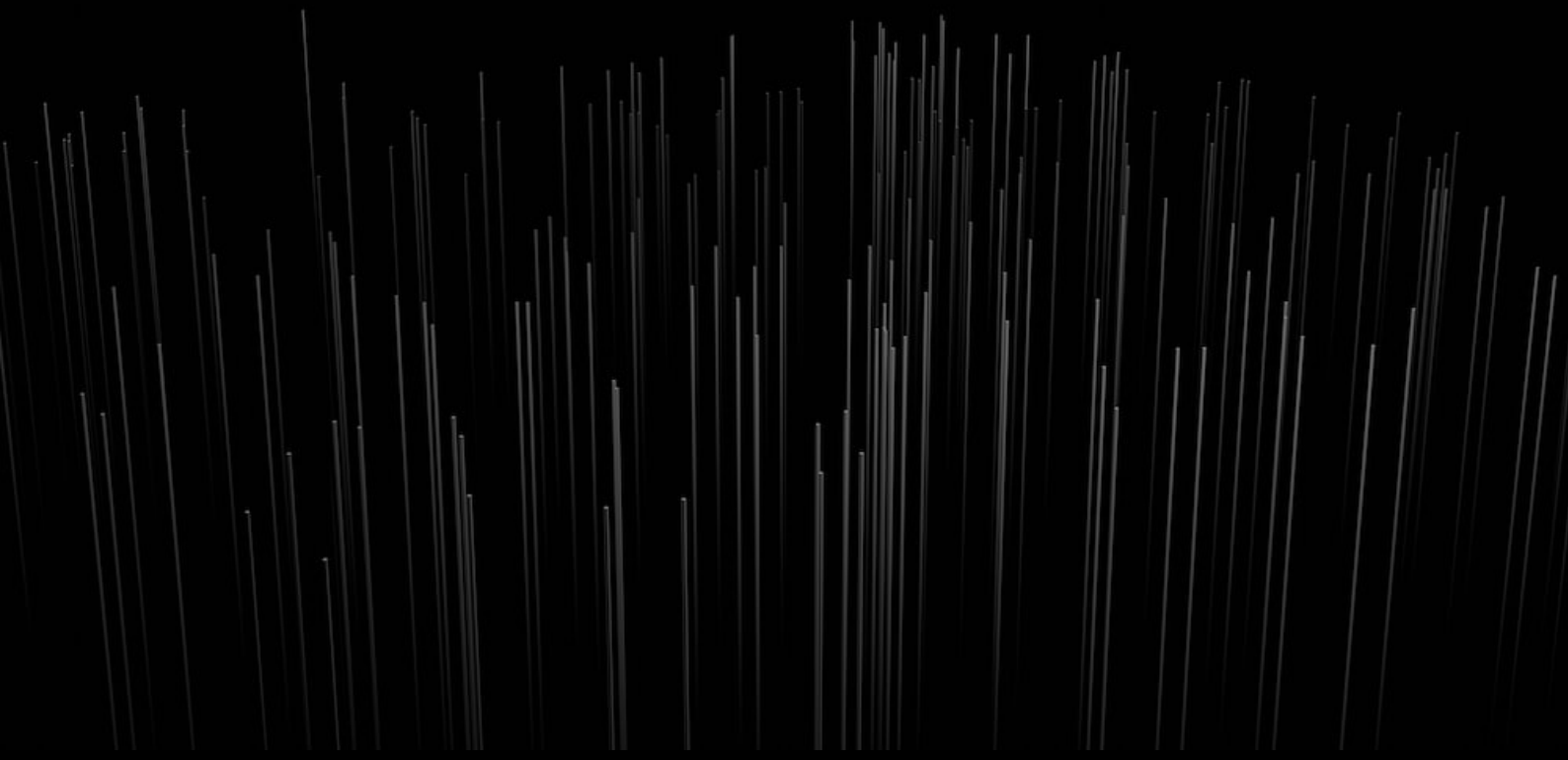
BECLR

Batch Enhanced Contrastive Unsupervised Few-Shot Learning

Stylianos Poulakakis-Daktylidis



BECLR

Batch Enhanced Contrastive Unsupervised Few-Shot Learning

by

Stylianos Poulakakis-Daktylidis

to obtain the degree of Master of Science

at the Delft University of Technology,

to be defended publicly on Tuesday September 5, 2023 at 10:00.

Student number:	5592615	
Project duration:	November 1, 2022 – August 31, 2023	
Thesis committee:	Dr. ir. H. Jamali-Rad,	TU Delft and Shell, Daily supervisor
	Prof. dr. ir. M. Reinders,	TU Delft, Advisor
	Dr. ir. H. Caesar,	TU Delft, External committee member

TU Delft

Preface

This report and scientific article: "BECLR: Batch Enhanced Contrastive Unsupervised Few-Shot Learning" presents the culmination of my work in the context of my Master's thesis project and marks the conclusion of my postgraduate studies at TU Delft. This research was conducted under the supervision of Dr. Marcel Reinders and the daily supervision of Dr. Hadi Jamali-Rad, within the Computer Vision Lab at TU Delft.

First, I would like to express my deepest appreciation for Dr. Hadi Jamali-Rad, who has not only been my daily supervisor but also a mentor figure for me in the exciting realm of research. I cannot thank you enough for investing so much of your time and effort in this thesis, as well as for your ongoing interest and support on every step of my path. Our, sometimes endless, discussions have not only imparted me with academic knowledge, but also invaluable life lessons. In addition, I would like to sincerely thank my thesis advisor, Dr. Marcel Reinders, for the trust he showed me in the assignment of this thesis and for the continuous support throughout our cooperation. I would also like to thank Dr. Holger Caesar for his time and interest in my work and for participating in my defense committee.

I would also like to take this opportunity to express my gratitude to the people closest to me: my parents, brothers, and girlfriend for constantly being there for me, supporting me in every possible way and loving me unconditionally. Finally, I want to warmly thank all my friends, fellow students and professors at TU Delft and all who stood by me throughout my postgraduate studies.

The past two years at Delft account for some of the most rewarding experiences and lessons in my life. I have been among some of the brightest people I know, met new friends, made countless memories, and further ignited my passion for research and computer science.

This report is structured in two main parts. The first part includes the scientific article on this research (BECLR) and contains the motivation, related work, developed methodology and modules, extensive evaluation, empirical results, and the concluding remarks of the current thesis. The second part discusses all the fundamental concepts and background information that have made this work a reality, in an attempt to make this report as self-contained as possible.

Stylianos Poulakakis-Daktylidis
Delft, September 2023

List of Publications

[1] Stylianos Poulakakis-Daktylidis, Ojas Kishore Shirekar, Hadi Jamali-Rad, “Batch Enhanced Contrastive Unsupervised Few-Shot Learning”, To be submitted to The 12th International Conference on Learning Representations (ICLR). 2024.

Contents

1 Introduction 1

2 Scientific Article (BECLR) 3

3 Deep Learning 23
- 3.1 Deep Feedforward Networks . . . 23
- 3.2 Activation Functions . . . 24
- 3.3 Optimisation and Backpropagation . . . 25
  - 3.3.1 Loss Function . . . 25
  - 3.3.2 Stochastic Gradient Descent . . . 25
  - 3.3.3 Backpropagation . . . 26

4 Convolutional Neural Networks 28
- 4.1 Convolution . . . 29
- 4.2 Pooling . . . 30
- 4.3 Feature Extraction with CNNs . . . 30
- 4.4 Deep Residual Networks . . . 31

5 Self-Supervised Learning 32
- 5.1 Contrastive SSL . . . 33
  - 5.1.1 SimCLR . . . 33
  - 5.1.2 NNCLR . . . 35
- 5.2 Self-Distillation SSL . . . 35
  - 5.2.1 BYOL . . . 36
  - 5.2.2 SimSiam . . . 37
- 5.3 Masked Image Modeling . . . 37
  - 5.3.1 Masked Siamese Networks . . . 39

6 Few-Shot Learning 41
- 6.1 Problem Formulation . . . 42
- 6.2 Model Agnostic Meta Learning . . . 43
- 6.3 Prototypical Networks . . . 44
- 6.4 Unsupervised Few-Shot Learning . . . 44
  - 6.4.1 Unsupervised Meta Learning . . . 45
  - 6.4.2 ProtoTransfer . . . 46
  - 6.4.3 PDA-Net . . . 47
  - 6.4.4 UniSiam . . . 49
  - 6.4.5 Connection to BECLR . . . 50

7 Optimal Transport 52
- 7.1 Continuous Optimal Transport . . . 53
- 7.2 Discrete Optimal Transport . . . 54
  - 7.2.1 Assignment Problem . . . 54
  - 7.2.2 Working with Asymmetric Distributions . . . 55
  - 7.2.3 The Kantorovich Relaxation . . . 55
  - 7.2.4 Entropic Regularization . . . 56
  - 7.2.5 Sinkhorn-Knopp Algorithm . . . 57

8 Conclusions and Future Directions 58

1

Introduction

Deep learning has been crucial in advancing major progress in various scientific fields, including computer vision, speech recognition, natural language processing, and more. However, despite these breakthroughs, deep learning models and artificial intelligence (AI) are becoming increasingly data hungry, requiring ever-growing amounts of training data and supervision to guarantee an acceptable performance and generalization ability on even the simplest of downstream tasks. In stark contrast, humans can quickly learn new skills much more efficiently with just a handful of data. Human intelligence allows for learning of new abstract concepts, detecting patterns and similarities between objects (analogous to how deep networks learn discriminative features), quick adaptation to novel tasks, or inferring much richer (than deep learning models) relationships and abstractions, with minimal supervision and "data" examples. For instance, people who know how to ride a bike could use this knowledge to learn to drive a motorcycle faster, with little demonstration. Additionally, humans from a very early age can associate similar objects by intuitively parsing major (discriminate) components of the object, e.g, children would be able to easily recognize cats and distinguish them from other animals after only encountering them a single or a few times. Notably, deep learning models are quite poor at inferring this type of abstractions.

Few-shot learning (FSL) aspires to bridge this *fundamental gap* between human and artificial intelligence, by learning in a data-deficient setting, and thus has recently gained an upsurge of interest. In particular, the few-shot model is trained to be able to generalize or adapt well to novel tasks and environments that have not been encountered during training. In practice, very limited data samples of the novel task configurations are used as part of a mini learning session for quickly adapting the model to the unseen test tasks. The fundamental principle of FSL lies in learning a prior, then used to solve unknown, downstream few-shot tasks. In the context of this work, we focus on the specific few-shot learning instance of **image classification**, where the downstream task involves the prediction of class labels for an unlabeled dataset (query set) based on a rather small labeled dataset (support set) with only a few samples (typically 1 to 5) per evaluated class. The query and support set are drawn from the same data distribution, and together comprise an *episode* or a few-shot *task*. Few-shot classification approaches typically consist of two sequential phases: *pretraining* on a large dataset of base classes, followed by a supervised *inference/fine-tuning* strategy on an unseen smaller dataset of novel classes. The problem of interest of this paper is the more challenging and realistic *unsupervised* few-shot learning (U-FSL), where we do not have access to the base class labels, and hence do not depend on an expensive annotated dataset. In this exciting space, the goal of the pretraining phase is to learn a feature extractor (i.e., backbone network or encoder) to capture the global structure of the unlabeled data, followed by fitting a (linear) classifier on top of the "frozen" feature extractor to quickly adapt to unseen but relevant downstream few-shot tasks. Ideally, the learned model (in the pretraining phase) should be able to complete new unseen tasks and quickly learn their underlying concepts and abstractions with minimal new samples (in the inference phase).

Various research approaches have been proposed to solve the challenging U-FSL problem from various perspectives. Earlier approaches (Hsu, Levine, and Finn 2018; Antoniou and Storkey 2019; Khodadadeh, Boloni, and Shah 2019; Khodadadeh, Zehtabian, et al. 2020) tackle U-FSL under the **meta-learning** paradigm, where synthetic learning tasks (or episodes), which mimic the downstream (also episodic) few-shot tasks, are used for pretraining the model. However, the complex meta-learning pretraining strategy has been shown to be data-inefficient (Dhillon et al. 2019; Yonglong Tian et al. 2020; Laenen and Bertinetto 2021), not fully utilizing information within a batch. Instead, more recent U-FSL approaches (Medina, Devos, and Grossglauser 2020; Z. Chen, Maji, and Learned-Miller 2021; L. Chen, K. Chen, and Chi-Guhn 2022; Kishorkumar Shirekar, A. Singh, and Jamali-Rad 2022; Jang, H. Lee, and Shin 2023) follow a simpler, non-episodic pretraining, based on **transfer learning**, for learning optimal representations. Exceptionally, state-of-the-art approaches (W. Chen et al. 2021; Lu et al. 2022; W. Hu et al. 2023) successfully employ some form of **contrastive learning** in their self-supervised pretraining and significantly outperform meta-learning methods. The underlying idea of contrastive representation learning (T. Chen et al. 2020; He, Fan, et al. 2020; Xinlei Chen and He 2021) is to attract similar samples (or *positives*) in the representation space while repelling dissimilar (or *negatives*) ones. Although fit for purpose, these contrastive approaches seem to overlook an important perspective: contrastive learning typically enforces consistency only at the *instance-level*, where each image within the batch (and its augmentations) corresponds to a unique class (unrealistic assumption!). As a result, potential positives (i.e., images of the same actual class), present within a batch, might be treated as negatives and pushed apart in the representation space.

To make matters worse, in the U-FSL setting, the (pretraining) base and (inference) novel classes are either mutually exclusive classes of the same dataset (*in-domain* setting) or originate from different datasets all together (*cross-domain* setting) - both of interest and investigated in this work. This *distribution shift* is usually addressed by supervised fine-tuning in conventional downstream task settings. However, in FSL problems, due to the limited number of support samples at test time, the support embeddings do not efficiently represent query characteristics, rendering fine-tuning inefficient and debilitating the downstream performance in the inference phase. This issue is often referred to as *sample bias* and is mostly ignored by U-FSL approaches, contrastive or otherwise.

Inspired by the efficiency of human intelligence in learning rich abstractions and semantic object relationships from just a few examples, we develop a novel end-to-end methodology: `BECLR`, which sets a new state-of-the-art in U-FSL settings. This body of work aims to learn rich representations, in a self-supervised manner, capable of capturing abstractions and relationships between images, and subsequently utilize those representations to tackle the few-shot learning problem. We build upon contrastive learning (which have been shown to be front-runners in U-FSL) and propose a novel batch enhanced contrastive U-FSL pretraining methodology (coined as `BECLR`) to infuse *instance-* and *class-level* insights within a contrastive learning framework. To enable the sampling of meaningful positives, within the contrastive loss of `BECLR`, we introduce an innovative dynamic clustered memory module (`DyCE`), which maintains *highly-separable latent space partitions*, through iterative equipartitioned batch updates. We also propose an effective, optimal transport (OT)-based feature alignment strategy (`OpTA`), to structurally address *sample bias* in the U-FSL inference stage and further improve the end-to-end performance of `BECLR` in low-shot settings.

2

Scientific Article (BECLR)

BATCH ENHANCED CONTRASTIVE UNSUPERVISED FEW-SHOT LEARNING

Stylianos Poulakakis-Daktylidis[1], **Ojas Kishorkumar Shirekar**[1,2] **& Hadi Jamali-Rad**[1,2]
[1]Delft University of Technology (TU Delft), The Netherlands
[2]Shell Global Solutions International B.V., Amsterdam, The Netherlands
s.poulakakisdaktylidis@student.tudelft.nl
{o.k.shirekar, h.jamalirad}@tudelft.nl

ABSTRACT

There exists a fundamental gap between human and artificial intelligence. Deep learning models are exceedingly data hungry for learning even the simplest of tasks, whereas humans can easily adapt to new tasks with just a handful of samples. Unsupervised few-shot learning (U-FSL) aspires to bridge this gap, without relying on costly annotations. Inspired by the efficiency of contrastive representation learning, we propose a novel batch enhanced contrastive U-FSL pretraining methodology (coined as BECLR) to infuse *instance-* and *class-level* insights within a contrastive framework. To enable the sampling of meaningful positives, we introduce an innovative dynamic clustered memory module (DyCE), which maintains *highly-separable latent space partitions*, through iterative equipartitioned updates. We also propose an effective, optimal transport (OT)-based feature alignment strategy (OpTA), to address *sample bias* in the U-FSL inference stage and further boost the end-to-end performance of BECLR in low-shot settings. Our extensive experimental evaluation corroborates the efficacy of our design choices in BECLR, which sets a new *state-of-the-art* on the most widely adopted U-FSL benchmarks miniImageNet and tieredImageNet (offering up to 14% and 12% improvements, respectively), as well as on challenging cross-domain scenarios.[1]

1 INTRODUCTION

Deep learning models are becoming increasingly data hungry, requiring ever-growing amounts of training data to guarantee an acceptable performance and generalization ability on even the simplest of downstream tasks. In stark contrast, humans can quickly learn new skills from a handful of data, without extensive supervision. Few-shot learning (FSL) aspires to bridge this fundamental gap between human and artificial intelligence, by learning in a data-deficient setting, and thus has recently gained an upsurge of interest. The fundamental principle of FSL lies in learning a prior, then used to solve unknown, downstream few-shot tasks. To this end, FSL approaches typically consist of two sequential phases: pretraining on a large dataset of base classes, followed by a *supervised* inference/fine-tuning strategy on an unseen smaller dataset of novel classes. The problem of interest of this paper is the more challenging and realistic *unsupervised* few-shot learning (U-FSL), where we do not have access to the base class labels,

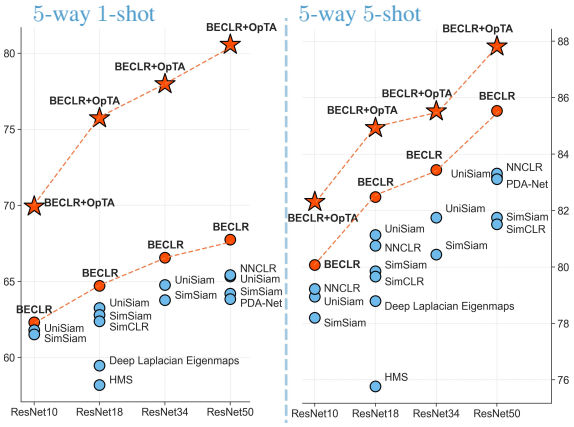


Figure 1: MiniImageNet (5-way, 1-shot) (left) and (5-way, 5-shot) (right) accuracy in the unsupervised few-shot learning landscape. Our method, BECLR, is shown in bold.

[1]Codebase and models are publicly available at https://github.com/stypoumic/BECLR.

and hence do not depend on an expensive annotated dataset. In this exciting space, the goal of the pretraining phase is to learn a feature extractor (i.e., backbone network or encoder) to capture the global structure of the unlabeled data, followed by fitting a (linear) classifier on top of the "frozen" feature extractor to quickly adapt to unseen but relevant downstream few-shot tasks.

Earlier approaches (Hsu et al., 2018; Khodadadeh et al., 2019; 2020) tackle U-FSL under the *meta-learning* paradigm, where synthetic learning tasks, which mimic the downstream few-shot tasks, are used for pretraining the model. However, the complex meta-learning pretraining strategy has been shown to be data-inefficient (Dhillon et al., 2019; Tian et al., 2020). Instead, more recent U-FSL approaches (Medina et al., 2020; Chen et al., 2021b; 2022; Jang et al., 2023) follow a simpler, non-episodic pretraining, based on *transfer learning*, for learning optimal representations. Exceptionally, state-of-the-art approaches (Chen et al., 2021a; Lu et al., 2022) successfully employ *contrastive learning* in their self-supervised pretraining and significantly outperform meta-learning methods. The underlying idea of contrastive representation learning (Chen et al., 2020a; He et al., 2020) is to attract similar samples in the representation space while repelling dissimilar ones. Although fit for purpose, these contrastive approaches seem to overlook an important perspective: contrastive learning typically enforces consistency only at the *instance-level*, where each image within the batch (and its augmentations) corresponds to a unique class (unrealistic assumption!). As a result, potential positives (i.e., images of the same actual class), present within a batch, might be pushed apart in the representation space. To address this problem, we argue that *membership/class-level* insights should be infused within the contrastive paradigm. Building upon this idea, we propose our `B`atch-`E`nhanced `C`ontrastive `LeaR`ning (coined as `BECLR`) self-supervised pretraining methodology and introduce a novel `Dy`namic `C`lustered m`E`mory (`DyCE`), which maintains a partitioned latent space of past representations. In other words, we propose to enhance the positive sampling strategy within a contrastive learning framework with a clustering-inspired perspective engraved through `DyCE`.

In U-FSL, the base and novel classes are either mutually exclusive classes of the same dataset (*in-domain* setting) or originate from different datasets (*cross-domain* setting) - both of interest and investigated in this paper. This *distribution shift* is usually addressed by supervised fine-tuning. However, due to the limited number of support samples at test time, the support embeddings do not efficiently represent query characteristics, debilitating the performance in the fine-tuning/inference phase. This issue is often referred to as *sample bias* and is mostly ignored by U-FSL approaches. To structurally address sample bias in U-FSL, we propose a simple add-on supervised inference strategy and introduce `Op`timal `T`ransport-based feature `A`lignment (`OpTA`), to align the representations of the "biased" support and "unbiased" query sets. Our main contributions can be summarized as:

1. We propose `BECLR`, a self-supervised contrastive pretraining methodology that integrates *instance-* and *class-level* cognizance within a contrastive framework, by *enhancing* the original batch with additional meaningful positives from past representations.
2. We introduce a novel dynamic clustered memory, `DyCE`, which is iteratively updated with *equipartitioned* batch assignments, converging into a *highly-separable* partitioned latent space and enabling the positive sampling strategy within `BECLR`.
3. We present `OpTA`, an effective (especially in lower-shot settings) optimal transport-based strategy, for addressing sample bias and inducing task awareness at the inference stage.
4. We perform extensive experimental evaluations to demonstrate that `BECLR` sets a new state-of-the-art performance, outperforming all existing U-FSL baselines on miniImagenet (see Fig. 1) and tieredImagenet (up to 14% and 12% improvement in the demanding 1-shot setting), as well as reporting competitive performance in the cross-domain setting.

2 Related Work

Self-Supervised Learning. Self-supervised learning (SSL) aims to obtain supervisory signals from within the data itself, without relying on expensive labels or annotations. SSL methods are able to learn robust representations in an unsupervised manner by defining pretext tasks, such as colorization (Larsson et al., 2016), rotation prediction (Gidaris et al., 2018), inpainting (Pathak et al., 2016), or solving jigsaw puzzles (Noroozi & Favaro, 2016). One of the most competitive SSL approaches is contrastive learning, which solves the *instance discrimination* pretext task (Chen et al., 2020a). Seminal contrastive approaches have pushed the state-of-the-art in U-FSL by introducing the popular

InfoNCE loss (Oord et al., 2018), self-distillation (or momentum) encoder schemes (Grill et al., 2020), an asymmetrical architecture (Chen & He, 2021), or encouraging channel-level consistency (Zbontar et al., 2021). BECLR builds on these ideas and tackles the challenging environment of unsupervised few-shot learning (U-FSL) within a contrastive learning framework.

Unsupervised Few-Shot Learning. U-FSL aims at pretraining a model from an unlabeled dataset of base classes, which is able to quickly generalize to novel tasks only with a few labeled examples. Meta-learning approaches (Hsu et al., 2018; Antoniou & Storkey, 2019; Khodadadeh et al., 2019; 2020) tackle U-FSL, by generating synthetic learning tasks (or episodes), which mimic the downstream, also episodic, few-shot tasks, for pretraining the model. Nevertheless, the complex pretraining strategy of the meta-learning paradigm has recently been shown to be data-inefficient, not fully utilizing information within a batch (Dhillon et al., 2019; Tian et al., 2020; Laenen & Bertinetto, 2021). Instead, state-of-the-art U-FSL approaches (Chen et al., 2021a; Lu et al., 2022; Chen et al., 2022; Shirekar et al., 2022b) follow a simpler non-episodic pretraining, based on transfer learning and contrastive SSL, outperforming meta-learning approaches by a significant margin. Despite their worthy attempts, current contrastive U-FSL approaches only enforce consistency at the *instance-level*, where each image within the batch corresponds to a unique class (unrealistic assumption!), and mostly ignore the *sample bias* problem in the inference stage of U-FSL. BECLR also builds on the contrastive pretraining setting, but infuses membership/class-level insights through our novel DyCE memory module, showing that it is possible to build an effective few-shot learner, without any base class labels. Finally, we also propose a simple add-on strategy, based on optimal transport, to structurally address sample bias and further improve downstream performance of BECLR.

3 Preliminaries

3.1 Problem Statement: Unsupervised Few-Shot Learning

The problem of interest of this paper is U-FSL. We follow the setup most commonly adopted in the literature (Chen et al., 2021a;b; Lu et al., 2022; Jang et al., 2023), which consists of: (i) *unsupervised* pretraining, followed by (ii) a *supervised* fine-tuning/inference strategy. Formally, we consider a large unlabeled base dataset $\mathcal{D}_{\text{tr}} = \{\boldsymbol{x}_i\}$ for pretraining our model. The fine-tuning/inference phase then involves transferring the model to unseen few-shot downstream tasks $\mathcal{T}_i$, drawn from a smaller labeled test dataset of novel classes $\mathcal{D}_{\text{tst}} = \{(\boldsymbol{x}_i, y_i)\}$. Each task $\mathcal{T}_i$ is composed of two parts $[\mathcal{S}, \mathcal{Q}]$: (i) the support set $\mathcal{S}$, from which the model learns to adapt to the novel classes, and (ii) the query set $\mathcal{Q}$, on which the model is evaluated. The support set $\mathcal{S} = \{\boldsymbol{x}_i^s, y_i^s\}_{i=1}^{NK}$ is constructed by drawing K labeled random samples from N different classes, resulting in the so-called (N-way, K-shot) settings. The query set $\mathcal{Q} = \{\boldsymbol{x}_j^q\}_{j=1}^{NQ}$ typically contains NQ unlabeled samples.

3.2 Contrastive Unsupervised Representation Learning

The underlying idea of contrastive representation learning (Wu et al., 2018; Chen et al., 2020a; He et al., 2020; Chen & He, 2021) is to attract "positive" samples in the representation space (i.e., representations of the augmentations of the same image), while repelling "negative" samples (i.e., representations of different images). A commonly adopted loss function to drive contrastive learning is infoNCE (Oord et al., 2018).

$$\mathcal{L}_{iNCE} = -\frac{1}{2B}\sum_{i=1}^{2B}\log\frac{\exp\left(\mathtt{d}[\boldsymbol{z}_i,\ \boldsymbol{z}_i^+]/\tau\right)}{\sum_{j\neq i}\exp\left(\mathtt{d}\left[\boldsymbol{z}_i,\ \boldsymbol{z}_j\right]/\tau\right)} = -\frac{1}{B}\sum_{i=1}^{B}\mathtt{d}[\boldsymbol{z}_i,\ \boldsymbol{z}_i^+]/\tau + \frac{1}{2B}\sum_{i=1}^{2B}\log\sum_{j\neq i}\exp\left(\mathtt{d}\left[\boldsymbol{z}_i,\ \boldsymbol{z}_j\right]/\tau\right), \quad (1)$$

where τ is the temperature parameter, $\mathtt{d}$ is a distance metric (negative cosine similarity in our case), B denotes the batch size, and $\boldsymbol{z}_i^+$ stands for the latent embedding of the positive sample, corresponding to sample i. The first term in Eq. 1 operates only on positive pairs, and the second term pushes each representation away from all other batch representations and has a regularization effect.

4 Proposed Method

In this section, we first introduce our proposed self-supervised pretraining methodology (coined as BECLR) including a novel Dynamic Clustered mEmory module (DyCE). Next, we discuss our

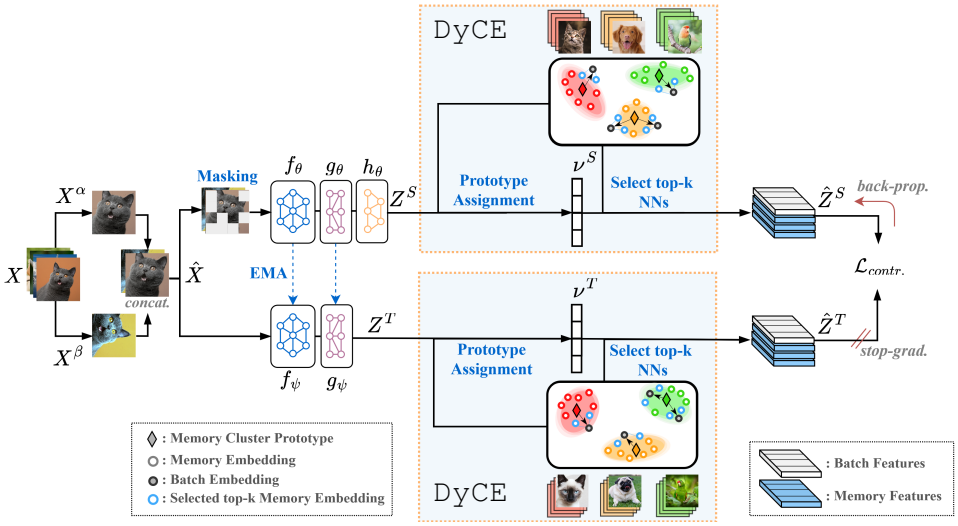


Figure 2: Overview of BECLR contrastive framework. Given two views $\boldsymbol{X}^{\{\alpha,\beta\}}$, both views are passed through a student $h_\theta \circ g_\theta \circ f_\theta \circ \mu$ and a teacher $g_\psi \circ f_\psi$ network and next, to our DyCE memory module. DyCE is responsible for (i) *enhancing* the original batch with meaningful positives and (ii) *dynamically updating* the memory partitions with equipartitioned assignments. A contrastive loss is applied on the enhanced batch.

simple yet efficient supervised inference strategy (referred to as OpTA), to induce downstream task awareness and alleviate sample bias in the U-FSL setting.

4.1 Self-Supervised Pretraining (BECLR)

Inspired by the recent success of contrastive representation learning frameworks in reaching state-of-the-art performance in the U-FSL setting (Chen et al., 2021a; Lu et al., 2022; Hu et al., 2023), we also build our proposed solution upon contrastive learning and solve the instance discrimination pretext task. The challenge with traditional contrastive learning approaches is that they enforce consistency only at the *instance-level*, where each image within the batch has to correspond to a unique class (unrealistic assumption!). As a result, potential positives, present within a batch, might be treated as negatives, which can have a detrimental impact on performance. A common strategy to combat this pitfall (also to avoid prohibitively large batch sizes) is to use a memory unit (Wu et al., 2018; Zhuang et al., 2019; He et al., 2020; Caron et al., 2020; Dwibedi et al., 2021). Exceptionally, NNCLR (Dwibedi et al., 2021) uses a nearest neighbor approach in sampling from the memory to promote more meaningful positive pair generation in the contrastive loss landscape. However, NNCLR uses a simplistic first-in-first-out memory queue and is oblivious to global memberships (i.e., class-level information) in the latent space. On the other hand, clustering-based SwAV (Caron et al., 2020) deviates from traditional contrastive learning and promotes soft clustering of label assignments. Instead, we build on the contrastive learning setting (which have been shown to be frontrunners in U-FSL) and propose to infuse membership/class-level insights through a novel Dynamic Clustered mEmory (DyCE) within our proposed Batch-Enhanced Contrastive LeaRning (BECLR) setting. Notably, neither NNCLR nor SwAV are devised for the U-FSL setting. Even so, we demonstrate later on that we outperform their reproduction by a margin in the U-FSL setting. In other words, we propose to enhance the positive sampling strategy within a contrastive learning framework with a clustering-inspired perspective engraved through DyCE. Figs. 2 and 3 provide a schematic illustration of our framework (BECLR) and its dynamic memory module (DyCE).

Batch Enhanced Contrastive Learning (BECLR). Let $\zeta^a, \zeta^b \sim \mathcal{A}$ be two randomly sampled data augmentations from the set of all available augmentations, $\mathcal{A}$. The current mini-batch can then be denoted as $\hat{\boldsymbol{X}} = [\hat{\boldsymbol{x}}_i]_{i=1}^{2B} = \left[[\zeta^a(\boldsymbol{x}_i)]_{i=1}^{B}, [\zeta^b(\boldsymbol{x}_i)]_{i=1}^{B}\right]$. As shown in Fig. 2, we adopt an enhanced student-teacher (a.k.a. Siamese) asymmetric momentum architecture similar to Grill et al. (2020); Chen & He (2021); Lu et al. (2022). The student $\boldsymbol{Z}^S$ (of size $2B \times d$, with d the latent embedding dimension) and teacher $\boldsymbol{Z}^T$ (same size as $\boldsymbol{Z}^S$) representations are obtained as follows:

$$\boldsymbol{Z}^S = h_\theta \circ g_\theta \circ f_\theta\big(\mu(\hat{\boldsymbol{X}})\big), \qquad \boldsymbol{Z}^T = g_\psi \circ f_\psi(\hat{\boldsymbol{X}}), \tag{2}$$

where $\mu(\cdot)$ is a patch-wise masking operator as in Assran et al. (2022), $f(\cdot)$ is the backbone feature extractor (ResNet (He et al., 2016) in our case), $g(\cdot)$ and $h(\cdot)$ are projection and prediction multi-layer perceptrons (MLPs), respectively. The teacher parameters ψ are an exponential moving averaged[2] (*EMA*) version of the student parameters θ (updated through SGD). Upon extracting $\boldsymbol{Z}^S$ and $\boldsymbol{Z}^T$, they are fed into our novel dynamic memory module (DyCE), where enhanced versions of the batch representations $\hat{\boldsymbol{Z}}^S$, $\hat{\boldsymbol{Z}}^T$ (both of size $2B(k+1) \times d$, with k denoting the number of selected nearest neighbors) are generated. Finally, we apply the contrastive loss in Eq. 5 on the enhanced batch representations $\hat{\boldsymbol{Z}}^{\boldsymbol{S}}$, $\hat{\boldsymbol{Z}}^{\boldsymbol{T}}$. Upon finishing unsupervised pretraining, only the student encoder (f_θ) is kept for the subsequent inference stage. BECLR is summarized in Algorithm 1, and a Pytorch-like pseudo-code can be found in Appendix D.

Algorithm 1: BECLR

Require: $\mathcal{A}$, θ, ψ, f_θ, f_ψ, g_θ, g_ψ, h_θ, μ, DyCE
$\hat{\boldsymbol{X}} = [\zeta^\alpha(\boldsymbol{X}), \zeta^\beta(\boldsymbol{X})]$ for $\zeta^\alpha, \zeta^\beta \sim \mathcal{A}$
$\boldsymbol{Z}^S = h_\theta \circ g_\theta \circ f_\theta(\mu(\hat{\boldsymbol{X}}))$
$\boldsymbol{Z}^T = g_\psi \circ f_\psi(\hat{\boldsymbol{X}})$.detach()
$\hat{\boldsymbol{Z}}^S,\ \hat{\boldsymbol{Z}}^T = \texttt{DyCE}(\boldsymbol{Z}^S),\ \texttt{DyCE}(\boldsymbol{Z}^T)$
Compute loss: $\mathcal{L}_{contr.}$ using Eq. 5 on $\hat{\boldsymbol{Z}}^S$, $\hat{\boldsymbol{Z}}^T$
Return: $\mathcal{L}_{contr.}$

Dynamic Clustered Memory (DyCE). *How do we manage to enhance our batch with meaningful true positives from a memory queue of unlabeled samples?* As we illustrate in Fig. 6, the efficiency of BECLR relies on establishing separable clusters within the memory (for both student and teacher branches), ideally each corresponding to a different class. We introduce DyCE: a dynamically updated clustered memory with equipartitioned batches to efficiently repel the memory prototypes, moderating the representation space during training. We demonstrate later on in Section 5 that our design choices in DyCE have a significant impact on both pretraining performance as well as the downstream few-shot classification.

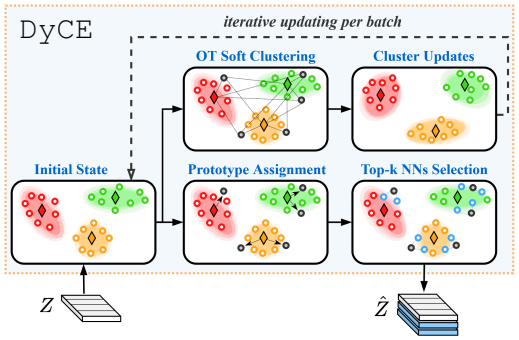


Figure 3: Overview of our dynamic clustered memory module (DyCE) and its two informational paths.

Algorithm 2: DyCE

Require: $\text{epoch}_{\text{thr}}$, $\mathcal{M}$, $\boldsymbol{\Gamma}$, $\boldsymbol{Z}$, B, k
1 **if** $|\mathcal{M}| = M$ **then**
2 // **Path (i)**: top-k and batch enhancement)
3 **if** epoch $\geq \text{epoch}_{\text{thr}}$ **then**
4 $\boldsymbol{\nu} = [\nu_i]_{i=1}^{2B} = \left[\underset{j \in [N]}{\texttt{argmin}} \langle \boldsymbol{z}_i, \boldsymbol{\gamma}_j \rangle\right]_{i=1}^{2B}$
5 $\boldsymbol{Y}_i \leftarrow \texttt{top-k}(\{\langle \boldsymbol{z}_i, \mathcal{P}_{\nu_i} \rangle\}), \forall i \in [2B]$
6 $\hat{\boldsymbol{Z}} = [\boldsymbol{Z}, \boldsymbol{Y}_1, \ldots, \boldsymbol{Y}_{2B}]$
7 // **Path (ii)**: iterative memory updating
8 Find OT plan between $\boldsymbol{Z}$ and $\boldsymbol{\Gamma}$: $\boldsymbol{\pi}^* \leftarrow$ Solve Eq. 4
9 Update $\mathcal{M}$ with new $\boldsymbol{Z}$: $\mathcal{M} \leftarrow \texttt{update}(\mathcal{M}, \boldsymbol{\pi}^*, \boldsymbol{Z})$
10 Discard $2B$ oldest batch embeddings: $\texttt{dequeue}(\mathcal{M})$
11 **else**
12 Store new batch: $\mathcal{M} \leftarrow \texttt{store}(\mathcal{M}, \boldsymbol{Z})$
Return: $\hat{\boldsymbol{Z}}$

Let us consider a memory unit $\mathcal{M}$ capable of storing a maximum of M latent embeddings (each of size d). To accommodate clustered memberships within DyCE, we consider up to P partitions in $\mathcal{M} = [\mathcal{P}_1, \ldots, \mathcal{P}_P]$, each of which is represented by a prototype stored in $\boldsymbol{\Gamma} = [\boldsymbol{\gamma}_1, \ldots, \boldsymbol{\gamma}_P]$. In practice, prototypes $\boldsymbol{\gamma}_i$ are the average of the latent embeddings stored in partition $\mathcal{P}_i$. As shown in Fig. 3 and in Algorithm 2, DyCE consists of two informational paths (i) the top-k neighbor selection and batch enhancement path (bottom branch of the figure), which uses the current state of $\mathcal{M}$ and $\boldsymbol{\Gamma}$, and (ii) the iterative memory updating via dynamic clustering path (top branch of the figure). DyCE takes student or teacher embeddings (we simply use $\boldsymbol{Z}$, for brevity) as input and returns the enhanced versions $\hat{\boldsymbol{Z}}$. Let us now walk you through the algorithm. Path (i) starts with assigning each $\boldsymbol{z}_i \in \boldsymbol{Z}$ to its nearest (based on the Euclidean distance $\langle \cdot \rangle$) neighbor prototype $\boldsymbol{\gamma}_{\nu_i}$, with the assignments being stored in vector $\boldsymbol{\nu}$ of size $2B$ (line 4, Algorithm 2). Next (in line 5), we select the top-k, most similar memory embeddings to $\boldsymbol{z}_i$ from the memory partition $\mathcal{P}_{\nu_i}$ and store them in $\boldsymbol{Y}_i$ (of size $k \times d$). Finally (in line 6), all $\boldsymbol{Y}_i, \forall i \in [2B]$ are concatenated in $\hat{\boldsymbol{Z}} = [\boldsymbol{Z}, \boldsymbol{Y}_1, \ldots, \boldsymbol{Y}_{2B}]$ of size $L \times d$ ($L = 2B(k+1)$).

We next follow up with iterative memory updates on path (ii). This is based on optimal transport (OT) (Cuturi, 2013) for enforcing an equipartitioning constraint to find a transport plan $\boldsymbol{\pi}$ mapping $\boldsymbol{Z}$ to $\boldsymbol{\Gamma}$ (in line 8) by solving:

$$\boldsymbol{\Pi}(\boldsymbol{r}, \boldsymbol{c}) = \left\{ \boldsymbol{\pi} \in \mathbb{R}_+^{2B \times P} \mid \boldsymbol{\pi} \mathbf{1}_P = \boldsymbol{r},\ \boldsymbol{\pi}^\top \mathbf{1}_{2B} = \boldsymbol{c} \right\}, \tag{3}$$

[2] $\psi \leftarrow m\psi + (1-m)\theta$, as in Grill et al. (2020), where m the momentum hyperparameter.

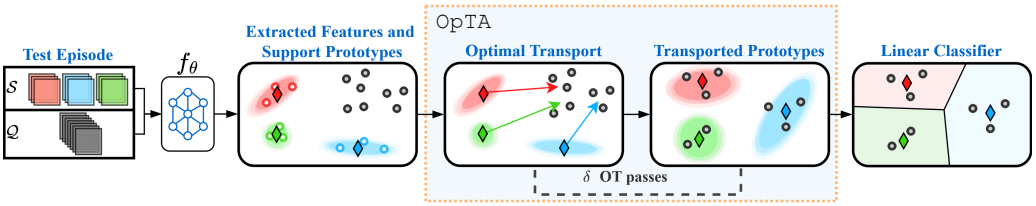


Figure 4: Overview of our inference strategy. Given a test episode, the support ($\mathcal{S}$) and query ($\mathcal{Q}$) sets are passed to the pretrained feature extractor (f_θ). Next, OpTA aligns support prototypes and query features.

with the added constraints $\boldsymbol{r} = \mathbf{1} \cdot {}^1\!/_{2B}$ and $\boldsymbol{c} = \mathbf{1} \cdot {}^1\!/_{P}$ for enforcing equipartitioning (i.e., uniform assignments), where $\boldsymbol{r} \in \mathbb{R}^{2B}$ denotes the distribution of batch embeddings $[\boldsymbol{z}_i]_{i=1}^{2B}$ and $\boldsymbol{c} \in \mathbb{R}^{P}$ the distribution of memory cluster prototypes $[\boldsymbol{\gamma}_i]_{i=1}^{P}$. A relaxed version of OT then finds the *optimal* transport plan (or assignment) $\boldsymbol{\pi}^\star$, which maximizes the overlap between the two distributions $(\boldsymbol{r}, \boldsymbol{c})$ as follows:

$$\boldsymbol{\pi}^\star = \underset{\boldsymbol{\pi} \in \boldsymbol{\Pi}(\boldsymbol{r},\boldsymbol{c})}{\operatorname{argmin}} \langle \boldsymbol{\pi}, \boldsymbol{D} \rangle_F - \varepsilon \mathbb{H}(\boldsymbol{\pi}), \tag{4}$$

where $\boldsymbol{D}$ is a pairwise distance matrix between the elements of $\boldsymbol{Z}$ and $\boldsymbol{\Gamma}$ (of size $2B \times P$), $\langle \cdot \rangle_F$ denotes the Frobenius dot product, ε controls the strength of the entropic regularisation, and $\mathbb{H}(\boldsymbol{\pi}) = \sum_{ij} \pi_{ij} \log(\pi_{ij})$ is the Shannon entropy. This is solved using the Sinkhorn-Knopp algorithm (Cuturi, 2013). Next (in line 9), we add the latest embeddings $\boldsymbol{Z}$ to $\mathcal{M}$ and use $\boldsymbol{\pi}^\star$ for updating the partitions $\mathcal{P}_i$ and prototypes $\boldsymbol{\Gamma}$ (EMA between the current and previous prototype values). Finally, we discard the $2B$ oldest memory embeddings (line 10). Note that at the beginning of training both the encoder representations and the memory embedding space are highly volatile. Thus, we allow for an adaptation period $\text{epoch} < \text{epoch}_{\texttt{thr}}$ (empirically $\text{epoch}_{\texttt{thr}} = 20\text{-}50$), during which the batch enhancement path (i) of DyCE is not activated. On the contrary, the memory updating path (ii) is activated for every training batch from the beginning of training, allowing our memory to reach a highly-separable converged state, as empirically shown in Fig. 6.

Loss Function. Despite the popularity of infoNCE (Eq. 1) recent studies (Poole et al., 2019; Song & Ermon, 2019) have shown that it is prone to high bias, especially when the batch size is small and the mutual information (MI) is large. We adopt a variant of InfoNCE, which maximizes the same MI objective, but has been shown to be less biased for edge cases (Lu et al., 2022):

$$\mathcal{L}_{contr.} = -\frac{1}{L} \sum_{i=1}^{L/2} \left(\mathtt{d}[\boldsymbol{z}_i^S, \boldsymbol{z}_i^{T+}] + \mathtt{d}[\boldsymbol{z}_i^{S+}, \boldsymbol{z}_i^T] \right) + \lambda \log \left(\frac{1}{L} \sum_{i=1}^{L} \sum_{j \neq i, i^+} \exp(\mathtt{d}[\boldsymbol{z}_i^S \cdot \boldsymbol{z}_j^S / \tau]) \right), \tag{5}$$

where $\mathtt{d}$ is the negative cosine similarity, λ is a weighting hyperparameter and $L = 2B(k+1)$. Our loss is asymmetric (i.e., both views are passed to both student and teacher), which has been shown to increase performance (Chen & He, 2021), when combined with an asymmetric architecture.

4.2 Supervised Inference

Supervised fine-tuning usually combats the distribution shift between training and test datasets. However, the limited number of samples at test time (especially in low-shot scenarios in FSL settings) leads to a significant performance degradation due to the so-called *sample bias*. This issue is mostly ignored in recent state-of-the-art U-FSL baselines (Chen et al., 2021a; Lu et al., 2022; Hu et al., 2023). As is customary in U-FSL, we evaluate our downstream performance on few-shot tasks, as such, we propose a simple add-on inference strategy, based on Optimal Transport feature Alignment (OpTA), to structurally address sample bias (distribution shift between query and support sets) contributing to the overall significant performance margin BECLR offers (see Section 5), especially in low-shot regimes.

Optimal Transport Feature Alignment (OpTA). Let $\mathcal{T} = \mathcal{S} \cup \mathcal{Q}$ be a downstream few-shot task. We first extract the support $\boldsymbol{Z}^{\mathcal{S}} = f_\theta(\mathcal{S})$ (of size $NK \times d$) and query $\boldsymbol{Z}^{\mathcal{Q}} = f_\theta(\mathcal{Q})$ (of size $NQ \times d$) embeddings and calculate the support set prototypes $\boldsymbol{P}^{\mathcal{S}}$ (class averages of size $N \times d$). Next, we find the optimal transport plan ($\boldsymbol{\pi}^\star$) between $\boldsymbol{P}^{\mathcal{S}}$ and $\boldsymbol{Z}^{\mathcal{Q}}$ using Eqs. 3 and 4, with $\boldsymbol{r} \in \mathbb{R}^{NQ}$ the distribution of $\boldsymbol{Z}^{\mathcal{Q}}$ and $\boldsymbol{c} \in \mathbb{R}^{N}$ the distribution of $\boldsymbol{P}^{\mathcal{S}}$. Finally, we use $\boldsymbol{\pi}^\star$ to create a barycentric

Table 1: Accuracies in (% ± std) on miniImageNet and tieredImageNet with `Sup.`: denoting supervised and `Unsup.`: unsupervised pretraining settings. Encoders: `RN`: ResNet, `WRN`: wide residual network backbones. †: denotes our reproduction. Style: **best** and second best.

			miniImageNet		tieredImageNet	
Method	**Backbone**	**Setting**	**5-way 1-shot**	**5-way 5-shot**	**5-way 1-shot**	**5-way 5-shot**
SimCLR (Chen et al., 2020a)	RN18	Unsup.	62.58 ± 0.37	79.66 ± 0.27	63.38 ± 0.42	79.17 ± 0.34
SwAV† (Caron et al., 2020)	RN18	Unsup.	59.84 ± 0.52	78.23 ± 0.26	65.26 ± 0.53	81.73 ± 0.24
NNCLR† (Dwibedi et al., 2021)	RN18	Unsup.	63.33 ± 0.53	80.75 ± 0.25	65.46 ± 0.55	81.40 ± 0.27
SimSiam (Chen & He, 2021)	RN18	Unsup.	62.80 ± 0.37	79.85 ± 0.27	64.05 ± 0.40	81.40 ± 0.30
HMS (Ye et al., 2022)	RN18	Unsup.	58.20 ± 0.23	75.77 ± 0.16	58.42 ± 0.25	75.85 ± 0.18
PsCo (Jang et al., 2023)	RN18	Unsup.	46.70 ± 0.42	63.26 ± 0.37	-	-
UniSiam + dist (Lu et al., 2022)	RN18	Unsup.	64.10 ± 0.36	82.26 ± 0.25	67.01 ± 0.39	84.47 ± 0.28
Meta-DM + UniSiam + dist (Hu et al., 2023)	RN18	Unsup.	65.64 ± 0.36	83.97 ± 0.25	67.11 ± 0.40	84.39 ± 0.28
MetaOptNet (Lee et al., 2019)	RN18	Sup.	64.09 ± 0.62	80.00 ± 0.45	65.99 ± 0.72	81.56 ± 0.53
Transductive CNAPS (Bateni et al., 2022)	RN18	Sup.	55.60 ± 0.90	73.10 ± 0.70	65.90 ± 1.10	81.80 ± 0.70
BECLR (Ours)	RN18	Unsup.	**75.74** ± 0.62	**84.93** ± 0.33	**76.35** ± 0.66	**84.85** ± 0.37
SwAV† (Caron et al., 2020)	RN50	Unsup.	63.34 ± 0.42	82.76 ± 0.24	68.02 ± 0.52	85.93 ± 0.33
NNCLR† (Dwibedi et al., 2021)	RN50	Unsup.	65.42 ± 0.44	83.31 ± 0.21	69.82 ± 0.54	86.41 ± 0.31
PDA-Net (Chen et al., 2021a)	RN50	Unsup.	63.84 ± 0.91	83.11 ± 0.56	69.01 ± 0.93	84.20 ± 0.69
UniSiam + dist (Lu et al., 2022)	RN50	Unsup.	65.33 ± 0.36	83.22 ± 0.24	69.60 ± 0.38	86.51 ± 0.26
Meta-DM + UniSiam + dist (Hu et al., 2023)	RN50	Unsup.	66.68 ± 0.36	85.29 ± 0.23	69.61 ± 0.38	86.53 ± 0.26
LEO (Rusu et al., 2018)	WRN	Sup.	61.76 ± 0.08	77.59 ± 0.12	66.33 ± 0.05	81.44 ± 0.09
CC+Rot (Gidaris et al., 2019)	WRN	Sup.	62.93 ± 0.45	79.87 ± 0.33	70.53 ± 0.51	84.98 ± 0.36
BECLR (Ours)	RN50	Unsup.	**80.57** ± 0.57	**87.82** ± 0.29	**81.69** ± 0.61	**87.86** ± 0.32

mapping of the support prototypes $\boldsymbol{P}^{\mathcal{S}}$ in the region occupied by the query embeddings $\boldsymbol{Z}^{\mathcal{Q}}$:

$$\hat{\boldsymbol{P}}^{\mathcal{S}} = \hat{\boldsymbol{\pi}}^{\star T} \boldsymbol{Z}^{\mathcal{Q}}, \quad \hat{\boldsymbol{\pi}}^{\star}_{i,j} = \frac{\boldsymbol{\pi}^{\star}_{i,j}}{\sum_j \boldsymbol{\pi}^{\star}_{i,j}}, \forall i \in [NQ], j \in [N], \tag{6}$$

where $\hat{\boldsymbol{\pi}}^{\star}$ is the normalized transport plan and $\hat{\boldsymbol{P}}^{\mathcal{S}}$ are the transported support prototypes. Finally, we fit a logistic regression classifier on $\hat{\boldsymbol{P}}^{\mathcal{S}}$ to infer on the unlabeled query set. As shown in Section 5, `OpTA` successfully minimizes the distribution shift (between support and query sets), and thus, the effects of sample bias. Note that `OpTA` can straightforwardly be applied on top any U-FSL approach. An overview of `OpTA` and our inference strategy is illustrated in Fig. 4.

5 Experimental Evaluation

Our Goal. In this section, we aim to address the following three questions:
(**Q1**) How does `BECLR` perform against state-of-the-art in *in-domain* and *cross-domain* settings?
(**Q2**) Does `DyCE` affect pretraining performance by establishing *separable memory partitions*?
(**Q3**) Does our proposed OT-based feature alignment strategy (`OpTA`) address the *sample bias*?

Our implementation and training details are fully described in Appendix A.

Benchmark Datasets. We evaluate `BECLR` in terms of its *in-domain* performance on the two most widely adopted few-shot image classification datasets: miniImageNet (Vinyals et al., 2016) and tieredImageNet (Ren et al., 2018). We also evluate `BECLR` in a *cross-domain* setting on the Caltech-UCSD Birds (CUB) dataset (Welinder et al., 2010) and a more recent cross-domain FSL (CDFSL) benchmark (Guo et al., 2020). In these settings, miniImageNet is used as the pretraining (source) dataset, and CUB (in Table 3), ChestX (Wang et al., 2017), ISIC (Codella et al., 2019), EuroSAT (Helber et al., 2019) and CropDiseases (Mohanty et al., 2016) (in Table 2) as the test (target) datasets.

5.1 Evaluation Results

We report test accuracies with 95% confidence intervals over 2000 test episodes, each with 15 query shots per class, for all tested datasets. The performance on miniImageNet, tieredImageNet and miniImageNet → CUB is evaluated on (5-way, {1, 5}-shot) classification tasks, whereas for mini-ImageNet → CDFSL we test on (5-way, {5, 20}-shot) tasks, as is customary across the literature (Guo et al., 2020; Ericsson et al., 2021). We compare our `BECLR` performance against a wide variety of methods: from (i) established SSL baselines (Chen et al., 2020a; Grill et al., 2020; Caron et al., 2020; Chen et al., 2020b; Zbontar et al., 2021; Chen & He, 2021; Dwibedi et al., 2021) to (ii) state-of-the-art U-FSL approaches (Chen et al., 2021a; Lu et al., 2022; Shirekar et al., 2022b; Chen

Table 2: Accuracies in (% ± std) on miniImageNet → CDFSL. †: our reproduc. Style: **best** and second best.

	ChestX		ISIC		EuroSAT		CropDiseases	
Method	**5 way 5-shot**	**5 way 20-shot**	**5 way 5-shot**	**5 way 20-shot**	**5 way 5-shot**	**5 way 20-shot**	**5 way 1-shot**	**5 way 20-shot**
SwAV† (Caron et al., 2020)	25.70 ± 0.28	30.41 ± 0.25	40.69 ± 0.34	49.03 ± 0.30	84.82 ± 0.24	90.77 ± 0.26	88.64 ± 0.26	95.11 ± 0.21
NNCLR† (Dwibedi et al., 2021)	25.74 ± 0.41	29.54 ± 0.45	38.85 ± 0.56	47.82 ± 0.53	83.45 ± 0.57	90.80 ± 0.39	90.76 ± 0.57	95.37 ± 0.37
SAMPTransfer (Shirekar et al., 2022b)	26.27 ± 0.44	34.15 ± 0.50	47.60 ± 0.59	**61.28** ± 0.56	85.55 ± 0.60	88.52 ± 0.50	91.74 ± 0.55	96.36 ± 0.28
PsCo (Jang et al., 2023)	24.78 ± 0.23	27.69 ± 0.23	44.00 ± 0.30	54.59 ± 0.29	81.08 ± 0.35	87.65 ± 0.28	88.24 ± 0.31	94.95 ± 0.18
UniSiam + dist (Lu et al., 2022)	**28.18** ± 0.45	**34.58** ± 0.46	45.65 ± 0.58	56.54 ± 0.5	86.53 ± 0.47	93.24 ± 0.30	92.05 ± 0.50	96.83 ± 0.27
ConFeSS (Das et al., 2021)	27.09	33.57	**48.85**	60.10	84.65	90.40	88.88	95.34
BECLR (Ours)	27.73 ± 0.24	33.81 ± 0.25	44.48 ± 0.31	57.89 ± 0.29	**88.55** ± 0.23	**93.92** ± 0.14	**94.21** ± 0.25	**97.72** ± 0.13

et al., 2022; Hu et al., 2023; Jang et al., 2023). Additionally, we compare against a set of competitive supervised baselines (Rusu et al., 2018; Gidaris et al., 2019; Lee et al., 2019; Bateni et al., 2022).

Q1-a: In-Domain Setting. The results on both miniImageNet and tieredImageNet in the (5-way, $\{1, 5\}$-shot) settings are reported in Table 1 (see Appendix C for a more complete version). Regardless of backbone depth, BECLR sets a *new state-of-the-art* on both datasets, showing up to a 14% (66.68 → 80.57) setting, and 2.5% (85.29 → 87.82) gains on miniImageNet over the prior art in U-FSL for the 1-shot and 5-shot settings, respectively. The results on tieredImageNet are analogous. Interestingly, BECLR outperforms all supervised baselines, without access to the base class labels. Notice that the *significant gains* we report above prior art in the 1-shot setting corroborate the efficacy of BECLR, as well as the major impact of the proposed OpTA inference strategy.

Q1-b: Cross-Domain Setting. These settings have recently been proposed for evaluating the generalization capability of FSL approaches to unseen (during pretraining) datasets. We follow the commonly adopted setting, where we pretrain on miniImageNet and evaluate on CDFSL (consisting of ChestX, ISIC, EuroSAT, CropDiseases) and CUB, the results of which are summarized in Tables 2 and 3, respectively (see Appendix C for more complete versions). BECLR again sets a new state-of-the-art on CUB, EuroSAT, and CropDiseases, and remains competitive to prior art on ChestX and ISIC. Notably, the data distributions of ChestX and ISIC are considerably different from those of miniImageNet. This leads to poor initial representations at the inference stage, which directly influence the efficacy of OpTA, as is empirically corroborated in Fig. 8.

Table 3: Accuracies in (% ± std) on miniImageNet → CUB. †: our reproduc. Style: **best** and second best.

	miniImageNet → CUB	
Method	**5-way 1-shot**	**5-way 5-shot**
SimCLR (Chen et al., 2020a)	38.25 ± 0.49	55.89 ± 0.46
SwAV† (Caron et al., 2020)	38.34 ± 0.51	53.94 ± 0.43
NNCLR† (Dwibedi et al., 2021)	39.37 ± 0.53	54.78 ± 0.42
Barlow Twins (Zbontar et al., 2021)	40.46 ± 0.47	57.16 ± 0.42
Laplacian Eigenmaps (Chen et al., 2022)	41.08 ± 0.48	58.86 ± 0.45
PsCo (Jang et al., 2023)	-	57.38 ± 0.44
BECLR (Ours)	**42.12** ± 0.55	**59.49** ± 0.45

Q1-c: Backbone Study. To substantiate the impact of our design choices in DyCE and BECLR, we compare against some of the most influential contrastive SSL approaches: SimCLR (Chen et al., 2020a), SwAV (Caron et al., 2020), NNCLR (Dwibedi et al., 2021) and the prior U-FSL state-of-the-art: UniSiam (Lu et al., 2022), in terms of *pure pretraining* performance, by directly evaluating the pretrained model on downstream FSL tasks (i.e., no fine-tuning or our add-on inference strategy OpTA). Fig. 5 summarizes this comparison for various network depths RN$\{10, 18, 34, 50\}$ in the (5-way, $\{1, 5\}$-shot) settings on miniImageNet. BECLR again outperforms all U-FSL/SSL frameworks for all backbone configurations, even without any supervised inference strategy or OpTA.

Q2: Latent Memory Space Evolution. As a qualitative demonstration, we visualize 30 memory embeddings from 25 partitions $\mathcal{P}_i$ within DyCE for the initial (left) and final (right) state of our latent memory space ($\mathcal{M}$). The 2-D UMAP (McInnes et al., 2018) plots in Fig. 6 provide qualitative evidence of a significant improvement in terms of cluster separation, as training progresses. To quantitatively substantiate this finding, the quality of the memory clusters is also measured by the Davies-Bouldin score (DBI) (Davies & Bouldin, 1979), with a lower DBI indicating better inter-cluster separation and intra-cluster "tightness". The DBI value is significantly lower between partitions $\mathcal{P}_i$ in the final state of $\mathcal{M}$, further corroborating DyCE's ability to establish highly separable partitions. This pseudo-class cognizance of DyCE is explored in more detail in Appendix C.

Q3-a: Impact of OT-based Feature Alignment (OpTA). We visualize the UMAP projections for a randomly sampled (3-way, 1-shot) miniImageNet test episode. Fig. 7 illustrates the original $\boldsymbol{P}^{\mathcal{S}}$ (left) and transported $\hat{\boldsymbol{P}}^{\mathcal{S}}$ (right) support prototypes (⧫), along with the embeddings of the query set $\boldsymbol{Z}^{\mathcal{Q}}$ (•) and their latent distributions (in contours). As can be seen on the left-hand side (before applying OpTA), the original prototypes $\boldsymbol{P}^{\mathcal{S}}$ are highly biased and deviate from the latent query

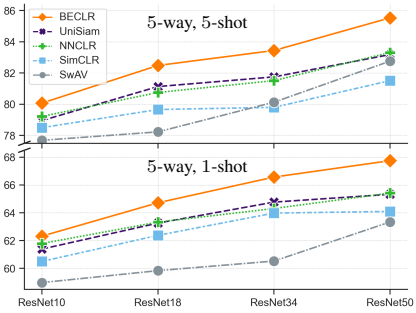


Figure 5: BECLR outperforms all baselines, in terms of pure pretraining performance on miniImageNet in the U-FSL setting.

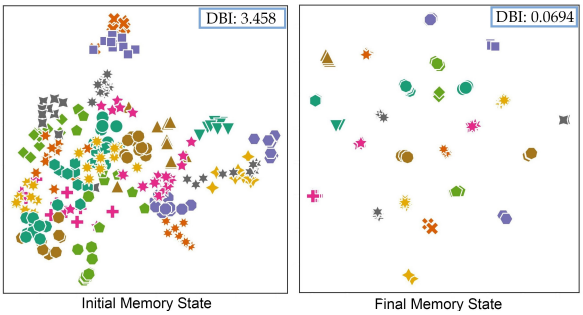


Figure 6: The dynamic updates of DyCE allow for our memory ($\mathcal{M}$) to evolve into a highly separable, partitioned latent space. Different combinations of (colors, markers) indicate partitions.

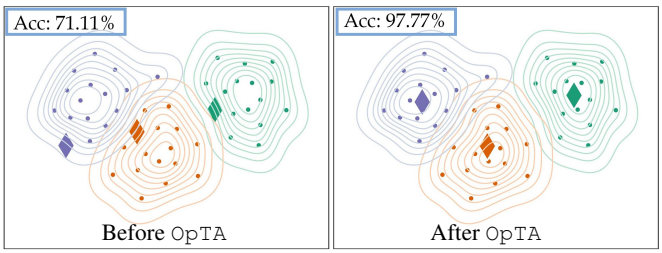


Figure 7: Our OT-based feature alignment (OpTA) effectively reduces the distribution shift between support and query sets, alleviating the sample bias. (⧫): support prototypes. (●): query set embeddings.

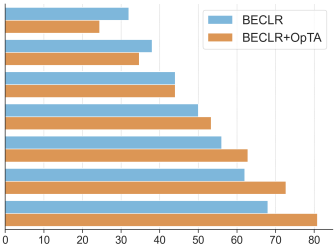


Figure 8: OpTA yields larger performance gains, the better the pretrain performance on miniImageNet.

distributions. In stark contrast, OpTA pushes the transported prototypes $\hat{\boldsymbol{P}}^{\mathcal{S}}$ much closer to the query distributions (contour centers), effectively diminishing sample bias. This also aligns with the significantly higher final accuracy of the episode using the transported prototypes (after OpTA), reported in the figure.

Q3-b: Relation between BECLR Pretraining and OpTA Inference Our proposed OT-based feature alignment strategy (OpTA) operates under the assumption that query embeddings $\boldsymbol{Z}^{\mathcal{Q}}$ are representative of actual class distributions. Consequently, its efficiency depends on representations of sufficient quality from the BECLR pretrained feature extractor. Fig. 8 assesses this intuition by comparing our pretraining and downstream performance on miniImageNet for the (5-way, 1-shot) setting. As can be seen, when the initial pretraining performance is poor, OpTA does not offer a meaningful performance boost (and actually leads to performance degradation). On the contrary, it offers a significant (ever-increasing) boost, as BECLR achieves a better pretraining performance. In a nutshell, these two steps (pretraining and inference) are highly intertwined in the U-FSL setting.

5.2 Ablation Studies

We investigate the impact of the main components of our approach, along with various important hyper-parameters. We use the (5-way, $\{1, 5\}$-shot) settings for 2000 miniImageNet test episodes and train with a ResNet50 backbone for all our ablation studies.

Main Component Ablation. As part of our ablations, we investigate the impact of each of the main components of BECLR: masking, momentum teacher encoder (EMA), our dynamic memory module (DyCE) and our inference strategy (OpTA). Table 4 demonstrates the necessity of each component, by sequentially adding the components one by one. When applied individually, masking degrades performance, while EMA gives a slight boost (1%)

Table 4: Ablation study of BECLR's main components for miniImageNet. Accuracies in (% ± std).

Masking	EMA	DyCE	OpTA	5-way 1-shot	5-way 5-shot
-	-	-	-	63.57 ± 0.43	81.42 ± 0.28
✓	-	-	-	54.53 ± 0.42	68.35 ± 0.27
-	✓	-	-	65.02 ± 0.41	82.33 ± 0.25
✓	✓	-	-	65.33 ± 0.44	82.69 ± 0.26
✓	✓	✓	-	67.75 ± 0.43	85.53 ± 0.27
✓	✓	✓	✓	**80.57** ± 0.57	**87.82** ± 0.29

Table 5: Hyperparameter ablation study for miniImageNet (5-way, 5-shot) tasks. Accuracies in (% ± std).

Masking Ratio		# of NNs (k)		# of Clusters (n)		Memory Size (M)		Neg. Loss Weight (λ)		Output Dim. (d)		Memory Updating Scheme	
Value	Accuracy	Value	Accuracy	Value	Accuracy	Value	Accuracy	Value	Accuracy	Value	Accuracy	Value	Accuracy
10%	86.59 ± 0.25	1	86.58 ± 0.27	100	85.27 ± 0.24	2048	85.38 ± 0.25	0.0	85.45 ± 0.27	256	85.16 ± 0.26	first-in-first-out	84.05 ± 0.39
30%	**87.82** ± 0.29	3	**87.82** ± 0.29	200	**87.82** ± 0.29	4096	86.28 ± 0.29	0.1	**87.82** ± 0.29	512	**87.82** ± 0.29	kmeans	85.37 ± 0.33
50%	83.36 ± 0.28	5	86.79 ± 0.26	300	85.81 ± 0.25	8192	**87.82** ± 0.29	0.3	86.33 ± 0.29	1024	85.93 ± 0.31	DyCE	**87.82** ± 0.29
70%	77.70 ± 0.20	10	86.17 ± 0.28	500	85.45 ± 0.20	12288	85.84 ± 0.22	0.5	85.63 ± 0.26	2054	85.42 ± 0.34		

for both $\{1, 5\}$-shot settings, and their combination further increases the performance of the model. As can be seen, DyCE and OpTA are the most crucial components of our approach, with the former producing a *consistent* 2.42% and 2.84% accuracy increase from the previous model in the 1-shot and 5-shot settings, respectively, and the latter a *significant* increase of 12.82% and 2.29%.

Additional Ablations. We further analyze the classification performance and robustness of BECLR across some of the most important hyperparameters, as is summarized in Table 5. In particular, we carry out ablations on: (i) the masking ratio of student images, (ii) the number of nearest neighbors selected k, (iii) the number of memory partitions/clusters P, (iv) the size of the memory M, (v) the negative loss term weighting hyperparameter λ, (vi) the embedding latent dimension d, and (vii) the memory updating mechanism. Looking at the results, a random masking ratio of 30% gives the best performance. Note that the optimal masking ratio depends on the depth of the backbone, as previously demonstrated by Assran et al. (2022). Regarding the number of neighbors (k) selected, we observe a trade-off between adding fewer true positives and including false positives, with a value of $k = 3$ being the sweet spot. We find the optimal value for the number of memory partitions (P) to be around $2\times$ the number of classes in the training dataset (i.e., $n = 200$ for 100 classes in miniImageNet). A trade-off can also be observed with respect to memory size (M), with performance being directly proportional to M, until the threshold of around $40\times$ the number of partitions (i.e., $M = 8192 \approx 40 \times 200$). Larger sizes actually degrade performance since older embeddings are kept in the memory space, which might no longer be representative of their latent classes. The negative loss term is not necessary to prevent a representation collapse, yet it allows the model to benefit from stronger data augmentations as illustrated by Lu et al. (2022). A weighting value of $\lambda = 0.1$ yields the best performance, which degrades slightly for values $\lambda \geq 0.3$. We find an embedding dimension of $d = 512$ to give the best results, which is in agreement with the literature (Chen & He, 2021; Lu et al., 2022). Finally, we compare the memory updating mechanism of DyCE (path (ii)) with simple first-in-first-out (He et al., 2020) updates and a kmeans-based approach, where batch embeddings $\boldsymbol{Z}$ are assigned to memory prototypes $\boldsymbol{\Gamma}$ via a kmeans clustering step instead of OT. We notice that the updating of DyCE is superior to both alternative approaches.

6 Concluding Remarks

In this work, we introduce a novel self-supervised pretraining methodology (coined as BECLR) that ingrains both instance- and class-level insights within a contrastive learning framework. BECLR employs a dynamic clustered memory (DyCE) module, for providing a meaningful positive sampling strategy and enhancing the original contrastive batch. We explore the effects of equipartitioned assignments via optimal transport for updating DyCE and maintaining a highly-separable latent memory space. We accentuate the sample bias problem in U-FSL and propose an intuitive and effective OT-based feature alignment (OpTA) inference strategy for alleviating its effects. Our extensive evaluation results and qualitative experiments corroborate the efficacy of our design choices in BECLR, DyCE and OpTA on a variety of both in-domain and cross-domain U-FSL tasks. We demonstrate that BECLR sets a new state-of-the-art on the two most widely adopted few-shot classification benchmarks: miniImageNet and tieredImageNet, as well as on miniImagenet $\rightarrow$ {CUB, CropDiseases, EuroSAT}. As future work, we plan to demonstrate the applicability of BECLR to additional downstream computer vision tasks (beyond FSL) by training on larger datasets, such as ImageNet. This would render BECLR an all-inclusive holistic approach towards representation learning. Finally, we could also explore the effects of combining our current contrastive framework with a clustering-based loss term, operating on the separable partitions of DyCE, or applying our OpTA on a supervised episodic FSL approach, within a meta-learning framework.

REFERENCES

Antreas Antoniou and Amos Storkey. Assume, augment and learn: Unsupervised few-shot meta-learning via random labels and data augmentation. *arXiv preprint arXiv:1902.09884*, 2019.

Mahmoud Assran, Mathilde Caron, Ishan Misra, Piotr Bojanowski, Florian Bordes, Pascal Vincent, Armand Joulin, Mike Rabbat, and Nicolas Ballas. Masked siamese networks for label-efficient learning. In *European Conference on Computer Vision*, pp. 456–473. Springer, 2022.

Peyman Bateni, Jarred Barber, Jan-Willem Van de Meent, and Frank Wood. Enhancing few-shot image classification with unlabelled examples. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 2796–2805, 2022.

Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. *Advances in neural information processing systems*, 33:9912–9924, 2020.

Lee Chen, Kuilin Chen, and Kuilin Chi-Guhn. Unsupervised few-shot learning via deep laplacian eigenmaps. *arXiv preprint arXiv:2210.03595*, 2022.

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

Wei-Yu Chen, Yen-Cheng Liu, Zsolt Kira, Yu-Chiang Frank Wang, and Jia-Bin Huang. A closer look at few-shot classification. *arXiv preprint arXiv:1904.04232*, 2019.

Wentao Chen, Chenyang Si, Wei Wang, Liang Wang, Zilei Wang, and Tieniu Tan. Few-shot learning with part discovery and augmentation from unlabeled images. *arXiv preprint arXiv:2105.11874*, 2021a.

Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 15750–15758, 2021.

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

Zitian Chen, Subhransu Maji, and Erik Learned-Miller. Shot in the dark: Few-shot learning with no base-class labels. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2668–2677, 2021b.

Noel Codella, Veronica Rotemberg, Philipp Tschandl, M Emre Celebi, Stephen Dusza, David Gutman, Brian Helba, Aadi Kalloo, Konstantinos Liopyris, Michael Marchetti, et al. Skin lesion analysis toward melanoma detection 2018: A challenge hosted by the international skin imaging collaboration (isic). *arXiv preprint arXiv:1902.03368*, 2019.

Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. Randaugment: Practical automated data augmentation with a reduced search space. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pp. 702–703, 2020.

Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. *Advances in neural information processing systems*, 26, 2013.

Debasmit Das, Sungrack Yun, and Fatih Porikli. Confess: A framework for single source cross-domain few-shot learning. In *International Conference on Learning Representations*, 2021.

David L Davies and Donald W Bouldin. A cluster separation measure. *IEEE transactions on pattern analysis and machine intelligence*, (2):224–227, 1979.

Guneet S Dhillon, Pratik Chaudhari, Avinash Ravichandran, and Stefano Soatto. A baseline for few-shot image classification. *arXiv preprint arXiv:1909.02729*, 2019.

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

Linus Ericsson, Henry Gouk, and Timothy M Hospedales. How well do self-supervised models transfer? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5414–5423, 2021.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pp. 1126–1135. PMLR, 2017.

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

Spyros Gidaris, Andrei Bursuc, Nikos Komodakis, Patrick Pérez, and Matthieu Cord. Boosting few-shot visual learning with self-supervision. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 8059–8068, 2019.

Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent-a new approach to self-supervised learning. *Advances in neural information processing systems*, 33:21271–21284, 2020.

Yunhui Guo, Noel C Codella, Leonid Karlinsky, James V Codella, John R Smith, Kate Saenko, Tajana Rosing, and Rogerio Feris. A broader study of cross-domain few-shot learning. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXVII 16*, pp. 124–141. Springer, 2020.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

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*, pp. 9729–9738, 2020.

Patrick Helber, Benjamin Bischke, Andreas Dengel, and Damian Borth. Eurosat: A novel dataset and deep learning benchmark for land use and land cover classification. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 12(7):2217–2226, 2019.

Kyle Hsu, Sergey Levine, and Chelsea Finn. Unsupervised learning via meta-learning. *arXiv preprint arXiv:1810.02334*, 2018.

Wentao Hu, Xiurong Jiang, Jiarun Liu, Yuqi Yang, and Hui Tian. Meta-dm: Applications of diffusion models on few-shot learning. *arXiv preprint arXiv:2305.08092*, 2023.

Huiwon Jang, Hankook Lee, and Jinwoo Shin. Unsupervised meta-learning via few-shot pseudo-supervised contrastive learning. *arXiv preprint arXiv:2303.00996*, 2023.

Siavash Khodadadeh, Ladislau Boloni, and Mubarak Shah. Unsupervised meta-learning for few-shot image classification. *Advances in neural information processing systems*, 32, 2019.

Siavash Khodadadeh, Sharare Zehtabian, Saeed Vahidian, Weijia Wang, Bill Lin, and Ladislau Bölöni. Unsupervised meta-learning through latent-space interpolation in generative models. *arXiv preprint arXiv:2006.10236*, 2020.

Steinar Laenen and Luca Bertinetto. On episodes, prototypical networks, and few-shot learning. *Advances in Neural Information Processing Systems*, 34:24581–24592, 2021.

Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. Learning representations for automatic colorization. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part IV 14*, pp. 577–593. Springer, 2016.

Dong Bok Lee, Dongchan Min, Seanie Lee, and Sung Ju Hwang. Meta-gmvae: Mixture of gaussian vae for unsupervised meta-learning. In *International Conference on Learning Representations*, 2020.

Kwonjoon Lee, Subhransu Maji, Avinash Ravichandran, and Stefano Soatto. Meta-learning with differentiable convex optimization. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10657–10665, 2019.

Shuo Li, Fang Liu, Zehua Hao, Kaibo Zhao, and Licheng Jiao. Unsupervised few-shot image classification by learning features into clustering space. In *European Conference on Computer Vision*, pp. 420–436. Springer, 2022.

Aristidis Likas, Nikos Vlassis, and Jakob J Verbeek. The global k-means clustering algorithm. *Pattern recognition*, 36(2):451–461, 2003.

Yuning Lu, Liangjian Wen, Jianzhuang Liu, Yajing Liu, and Xinmei Tian. Self-supervision can be a good few-shot learner. In *European Conference on Computer Vision*, pp. 740–758. Springer, 2022.

Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018.

Carlos Medina, Arnout Devos, and Matthias Grossglauser. Self-supervised prototypical transfer learning for few-shot classification. *arXiv preprint arXiv:2006.11325*, 2020.

Sharada P Mohanty, David P Hughes, and Marcel Salathé. Using deep learning for image-based plant disease detection. *Frontiers in plant science*, 7:1419, 2016.

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

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

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A Efros. Context encoders: Feature learning by inpainting. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2536–2544, 2016.

Ben Poole, Sherjil Ozair, Aaron Van Den Oord, Alex Alemi, and George Tucker. On variational bounds of mutual information. In *International Conference on Machine Learning*, pp. 5171–5180. PMLR, 2019.

Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. In *International conference on learning representations*, 2016.

Mengye Ren, Eleni Triantafillou, Sachin Ravi, Jake Snell, Kevin Swersky, Joshua B Tenenbaum, Hugo Larochelle, and Richard S Zemel. Meta-learning for semi-supervised few-shot classification. *arXiv preprint arXiv:1803.00676*, 2018.

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

Andrei A Rusu, Dushyant Rao, Jakub Sygnowski, Oriol Vinyals, Razvan Pascanu, Simon Osindero, and Raia Hadsell. Meta-learning with latent embedding optimization. *arXiv preprint arXiv:1807.05960*, 2018.

Ojas Shirekar, Ojas Shirekar, and Hadi Jamali-Rad. Self-supervised class-cognizant few-shot classification. In *2022 IEEE International Conference on Image Processing (ICIP)*, pp. 976–980. IEEE, 2022a.

Ojas Shirekar, Anuj Singh, and Hadi Jamali-Rad. Self-attention message passing for contrastive few-shot learning. *arXiv e-prints*, pp. arXiv–2210, 2022b.

Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. *Advances in neural information processing systems*, 30, 2017.

Jiaming Song and Stefano Ermon. Understanding the limitations of variational mutual information estimators. *arXiv preprint arXiv:1910.06222*, 2019.

Yonglong Tian, Yue Wang, Dilip Krishnan, Joshua B Tenenbaum, and Phillip Isola. Rethinking few-shot image classification: a good embedding is all you need? In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XIV 16*, pp. 266–282. Springer, 2020.

Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. *Advances in neural information processing systems*, 29, 2016.

Haoqing Wang and Zhi-Hong Deng. Cross-domain few-shot classification via adversarial task augmentation. *arXiv preprint arXiv:2104.14385*, 2021.

Xiaosong Wang, Yifan Peng, Le Lu, Zhiyong Lu, Mohammadhadi Bagheri, and Ronald M Summers. Chestx-ray8: Hospital-scale chest x-ray database and benchmarks on weakly-supervised classification and localization of common thorax diseases. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2097–2106, 2017.

Peter Welinder, Steve Branson, Takeshi Mita, Catherine Wah, Florian Schroff, Serge Belongie, and Pietro Perona. Caltech-ucsd birds 200. 2010.

Zhirong Wu, Yuanjun Xiong, Stella X Yu, and Dahua Lin. Unsupervised feature learning via non-parametric instance discrimination. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3733–3742, 2018.

Han-Jia Ye, Lu Han, and De-Chuan Zhan. Revisiting unsupervised meta-learning via the characteristics of few-shot tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(3): 3721–3737, 2022.

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

Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, and Stéphane Deny. Barlow twins: Self-supervised learning via redundancy reduction. In *International Conference on Machine Learning*, pp. 12310–12320. PMLR, 2021.

Chengxu Zhuang, Alex Lin Zhai, and Daniel Yamins. Local aggregation for unsupervised learning of visual embeddings. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 6002–6012, 2019.

A Implementation Details

BECLR is implemented on PyTorch (Paszke et al., 2019). We use the ResNet family (He et al., 2016) for our backbone networks (f_θ, f_ψ). The projection (g_θ, g_ψ) and prediction (h_θ) heads are 3- and 2-layer MLPs, respectively, as in Chen & He (2021). Batch normalization (BN) and a ReLU activation function are applied on each fully-connected (fc) layer, except for the output layers: no ReLU for projection heads (g_θ, g_ψ) and no BN, ReLU for prediction head (h_θ). We use a latent dimension of $d = 512$ in all our models and experiments, unless otherwise stated. Our DyCE memory module consists of a memory unit $\mathcal{M}$, initialized as a random table (of size $M \times d$). We also maintain up to P partitions in $\mathcal{M} = [\mathcal{P}_1, \ldots, \mathcal{P}_P]$, each of which is represented by a prototype stored in $\mathbf{\Gamma} = [\boldsymbol{\gamma}_1, \ldots, \boldsymbol{\gamma}_P]$. In practice, prototypes $\boldsymbol{\gamma}_i$ are the average of the latent embeddings stored in partition $\mathcal{P}_i$. When the memory is full for the first time, we initialize the memory cluster prototypes $\mathbf{\Gamma}$ and pseudo-labels for all stored memory embeddings, via a kmeans (Likas et al., 2003) clustering step. The memory is of size $M = 8192$ and contains $P = 200$ partitions $\mathcal{P}_i$ and cluster prototypes γ_i, when training on miniImageNet ($M = 40960$, $P = 1000$ for tieredImageNet). Note that both M and P are important hyperparameters, which would need to be carefully tuned on an unknown training dataset.

BECLR is pretrained on the training splits of miniImageNet and tieredImageNet with a batch size of 256 and 512, respectively. Following Chen & He (2021), images are resized to 224×224 for all configurations. We use the SGD optimizer with a weight decay of 10^{-4}, a momentum of 0.995, and a cosine decay schedule of the learning rate. Note that we do not require large-batch optimizers, such as LARS (You et al., 2017), or early stopping. Similarly to Lu et al. (2022), the initial learning rate is set to 0.3 for the smaller miniImageNet dataset and 0.1 for tieredImageNet, and we train for 400 and 200 epochs, respectively. Note that at the beginning of training, both the encoder representations and the memory embedding space are highly volatile. Thus, we allow for an adaptation period (empirically 20-50 epochs), during which the batch enhancement path of DyCE is not activated (i.e., $L = 2B$). In contrast, the memory update path of DyCE is activated for each training batch from the beginning of training, allowing our memory to reach a highly-separable, converged state. The temperature scalar in the loss function is set to $\tau = 2$. Upon finishing unsupervised pretraining, we only keep the last training epoch checkpoint of the student encoder (f_θ) for the subsequent inference stage (f_ψ, g_θ, g_ψ, h_θ are discarded). For the inference and downstream few-shot classification stage, we create (N-way, K-shot) tasks from the validation and testing splits of miniImageNet and tieredImageNet for model selection and evaluation, respectively. In the inference stage and in particular in the (5-way, 1-shot) setting, where the sample bias is most significant, we sequentially perform $\delta = 3$ consecutive passes of OpTA, with the transported prototypes of each pass acting as the input of the following pass.

Table 6: Pytorch-like descriptions of the data augmentations, applied in the pretraining stage of BECLR.

Data Augmentations	Description
Default	RandomResizedCrop(size=224, scale=(0.2, 1)) RandomApply([ColorJitter(brightness=0.4, contrast=0.4, saturation=0.4, hue=0.1)], p=0.1) RandomGrayScale(p=0.2) RandomApply([GaussianBlur([0.1, 2.0])], p=0.5) RandomHorizontalFlip(p=0.5)
Strong	RandomResizedCrop(size=224, scale=(0.2, 1)) RandomApply([ColorJitter(brightness=0.4, contrast=0.4, saturation=0.2, hue=0.1)], p=0.1) RandomGrayScale(p=0.2) RandomApply([GaussianBlur([0.1, 2.0])], p=0.5) RandAugment(n=2, m=10, mstd=0.5) RandomHorizontalFlip(p=0.5) RandomVerticalFlip(p=0.5)

Augmentations We describe our data augmentations in Table 6, applied on both miniImageNet and tieredImageNet images. The default data augmentations follow a common data augmentation strategy in self-supervised learning, including RandomResizedCrop (with scale in [0.2, 1.0]), random ColorJitter (Wu et al., 2018) of {brightness, contrast, saturation, hue} with a probability of 0.1, RandomGrayScale with a probability of 0.2, random GaussianBlur with a probability of 0.5 and a Gaussian kernel in $[0.1, 2.0]$, and finally, RandomHorizontalFlip with a probability of 0.5. Following

Lu et al. (2022), the default data augmentations are expanded, to yield the strong data augmentations, which also include RadomVerticalFlip with a probability of 0.5 and RandAugment (Cubuk et al., 2020) with $n = 2$ layers, a magnitude of $m = 10$, and a noise of the standard deviation of the magnitude of $mstd = 0.5$. Unless otherwise stated, we use strong data augmentations for all our experiments, and images are resized to 224×224, before being passed to the backbone networks.

B Experimental Setup

We here provide additional, detailed information on the tested in-domain and cross-domain few-shot benchmark datasets, along with our followed procedure, for evaluating the downstream performance of BECLR in the (5-way, $\{1, 5, 20\}$-shot) U-FSL settings.

B.1 Additional Details of Benchmark Datasets

MiniImageNet is a subset of ImageNet (Russakovsky et al., 2015), containing 100 classes with 600 images per class. We randomly select 64, 16, and 20 classes for training, validation, and testing, following the predominantly adopted settings of Ravi & Larochelle (2016).

TieredImageNet is a larger subset of ImageNet, containing 608 classes and a total of $779,165$ images, grouped into 34 high-level categories, 20 (351 classes) of which are used for training, 6 (97 classes) for validation and 8 (160 classes) for testing.

CDFSL consists of four distinct datasets with decreasing domain similarity to ImageNet, ranging from crop disease images in CropDiseases (Mohanty et al., 2016) and aerial satellite images in EuroSAT (Helber et al., 2019) to dermatological skin lesion images in ISIC2018 (Codella et al., 2019) and grayscale chest X-ray images in ChestX (Wang et al., 2017).

CUB consists of 200 classes and a total of $11,788$ images, split into 100 classes for training and 50 for both validation and testing as in Chen et al. (2019). Additional information for the cross-domain few-shot benchmarks is provided in Table 7.

Table 7: Overview of cross-domain few-shot benchmarks, on which BECLR is evaluated. The datasets are sorted with decreasing domain similarity to ImageNet.

ImageNet similarity	Dataset	# of classes	# of samples
High	CUB (Welinder et al., 2010)	200	11,788
Low	CropDiseases (Mohanty et al., 2016)	38	43,456
Low	EuroSAT (Helber et al., 2019)	10	27,000
Low	ISIC (Codella et al., 2019)	7	10,015
Low	ChestX (Wang et al., 2017)	7	25,848

B.2 Evaluation procedures

To evaluate the performance of BECLR, we use three independently-trained models of different random seeds (for each configuration) and report the average performance. We report test accuracies with 95% confidence intervals over 2000 test episodes, each with 15 query shots per class, for all tested datasets. The performance on miniImageNet, tieredImageNet and miniImageNet $\rightarrow$ CUB is evaluated on (5-way, $\{1, 5\}$-shot) classification tasks, whereas for miniImageNet $\rightarrow$ CDFSL we test on (5-way, $\{5, 20\}$-shot) tasks, as is customary across the literature (Guo et al., 2020).

C Additional Experimental Analysis

To further illustrate the class-cognisant aspect of DyCE (i.e., its ability to capture latent class distributions) we extract random (3-way, 3-shot) tasks from miniImageNet. For each of those embeddings, we visualize their position in the latent memory space along with their assigned memory prototype. In expectation, all 3-shot embeddings for each class should be mapped to similar areas and assigned to the same memory prototype. In Fig. 9 we illustrate the 2-D UMAP projections for a random (3-way, 3-shot) miniImageNet test task ($\bullet$), the assigned memory prototypes (γ_i) for each class ($\star$), and

some additional random memory prototypes for reference. As ca be seen, our intuition is confirmed, since all 3-shots for each class get assigned to the same memory prototype, which corroborates DyCE's ability of enhancing the positive sampling strategy within a contrastive learning framework.

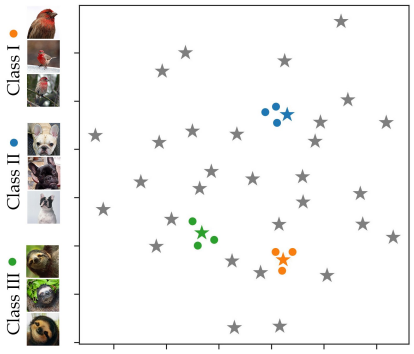


Figure 9: All three images of class $i, \forall i \in [3]$ get assigned to a single memory prototype γ_{ν_i} for a random (3-way, 3-shot) miniImageNet test task. Test task images (●), memory prototypes (⋆).

C.1 In-domain Additional Experimental Results

We here provide the full experimental results of Table 1, where we compare against additional baselines, as seen in Table 8. We compare the performance of BECLR against a wide variety of methods: from (i) established SSL baselines (Chen et al., 2020a; Chen & He, 2021) to (ii) state-of-the-art U-FSL approaches (Chen et al., 2021a; Ye et al., 2022; Shirekar et al., 2022a; Li et al., 2022; Shirekar et al., 2022b; Lu et al., 2022; Jang et al., 2023; Hu et al., 2023). Additionally, we compare with a set of supervised baselines (Finn et al., 2017; Snell et al., 2017; Rusu et al., 2018; Gidaris et al., 2019; Lee et al., 2019; Bateni et al., 2022). The top and bottom sections of the table correspond to shallower and deeper backbones, respectively.

Table 8: Accuracies in (% ± std) on miniImageNet and tieredImageNet. Problem Setting: Sup.: supervised pretraining, Unsup.: unsupervised pretraining. Encoder: Conv: convolutional blocks, RN: ResNet backbone, WRN: wide residual network backbone. †: reproduction. Style: **best** and second best. Results are grouped based on backbone depth: from shallower (Conv, RN12-34) to deeper (RN50, WRN) backbones.

			miniImageNet		tieredImageNet	
Method	**Backbone**	**Setting**	**5-way 1-shot**	**5-way 5-shot**	**5-way 1-shot**	**5-way 5-shot**
CACTUs-Proto (Hsu et al., 2018)	Conv4	Unsup.	39.18 ± 0.71	53.36 ± 0.70	-	-
ProtoTransfer (Medina et al., 2020)	Conv4	Unsup.	45.67 ± 0.79	62.99 ± 0.75	-	-
Meta-GMVAE (Lee et al., 2020)	Conv4	Unsup.	42.82 ± 0.45	55.73 ± 0.39	-	-
C3LR (Shirekar et al., 2022a)	Conv4	Unsup.	47.92 ± 1.20	64.81 ± 1.15	42.37 ± 0.77	61.77 ± 0.25
SAMPTransfer (Shirekar et al., 2022b)	Conv4b	Unsup.	61.02 ± 1.05	72.52 ± 0.68	49.10 ± 0.94	65.19 ± 0.82
LF2CS (Li et al., 2022)	RN12	Unsup.	53.14 ± 0.62	67.36 ± 0.5	53.16 ± 0.66	66.59 ± 0.57
UMTRA (Khodadadeh et al., 2019)	RN18	Unsup.	43.09 ± 0.35	53.42 ± 0.31	-	-
SimCLR (Chen et al., 2020a)	RN18	Unsup.	62.58 ± 0.37	79.66 ± 0.27	63.38 ± 0.42	79.17 ± 0.34
SwAV† (Caron et al., 2020)	RN18	Unsup.	59.84 ± 0.52	78.23 ± 0.26	65.26 ± 0.53	81.73 ± 0.24
NNCLR† (Dwibedi et al., 2021)	RN18	Unsup.	63.33 ± 0.53	80.75 ± 0.25	65.46 ± 0.55	81.40 ± 0.27
SimSiam (Chen & He, 2021)	RN18	Unsup.	62.80 ± 0.37	79.85 ± 0.27	64.05 ± 0.40	81.40 ± 0.30
HMS (Ye et al., 2022)	RN18	Unsup.	58.20 ± 0.23	75.77 ± 0.16	58.42 ± 0.25	75.85 ± 0.18
PsCo (Jang et al., 2023)	RN18	Unsup.	46.70 ± 0.42	63.26 ± 0.37	-	-
UniSiam + dist (Lu et al., 2022)	RN18	Unsup.	64.10 ± 0.36	82.26 ± 0.25	67.01 ± 0.39	84.47 ± 0.28
Meta-DM + UniSiam + dist (Hu et al., 2023)	RN18	Unsup.	65.64 ± 0.36	83.97 ± 0.25	67.11 ± 0.40	84.39 ± 0.28
MetaOptNet (Lee et al., 2019)	RN18	Sup.	64.09 ± 0.62	80.00 ± 0.45	65.99 ± 0.72	81.56 ± 0.53
Transductive CNAPS (Bateni et al., 2022)	RN18	Sup.	55.60 ± 0.90	73.10 ± 0.70	65.90 ± 1.10	81.80 ± 0.70
MAML (Finn et al., 2017)	RN34	Sup.	51.46 ± 0.90	65.90 ± 0.79	51.67 ± 1.81	70.30 ± 1.75
ProtoNet (Snell et al., 2017)	RN34	Sup.	53.90 ± 0.83	74.65 ± 0.64	51.67 ± 1.81	70.30 ± 1.75
BECLR (Ours)	RN18	Unsup.	**75.74** ± 0.62	**84.93** ± 0.33	**76.35** ± 0.66	**84.85** ± 0.37
SwAV† (Caron et al., 2020)	RN50	Unsup.	63.34 ± 0.42	82.76 ± 0.24	68.02 ± 0.52	85.93 ± 0.33
NNCLR† (Dwibedi et al., 2021)	RN50	Unsup.	65.42 ± 0.44	83.31 ± 0.21	69.82 ± 0.54	86.41 ± 0.31
UBC-FSL (Chen et al., 2021b)	RN50	Unsup.	56.20 ± 0.60	75.40 ± 0.40	66.60 ± 0.70	83.10 ± 0.50
PDA-Net (Chen et al., 2021a)	RN50	Unsup.	63.84 ± 0.91	83.11 ± 0.56	69.01 ± 0.93	84.20 ± 0.69
UniSiam + dist (Lu et al., 2022)	RN50	Unsup.	65.33 ± 0.36	83.22 ± 0.24	69.60 ± 0.38	86.51 ± 0.26
Meta-DM + UniSiam + dist (Hu et al., 2023)	RN50	Unsup.	66.68 ± 0.36	85.29 ± 0.23	69.61 ± 0.38	86.53 ± 0.26
LEO (Rusu et al., 2018)	WRN	Sup.	61.76 ± 0.08	77.59 ± 0.12	66.33 ± 0.05	81.44 ± 0.09
CC+Rot (Gidaris et al., 2019)	WRN	Sup.	62.93 ± 0.45	79.87 ± 0.33	70.53 ± 0.51	84.98 ± 0.36
BECLR (Ours)	RN50	Unsup.	**80.57** ± 0.57	**87.82** ± 0.29	**81.69** ± 0.61	**87.86** ± 0.32

C.2 Cross-domain Additional Experimental Results

We here provide the full experimental results of Tables 2 and 3, where we compare against additional baselines, as seen in Tables 9 and 10, respectively. We compare against any existing unsupervised baselines (Hsu et al., 2018; Khodadadeh et al., 2019; Chen et al., 2020a; Caron et al., 2020; Medina et al., 2020; Grill et al., 2020; Chen et al., 2020b; Zbontar et al., 2021; Ye et al., 2022; Chen et al., 2022; Lu et al., 2022; Shirekar et al., 2022b;a; Jang et al., 2023; Lee et al., 2020), for which this more challenging cross-domain experiment has been conducted, along with two recent methods dedicated to solving the cross-domain few-shot learning problem: ConFess (Das et al., 2021) and ATA (Wang & Deng, 2021).

Table 9: Accuracies in (% ± std) on miniImageNet → CDFSL. †: our reproduc. Style: **best** and second best.

	ChestX		ISIC		EuroSAT		CropDiseases	
Method	**5 way 5-shot**	**5 way 20-shot**	**5 way 5-shot**	**5 way 20-shot**	**5 way 5-shot**	**5 way 20-shot**	**5 way 1-shot**	**5 way 20-shot**
ProtoTransfer (Medina et al., 2020)	26.71 ± 0.46	33.82 ± 0.48	45.19 ± 0.56	59.07 ± 0.55	75.62 ± 0.67	86.80 ± 0.42	86.53 ± 0.56	95.06 ± 0.32
BYOL (Grill et al., 2020)	26.39 ± 0.43	30.71 ± 0.47	43.09 ± 0.56	53.76 ± 0.55	83.64 ± 0.54	89.62 ± 0.39	92.71 ± 0.47	96.07 ± 0.33
MoCo v2 (Chen et al., 2020b)	25.26 ± 0.44	29.43 ± 0.45	42.60 ± 0.55	52.39 ± 0.49	84.15 ± 0.52	88.92 ± 0.41	87.62 ± 0.60	92.12 ± 0.46
SwAV† (Caron et al., 2020)	25.70 ± 0.28	30.41 ± 0.25	40.69 ± 0.34	49.03 ± 0.30	84.82 ± 0.24	90.77 ± 0.26	88.64 ± 0.26	95.11 ± 0.21
SimCLR (Chen et al., 2020a)	26.36 ± 0.44	30.82 ± 0.43	43.99 ± 0.55	53.00 ± 0.54	82.78 ± 0.56	89.38 ± 0.40	90.29 ± 0.52	94.03 ± 0.37
NNCLR† (Dwibedi et al., 2021)	25.74 ± 0.41	29.54 ± 0.45	38.85 ± 0.56	47.82 ± 0.53	83.45 ± 0.57	90.80 ± 0.39	90.76 ± 0.57	95.37 ± 0.37
C3LR (Shirekar et al., 2022a)	26.00 ± 0.41	33.39 ± 0.47	45.93 ± 0.54	59.95 ± 0.53	80.32 ± 0.65	88.09 ± 0.45	87.90 ± 0.55	95.38 ± 0.31
SAMPTransfer (Shirekar et al., 2022b)	26.27 ± 0.44	34.15 ± 0.50	47.60 ± 0.59	**61.28** ± 0.56	85.55 ± 0.60	88.52 ± 0.50	91.74 ± 0.55	96.36 ± 0.28
PsCo (Jang et al., 2023)	24.78 ± 0.23	27.69 ± 0.23	44.00 ± 0.30	54.59 ± 0.29	81.08 ± 0.35	87.65 ± 0.28	88.24 ± 0.31	94.95 ± 0.18
UniSiam + dist (Lu et al., 2022)	**28.18** ± 0.45	**34.58** ± 0.46	45.65 ± 0.58	56.54 ± 0.5	86.53 ± 0.47	93.24 ± 0.30	92.05 ± 0.50	96.83 ± 0.27
ConFeSS (Das et al., 2021)	27.09	33.57	**48.85**	60.10	84.65	90.40	88.88	95.34
ATA (Wang & Deng, 2021)	24.43 ± 0.2	-	45.83 ± 0.3	-	83.75 ± 0.4	-	90.59 ± 0.3	-
BECLR (Ours)	27.73 ± 0.24	33.81 ± 0.25	44.48 ± 0.31	57.89 ± 0.29	**88.55** ± 0.23	**93.92** ± 0.14	**94.21** ± 0.25	**97.72** ± 0.13

Table 10: Accuracies in (% ± std) on miniImageNet → CUB. †: our reproduc. Style: **best** and second best.

	miniImageNet → CUB	
Method	**5-way 1-shot**	**5-way 5-shot**
CACTUs-MAML (Hsu et al., 2018)	33.48 ± 0.49	49.97 ± 0.41
UMTRA (Khodadadeh et al., 2019)	33.59 ± 0.48	50.21 ± 0.45
Meta-GMVAE (Lee et al., 2020)	38.09 ± 0.47	55.65 ± 0.42
SimCLR (Chen et al., 2020a)	38.25 ± 0.49	55.89 ± 0.46
MoCo v2 (Chen et al., 2020b)	39.29 ± 0.47	56.49 ± 0.44
BYOL (Grill et al., 2020)	40.63 ± 0.46	56.92 ± 0.43
SwAV† (Caron et al., 2020)	38.34 ± 0.51	53.94 ± 0.43
NNCLR† (Dwibedi et al., 2021)	39.37 ± 0.53	54.78 ± 0.42
Barlow Twins (Zbontar et al., 2021)	40.46 ± 0.47	57.16 ± 0.42
Laplacian Eigenmaps (Chen et al., 2022)	41.08 ± 0.48	58.86 ± 0.45
HMS (Ye et al., 2022)	40.75	58.32
PsCo (Jang et al., 2023)	-	57.38 ± 0.44
BECLR (Ours)	**42.12** ± 0.55	**59.49** ± 0.45

D Pseudocode

Here, we provide the algorithms for our BECLR pretraining methodology and our dynamic clustered memory (DyCE) in a Pytorch-like pseudocode format. Algorithm 3 provides an overview of BECLR and is equivalent to Algorithm 1, while Algorithm 4 describes the two informational paths of DyCE, similar to Algorithm 2.

Algorithm 3: Batch Enhanced Contrastive Learning (BECLR): PyTorch-like Pseudocode

```
# {f, g, h}_student: student backbone, projector, and predictor
# {f, g}_teacher: teacher backbone and projector
# DyCE_{student, teacher}: our dynamic clustered memory module for student and teacher paths (see Algorithm. 4)
def BECLR(x):                                                          # x: a mini-batch of L samples
    x = [x1,  x2] = [aug1(x),  aug2(x)]                                # concatenate the two augmented views of x
    z_s = h_student( g_student( f_student( mask( x))))                 # (2B × d): student representations
    z_t = g_teacher( f_teacher( x)).detach()                           # (2B × d): teacher representations
    z_s, z_t = DyCE_student(z_s), DyCE_teacher(z_t) # update memory via optimal transport & compute enhanced batch (2B(k+1) × d)
    loss_pos = - (z_s ∗ z_t).sum(dim=1).mean()                         # compute positive loss term
    loss_neg = (matmul(z_s, z_t.T) ∗ mask).div(temp).exp().sum(dim=1)).div(n_neg).mean().log()   # compute negative loss term
    return  loss_pos + lambda ∗ loss_neg                               # return final loss
```

Algorithm 4: `DyCE`: PyTorch-like Pseudocode

```
# z: batch representations (2B×d)
# self.memory: memory embedding space (M×d)
# self.prototypes: memory partition prototypes (P×d)
def DyCE(self, z):
  if self.memory.shape[0] == M:
    # - - - - - Path I: Top-k NNs Selection and Batch Enhancement - - - - -
    if epoch ≥ epoch_thr
      batch_prototypes = assign_prototypes(z, self.prototypes)      # (2B×d): find nearest memory prototype for each batch embedding
      y_mem = top-k(self.memory, z, batch_prototypes)          # (2Bk×d): find top-k NNs, from memory partition of nearest prototype
      z = [z, y_mem]                          # (2B(k+1)×d): concatenate batch and memory representations to create the final enhanced batch
    # - - - - - Path II: Iterative Memory Updating - - - - -
    opt_plan = sinkhorn(D(z, self.prototypes))          # get optimal assignments between batch embeddings and prototypes (Solve Eq. 4)
    self.update(z, opt_plan)  # add latest batch to memory and update memory partitions and prototypes, using the optimal assignments
    self.dequeue()                                                                  # discard the 2B oldest memory embeddings
  else:
    self.enqueue(z)                                                  # simply store latest batch until the memory is full for the first time
  return z
```

3

Deep Learning

Deep learning constitutes a subbranch of machine learning that employs biologically inspired computational models (McCulloch and Pitts 1943), known as neural networks. Such networks are becoming increasingly popular, since they are eligible to perform a wide variety of tasks from image classification, detection, object localization and scene segmentation to 3D depth estimation, text modeling, and speech synthesis, among other tasks. These networks are typically made up of individual and rather simple nodes (or **"neurons"**), stacked on top of each other to approximate complex mathematical functions. The main distinction between deep learning and classical machine learning approaches lies in the fact that the former automatically learns to extract relevant features (or representations), instead of relying on manual feature selection and design, and hence it is also known as **representation learning**.

3.1. Deep Feedforward Networks

Deep feedforward networks, also known as **multilayer perceptrons (MLPs)**, are the most standardized deep learning networks and are designed to process data and extract patterns and meaningful information for statistical generalization. Different functions (or interchangeably called layers), interconnected in a sequential chain, compose this type of network. For instance, the network shown on Figure 3.1 is comprised of 3 layers: $f^{(1)}, f^{(2)}$ and $f^{(3)}$ and can be described by the function: $f = f^{(1)} \circ f^{(2)} \circ f^{(3)}$. Such chain structures are the building blocks of neural networks. The first ($f^{(1)}$) and last ($f^{(3)}$) layers are called **input** and **output layers** respectively, whereas any intermediate layers (such as $f^{(2)}$) are called **hidden layers**. The process of passing the input x through the functions of the network to produce an output $f(x)$, is called a **forward pass**, hence the name of these networks.

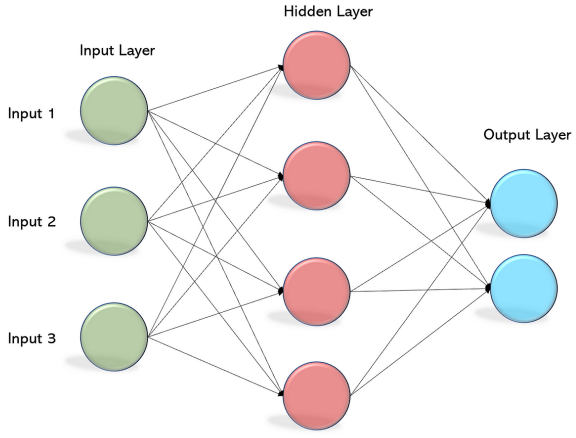


Figure 3.1: A 3-layer MLP with one hidden layer consisting of 4 hidden nodes.

Formally, the output of an MLP layer can be calculated as:

$$\boldsymbol{Y} = f(\boldsymbol{X}; \boldsymbol{\theta}) = f(\boldsymbol{X}; \boldsymbol{W}, \boldsymbol{b}) = \boldsymbol{X}\boldsymbol{W} + \boldsymbol{b}, \tag{3.1}$$

where $\boldsymbol{X} \in \mathbb{R}^{N \times d}$ represents the input of N samples and d features, $\boldsymbol{W} \in \mathbb{R}^{d \times h}$ and $\boldsymbol{b} \in \mathbb{R}^{1 \times h}$ correspond to the weights and biases of the layer respectively, and $\boldsymbol{Y} \in \mathbb{R}^{N \times h}$ represents the output.

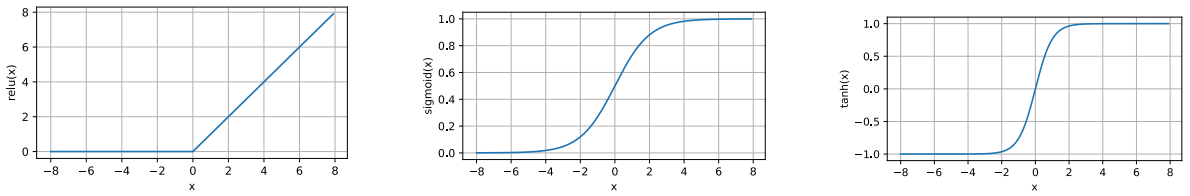
The objective of deep forward networks is to approximate some true ideal function $y = f^{\star}(x)$, mapping the input x to the output y. Network parameters θ are optimized during training, resulting in the best approximation of the true function: $f(x; \theta) \approx f^{\star}(x)$. To understand the efficiency of MLPs and the potential of deep learning in general, we need to ask the question: *"How good are these networks in approximating an ideal mathematical function ?"*. In fact, several works (Micchelli 1984; Hornik 1991; Cybenko 1992) have corroborated the property of MLPs as **universal function approximators**. That is, even with a single layer, given sufficient neurons, an MLP is capable of learning such parameters θ to approximate any mathematical function. It should be noted, however, that standard MLP layers, such as those in Figure 3.1, are only linear functions of their input. Consequently, to ensure the validity of the global approximation theory, nonlinear fixed transformations, called **activation functions**, need to be added to the affine transforms (controlled by θ) of standard layers.

3.2. Activation Functions

Activation functions are differentiable operators, without trainable parameters of their own, that determine whether a node should be activated or not. They are utilized, in order to introduce some form of non-linearity to an MLP layer:

$$\boldsymbol{Y} = g \circ f(\boldsymbol{X}; \boldsymbol{W}, \boldsymbol{b}) = g(\boldsymbol{X}\boldsymbol{W} + \boldsymbol{b}), \tag{3.2}$$

where $g(\cdot)$ the activation function. Some of the most popular activations are shown in Figure 3.2.



(a) ReLU **(b)** Sigmoid **(c)** Tanh

Figure 3.2: Popular activation functions.

The Rectified Linear Unit (ReLU) (Fukushima 1975) is perhaps the most popular activation function, that directly outputs the input in case it is positive, and zero otherwise. It is often preferred due to its computational simplicity and ability to output true zero values.

$$relu(\boldsymbol{z}) = \max(0, \boldsymbol{z}) \tag{3.3}$$

Another really important function is the **sigmoid**, which maps the input to a real value between $(0, 1)$. A sigmoid is a smooth differentiable approximation of thresholding units (McCulloch and Pitts 1943), which means that when the input value is below or above a certain threshold the output takes a value of 0 or 1 respectively. It is often applied on binary classification or logistic regression problems, but can be prone to vanishing gradients.

$$\text{sigmoid}(\boldsymbol{z}) = \frac{1}{1 + \exp(-\boldsymbol{z})} \tag{3.4}$$

The **tanh** function behaves similarly to the sigmoid and has been historically preferred for yielding better performance with MLPs. Nevertheless, it does not solve the vanishing gradient problem that sigmoids suffer from, which was tackled more effectively with the introduction of ReLU activations.

$$\tanh(\boldsymbol{z}) = \frac{1 - \exp(-2\boldsymbol{z})}{1 + \exp(-2\boldsymbol{z})} \tag{3.5}$$

3.3. Optimisation and Backpropagation

Until now, we have focused solely on the forward pass of deep neural networks, where the input x is processed by the network to output an approximation $f(x;\theta) \approx f^{\star}(x)$ of the desired output y. However, considering the random initialization of the network parameters θ, in order for this approximation to be accurate, the network needs to learn an optimal set of values for its parameters θ. Consequently, networks must be **optimized**, with respect to their parameters θ. Neural networks typically achieve this through **gradient-based** optimization algorithms, which attempt to iteratively find better sets of parameter values, to minimize a target function, known as the **loss function**.

3.3.1. Loss Function

It has already been established that MLPs act as universal function approximators. Therefore, we need both a way to measure the efficiency of this approximation and a way to update the parameters θ using gradient-based optimization. It turns out that the loss function of a neural network can satisfy both of these functions. Let $\{\boldsymbol{x}^{(i)}, y^{(i)}\}_{i=1}^{N}$ be the network training set, where $\boldsymbol{x}_i$ an input feature vector and y_i the ground truth label. For each sample in the training set, a **loss function** can then be defined to compare the prediction of the network $f(x^{(i)};\theta)$ with the expected output y and measure their similarity.

The **mean squared error (MSE)** is a simple, and rather intuitive loss function, which is really popular for regression problems, and can be defined for a single training sample as:

$$\mathcal{L}_i^{MSE}(\boldsymbol{x}^{(i)}, y^{(i)}, \boldsymbol{\theta}) = \frac{1}{2}\|f(\boldsymbol{x}^{(i)};\boldsymbol{\theta}) - y^{(i)}\|^2 \tag{3.6}$$

The **cost function** is then defined as the average of individual losses for all the training set samples:

$$\mathcal{J}^{MSE}(\boldsymbol{\theta}) = \frac{1}{N}\sum_{i=1}^{N}\mathcal{L}_i^{MSE}(\boldsymbol{x}^{(i)}, y^{(i)}, \boldsymbol{\theta}) = \frac{1}{2N}\sum_{i=1}^{N}\|f(\boldsymbol{x}^{(i)};\boldsymbol{\theta}) - y^{(i)}\|^2 \tag{3.7}$$

Another loss function, being frequently applied on deep networks, is the **cross-entropy (CE)** loss, which utilizes the **softmax** function to measure differences between two probability distributions. Let us consider a multiclass classification problem, where $y \in \{1, ..., K\}$, $f(\boldsymbol{x})$ corresponds to a vector of class probabilities (i.e., probabilities for $\boldsymbol{x}$ to belong to a particular class), and $\boldsymbol{y} = (0, ..., 0, 1, 0, ..., 0)$ the one-hot encoded vector of y. We can define the cross-entropy loss for a single training sample as follows:

$$\begin{aligned} \mathcal{L}_i^{CE}(\boldsymbol{x}^{(i)}, \boldsymbol{y^{(i)}}, \boldsymbol{\theta}) &= -\sum_{j=1}^{K} y_j^{(i)} \log(\hat{y}_j^{(i)}) \ , \ where \\ \hat{\boldsymbol{y}} = \text{softmax}(f(\boldsymbol{x}^{(i)};\boldsymbol{\theta}) &= \frac{\exp(f(\boldsymbol{x}^{(i)};\boldsymbol{\theta}))}{\sum_{k=1}^{K}\exp(f(\boldsymbol{x}^{(k)};\boldsymbol{\theta})} \end{aligned} \tag{3.8}$$

The softmax operator ensures that the network's output vector of class probabilities is non-negative and sums up to 1, thus $\boldsymbol{y}$ and $\hat{\boldsymbol{y}}$ can be viewed as two distinct probability distributions. By averaging over the entire training set, we can once again extract the aggregate cost function ($\mathcal{J}(\boldsymbol{\theta})$):

$$\mathcal{J}^{CE}(\boldsymbol{\theta}) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{K} y_j^{(i)} \log(\hat{y}_j^{(i)}), \tag{3.9}$$

where $y_j^{(i)}$ and $\hat{y}_j^{(i)}$ are the true and predicted output for the i^{th} input sample $\boldsymbol{x}_i$ and the j^{th} class.

3.3.2. Stochastic Gradient Descent

Gradient Descent (Robbins and Monro 1951), along with its variations, constitute the most widely used parameter optimization methods in deep learning. As suggested by its name, gradient descent is a gradient-based (or first-order) iterative optimization algorithm capable of finding a local minimum or maximum on convex functions. Let us denote a differentiable function to be optimised $y = f(x)$, where $x, y \in \mathbb{R}$, and $f'(x) = \frac{dx}{dy}$ its derivative (or **gradient**), corresponding to the slope of the function. The slope indicates the effect of making a small change ϵ to the input x on the output of the network:

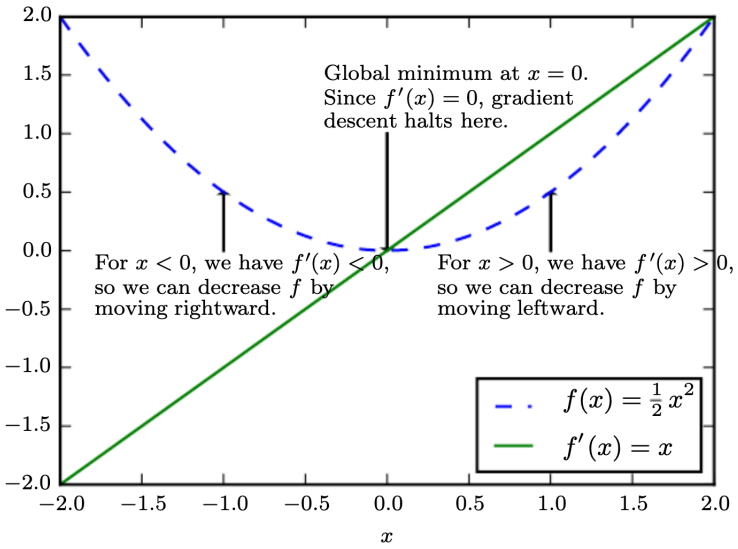


Figure 3.3: Illustration of the gradient descent algorithm, which utilizes the slope of the cost function to guide the direction of movement for the parameters. Regardless of the initial position (positive or negative) of x, the gradient descent will converge to a local minima. Figure courtesy of Ian Goodfellow and Courville 2018.

$f(x + \epsilon) \approx f(x) + \epsilon f'(x)$. Hence, the gradient information can guide the direction of movement of the parameters θ, until they converge in a local minima, where $f'(x) \approx 0$, as illustrated on Figure 3.3. The direction of steepest descent guides the iterative process and can define the following update rule:

$$\boldsymbol{x}' = \boldsymbol{x} - \epsilon \nabla_x f(\boldsymbol{x}) \tag{3.10}$$

In the context of deep learning, gradient descent operates on the **cost function** and we are mostly interested in finding a local minima (close to the global minimum), because of the vast parameter space (in the thousands or even millions). A gradient descent step can be defined as:

$$\nabla_{\boldsymbol{\theta}} \mathcal{J}(\boldsymbol{\theta}) = \frac{1}{N} \sum_{i=1}^{N} \nabla_{\boldsymbol{\theta}} \mathcal{L}_i(\boldsymbol{x}^{(i)}, y^{(i)}, \boldsymbol{\theta}), \tag{3.11}$$

with a computation cost of $\mathcal{O}(N)$. However, deep learning is extremely "data-hungry", often relying on very large training datasets for better generalization, which can make gradient descent computationally intractable, when the training set contains millions of samples. A possible workaround is **stochastic gradient descent (SGD)** (Amari 1993), where only a significantly smaller minibatch of $m << N$ samples is used to calculate the gradient. In practice, the expectation of the gradient, utilized by SGD, is a good approximation of the full gradient and has been broadly adopted by deep learning approaches. A stochastic gradient descent step can be defined as:

$$\boldsymbol{G} = \frac{1}{m} \nabla_{\boldsymbol{\theta}} \sum_{i=1}^{m} \mathcal{L}_i(\boldsymbol{x}^{(i)}, y^{(i)}, \boldsymbol{\theta}) \tag{3.12}$$

The entire training set (of size N) is used for making multiple passes of randomly sampled mini-batches of size m. Finally, the parameters of the network θ are updated by taking steps in the opposite direction of the gradient: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \epsilon \boldsymbol{G}$.

3.3.3. Backpropagation

In the previous subsection, the stochastic gradient descent procedure was explained, which necessitates computing the derivative of the cost function with respect to the network parameters θ, which can be thousands or even millions. The **backpropagation** algorithm (Rumelhart, Hinton, and Williams 1986) facilitates this need for an efficient stochastic gradient descent computation. It involves computing the gradients of the outputs of the function (or layer) with respect to their inputs and parameters,

starting from the final output of the network (corresponding to the value of the loss function) and working **backward** through the network layers. This recursive computation is terminated when the input layer of the neural network has been reached, ensuring that all required derivatives are calculated in the process. In essence, the whole procedure is an application of the **chain rule of derivatives** of calculus. Finally, given a deep neural network with h layers, backpropagation collects all the computed derivatives into a vector of gradients as follows:

$$\boldsymbol{\nabla}_{\boldsymbol{\theta}}\mathcal{L}(f(\boldsymbol{x};\boldsymbol{\theta}),y) = \left[\frac{\partial\mathcal{L}}{\partial w^{(h)}}, \frac{\partial\mathcal{L}}{\partial b^{(h)}}, \frac{\partial L}{\partial w^{(h-1)}}, \frac{\partial\mathcal{L}}{\partial b^{(h-1)}}, \cdots, \frac{\partial\mathcal{L}}{\partial w^{(1)}}, \frac{\partial\mathcal{L}}{\partial b^{(1)}}\right]^{\top}, \tag{3.13}$$

where $w^{(l)}$ and $b^{(l)}$ are the weights and biases of layer l and $\boldsymbol{\nabla}_{\boldsymbol{\theta}}L(f(\boldsymbol{x};\boldsymbol{\theta}),y)$ a vector of partial derivatives with respect to the neural network's parameters.

4

Convolutional Neural Networks

The novel **convolutional neural network (CNN)** architecture, first introduced by LeCun, Boser, et al. (1989), is specifically designed to handle contextual data with a "grid-like" structure. This property makes CNNs highly suitable for processing image data (represented in $2D$ pixel grids) and thus a natural tool for efficiently tackling computer vision tasks. The first CNN variants (W. Zhang et al. 1990; LeCun, Jackel, et al. 1995), were capable of extracting simple visual features, such as edges or corners, but it was not until AlexNet (Krizhevsky, Sutskever, and Hinton 2012), when CNNs started achieving groundbreaking results and firmly established a dominant role in computer vision problems. Before further elaborating on the details and attractive qualities of the CNN architecture, let us consider the limitations of MLPs when applied to image data. Firstly, an MLP would need to flatten an input image ($N \times N$ pixels), yielding an input dimension of N^2. Therefore, when the image resolution is high, the network would need a massive number of parameters, which makes the optimization process both time and space inefficient. Additionally, an MLP operates on the latent assumption that each pixel is independent (because of the flattened input), completely disregarding the **spatial structure** of an image, which is really important for learning discriminative image features. CNNs are capable of overcoming both of these MLP shortcomings, thanks to the **convolution** operator, finding applications in a wide range of visual tasks (Eickenberg et al. 2017; Kuzovkin et al. 2018; Lindsay 2021).

Figure 4.1 provides an overview of the CNN architecture, which is typically composed of two major components: (i) a feature extractor and (ii) a classification head. The building blocks of the feature extractor are convolutional layers, followed by non-linear activation functions, and some spatial pooling, whereas the classification head flattens the feature extractor's output, before passing it to fully connected layers and finally a softmax operator, to extract class probabilities.

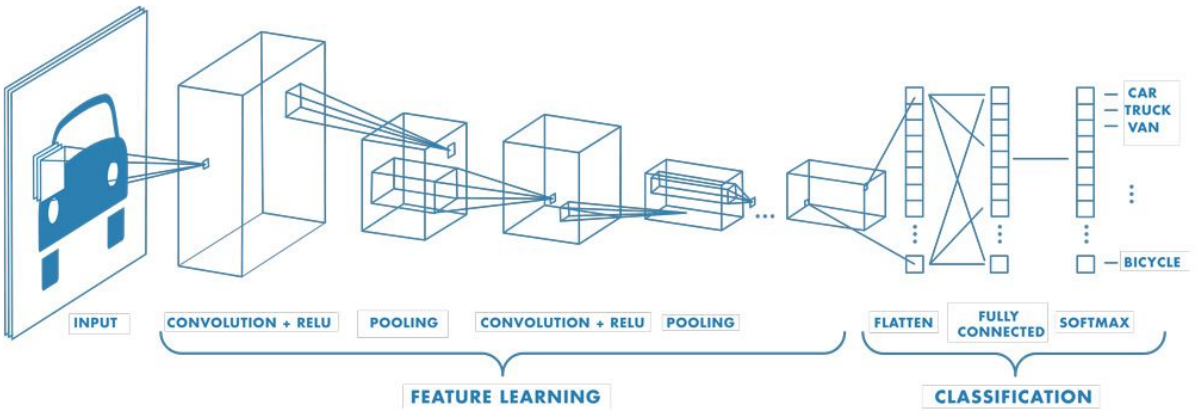


Figure 4.1: An overview of the CNN architecture.

4.1. Convolution

Before delving deeper into CNNs, it seems appropriate to thoroughly understand the **convolution** operator, from which CNNs are named. In particular, given two real-valued functions $x, w \in \mathbb{R}$, the convolution function can be defined as:

$$s(t) = (x * w)(t) = \int x(a)w(t-a)\,da, \tag{4.1}$$

where x is referred to as the **input**, w as the **kernel**, and the output $s(t)$ as a **feature map**. Intuitively, the overlap between input functions is measured by convolution, when one (the kernel w in this case) has been flipped and shifted by t. For discrete functions, the integral becomes a summation:

$$s(t) = (x * w)(t) = \sum_{-\infty}^{\infty} x(a)w(t-a) \tag{4.2}$$

However, in deep learning and particularly computer vision settings we are interested in multi-dimensional data (such as 2-dimensional images), and thus need to apply the convolution operator over multiple axes. In the $2D$ case, the convolution function can be defined as

$$\boldsymbol{S}(i,j) = (\boldsymbol{I} * \boldsymbol{K})(i,j) = \sum_m \sum_n \boldsymbol{I}(m,n)\boldsymbol{K}(i-m, j-n), \tag{4.3}$$

where $\boldsymbol{I}$, $\boldsymbol{K}$ the 2-dimensional **input** and **kernel**, respectively. Due to the *commutative* property of convolutions, Equation (4.3) can be rewritten to accommodate a more convenient implementation:

$$\boldsymbol{S}(i,j) = (\boldsymbol{K} * \boldsymbol{I})(i,j) = \sum_m \sum_n \boldsymbol{I}(i-m, j-n)\boldsymbol{K}(m,n). \tag{4.4}$$

Due to the *commutative* property of convolution, it seems like the kernel is flipped relative to the input. This means that as m, n increase, the input and kernel indexes increase and decrease, respectively. Empirically, it has been shown that while this is useful for mathematical proofs, it is not, in fact, necessary for actual implementation purposes. Instead, most deep learning libraries, such as PyTorch (Paszke et al. 2019) implement **cross-correlation** as in Equation (4.5), which is equivalent to convolution, without the flipping of the kernel. For the remainder of this report, the terms *cross-correlation* and *convolution* will be used interchangeably, as is common practice in the deep learning literature.

$$\boldsymbol{S}(i,j) = (\boldsymbol{K} * \boldsymbol{I})(i,j) = \sum_m \sum_n \boldsymbol{I}(i+m, j+n)\boldsymbol{K}(m,n). \tag{4.5}$$

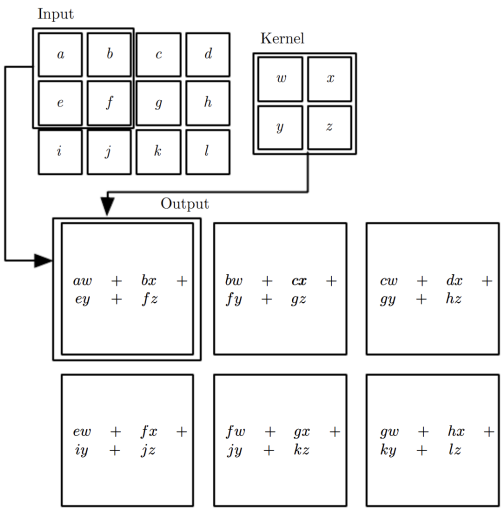


Figure 4.2: An example of $2D$ convolution (cross-correlation), as implemented by most Deep Learning frameworks. Figure courtesy of Ian Goodfellow and Courville 2018.

Figure 4.2 provides a visual interpretation of the convolution operator, when applied on a $2D$ grid (which could be interpreted as an image). In practise, the $2D$ kernel is a **sliding window** over the input feature map. At each location, the products between the kernel and the input elements are calculated and subsequently summed, resulting in the convolution output. Each output element corresponds to a single application of the kernel on the input. In this simple example, only a single kernel is applied on the input, but in the more general case multiple kernels could be applied by the same convolutional layer. The number of kernels in a convolutional layer is called the number of **channels**.

4.2. Pooling

Another fundamental component of the CNN architecture are **pooling** operations. The motivation behind the pooling operation is to **summarize** certain regions of a feature map using aggregation functions, for example, taking the maximum or average values over a window of the input. Consequently, pooling can be used effectively to reduce the size of the input feature map (pooling with downsampling). Pooling operators are applied similarly to the convolutional operators as in Figure 4.2, i.e., by using a sliding window over the input feature map. The only distinction is that the linear combination of input and kernel elements of the convolution is replaced by an aggregation function (e.g., maximum or average) over the input elements in the case of pooling.

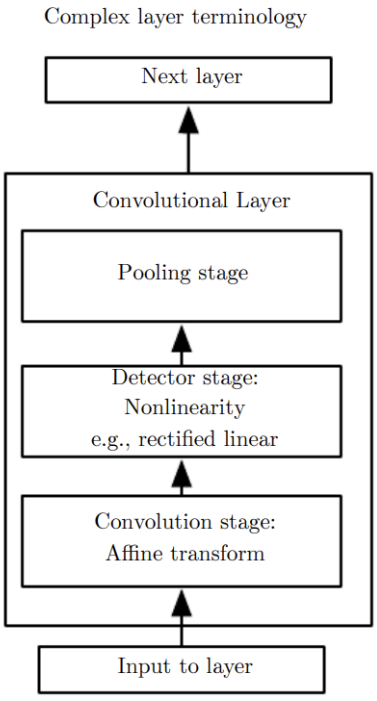


Figure 4.3: The main components of a typical convolutional layer. Figure courtesy of Ian Goodfellow and Courville 2018.

Pooling plays a crucial role in learning meaningful representations from images, in that it allows the network to be approximately **invariant** to small translations of the input. Invariance to local translation implies that if we shift the input by a small amount, the majority of the pooled output values remain unchanged. Invariance can be highly valuable in cases where we are mostly interested in the presence of a particular feature rather than its precise location. For instance, a class-discriminative feature can help us recognize an image of its particular class regardless of its spatial position in the image. Due to the pooling operators, CNNs are robust to slight shifts in the input data, allowing them to generalize better and focus on detecting important features rather than their exact positions. Finally, Figure 4.3 illustrates the components of a convolutional layer, which is composed of a **convolution stage**, followed by an **activation function** (non-linearity) and finally a **pooling stage**, and in turn constitutes the building block of the CNN architecture.

4.3. Feature Extraction with CNNs

Several components influence the behavior of a convolutional layer (in a CNN), such as the **padding** and **stride** values, along with the **kernel size** and the number of **channels**. Therefore, it is not trivial to infer the impact of each kernel and convolutional operation. Empirically, however, it has been shown that each kernel acts as an **individual feature extractor**, and thus by stacking these kernels and learning their weights, a CNN is, in fact, learning a wide range of image features.

Figure 4.4 was generated using feature visualization techniques (Olah, Mordvintsev, and Schubert 2018) and showcases the different features and patterns a CNN is capable of extracting from images. It should be noted that the deeper a convolutional layer is, the more intricate and complicated patterns it is able to detect. In particular, the first layers typically learn **simple and much more generic visual features**, such as edges, lines, and basic shapes and textures. As the number of layers increases, however, the network is able to aggregate the information from previous layers to learn much **more specific features, associated with objects and classes.** For instance, in Figure 4.4 the network has learnt features that correspond to the shape of a dog head and snout. This demonstrates

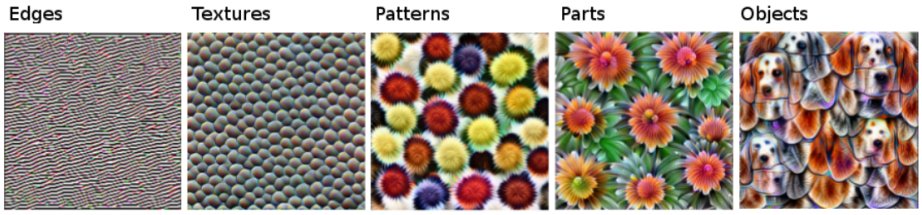


Figure 4.4: Features learned by a CNN. As the number of layers increases, the complexity of the features and patterns learned by the network increases as well. Figure courtesy of Molnar 2020.

how CNNs progressively acquire more sophisticated and discerning representations as information passes through deeper layers of the network.

4.4. Deep Residual Networks

A recurring question regarding deep neural networks, and CNNs in particular, can be formulated as: **"Is deeper always better ?"**. In fact, several studies (Simonyan and Zisserman 2014; Szegedy et al. 2015) have shown that network depth is a crucial factor for CNN performance. However, a possible obstacle to stacking an ever increasing number of layers is the **vanishing gradient** problem (Hochreiter 1991; Bengio, Simard, and Frasconi 1994), which hampers the optimization and convergence of the network. Moreover, other studies (He and Sun 2015; He, X. Zhang, et al. 2016) have explored a **degradation** behavior: as the network depth increases the accuracy saturates and then degrades rapidly. This higher training error of deeper models has been shown to be unrelated to overfitting (He and Sun 2015), indicating that not all parts of a deep network are similarly easy to optimize.

Deep **residual networks (ResNets)** (He, X. Zhang, et al. 2016) address both of these issues and have been shown to (i) make deeper networks much easier to optimize and (ii) significantly improve performance on several downstream tasks. Rather than relying on the expectation that a few stacked layers will directly learn the desired underlying mapping, ResNets explicitly let these layers fit a residual mapping. In this setting, the layers are designed to learn the residual or the difference between the desired mapping and the input, and, in doing so, they can focus on capturing any deviations needed to push the initial input closer to the desired output. ResNets are implemented by leveraging identity **short-cut connections** (Bishop 1995; Ripley 2007) as shown in Figure 4.5, which skip one or more layers and are used to stack the outputs or shallower and deeper layers. Finally, ResNets can be easily implemented by most deep learning libraries, such as PyTorch (Paszke et al. 2019), without adding extra parameters or computational complexity. `BECLR` utilizes the ResNet architecture as the backbone network (or feature extractor) for both the student and teacher paths. After pretraining is completed, only the student ResNet encoder is kept, which is in turn utilised for few-shot image classification.

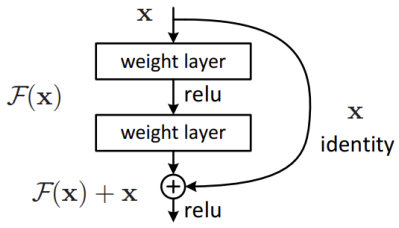


Figure 4.5: The building block of residual learning. Figure courtesy of He, X. Zhang, et al. 2016.

5

Self-Supervised Learning

Self-supervised learning (SSL) constitutes a really promising approach to efficiently advance machine learning and has even been referred to as the *"dark matter of artificial intelligence"* (Balestriero et al. 2023). In contrast to traditional supervised learning, which relies on labeled data, SSL can take advantage of large amounts of unlabeled data to learn meaningful representations (T. Chen et al. 2020; Misra and Maaten 2020). SSL has already been instrumental in pushing the state-of-the-art in the **natural language processing (NLP)** (Brown et al. 2005; Popel et al. 2020) domain on various tasks by exploiting large-scale unlabeled text corpora. Following the success of SSL in NLP, there has been increasing interest in applying SSL on computer vision tasks. In fact, SEER (Goyal, Caron, et al. 2021) was trained on 1 billion images and successfully demonstrated how SSL approaches can be on par or even surpass supervised models on highly competitive computer vision benchmarks like ImageNet (Deng et al. 2009), without having access to any labels.

In the absence of supervision, SSL methods define **pretext tasks** from unlabeled inputs to generate descriptive and interpretable representations (Hastie et al. 2009; Ian Goodfellow and Courville 2018). In the NLP case for instance, a common SSL pretext task involves the masking of a word and the prediction of its surrounding words, which encourage the network to capture inter-word dependencies. The same intuition can be transferred to the computer vision domain, where SSL models are learning to predict masked patches of an image or representation (Grill et al. 2020; He, Xinlei Chen, et al. 2022a). An alternative SSL objective is to encourage different views of the same image, typically generated through random data augmentation techniques, to be mapped to similar locations in the **representation space** (T. Chen et al. 2020; Xinlei Chen and He 2021). The model is pre-trained on the aforementioned pretext tasks, which allow for the learning of useful representations (stored in encoder's frozen weights) that can in turn be applied to various downstream tasks.

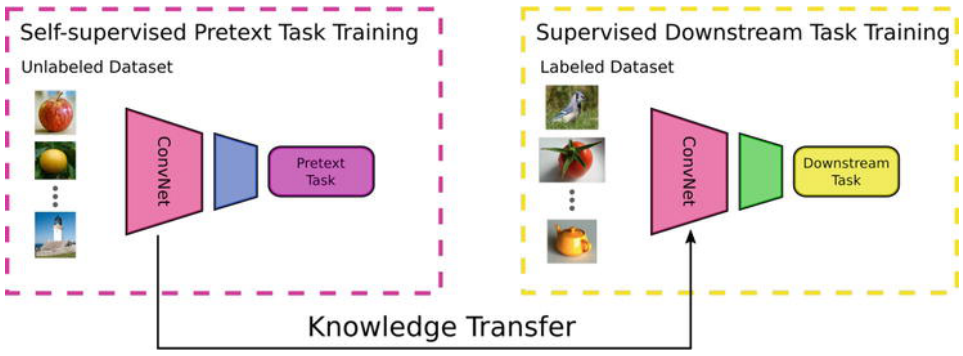


Figure 5.1: General pipeline of SSL. The knowledge (representations, feature extractor) learned from a pretext task is transferred to downstream tasks. Figure courtesy of Bastanlar and Orhan 2022.

SSL methods can be applied on vast corpora of unlabeled data, which comes with many benefits, such as learning generic representations, applicable to multiple tasks. In contrast, supervised learning methods are trained on an *"a priori"* specified task based on the availability of labeled data, and their learned features can be too rigid to adapt to novel tasks. SSL methods have also gained popularity in domains where labels are naturally scarce or costly to obtain, such as medicine, or when the target task cannot be known "a priori" (Krishnan, Rajpurkar, and Topol 2022; Ciga, T. Xu, and Martel n.d.), whereas their learned representations have been shown to be more robust to label corruption and input perturbations (Hendrycks et al. 2019; Goyal, Duval, et al. 2022) compared to supervised features.

A high-level overview of the SSL pipeline is given in Figure 5.1. Visual representations (CNN feature extractor) are learned through self-supervision by solving a **pretext task** on an unlabeled dataset and are subsequently **transferred** and adapted to other downstream computer vision tasks (e.g., image classification) through supervised **fine-tuning**. The performance of **downstream tasks** is highly dependent on the quality of the SSL learned representations; therefore, SSL can also be considered as a form of **representation learning**. In representation learning we are not interested in the pretext performance, but rather in the quality of the learned representations. Ideally, the representation space should include semantic and structural information, e.g., embeddings from the same latent class should correspond to similar areas of the embedding space, while being separable from embeddings of different classes, as shown in Figure 5.2.

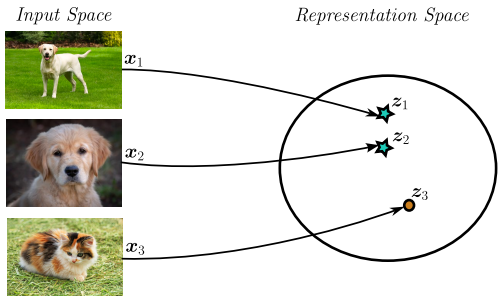


Figure 5.2: Given three distinct images x_1, x_2 and x_3, the network should ideally learn to place the dogs closer together (z_1 and z_2) in the representation space and the cat (z_3) further away.

5.1. Contrastive SSL

The principle of encouraging similarity between semantically transformed versions of input constitutes the basis of **contrastive learning** in the context of SSL. This principle was first turned into a learning objective by the contrastive loss, introduced in (Bromley et al. 1993) and formally defined in (Chopra, Hadsell, and LeCun 2005; Hadsell, Chopra, and LeCun 2006). In contrastive learning, the goal is to train a network to be able to predict whether two inputs belong to the same class or not and accordingly adjust their embeddings to be closer or farther from each other. In the SSL setting, known **semantic-preserving** data transformations are applied to identify similar inputs, since we do not have access to ground-truth labels. For each input image, the variants produced through the transformations are called **positives** and all remaining images and their variations are considered **negatives**. One of the most prominent methods coming from the contrastive paradigm is **SimCLR** (T. Chen et al. 2020), which we will also use as an archetype to dive deeper into contrastive learning.

5.1.1. SimCLR

The fundamental idea behind SimCLR is rather elegant and intuitive, **encouraging the similarity between two augmented views of an image**. First, a minibatch of N random images is sampled. The process then involves forming two positive views (x_i, x_i^+) of an image by applying a combination of random image augmentations, such as random cropping, resizing, color jittering *etc.*, resulting in a total of $2N$ input images. Figure 5.3 showcases some of the typical data augmentations applied by SimCLR and other contrastive methods. Negative samples are not sampled explicitly. Instead, given a positive pair (x_i, x_i^+), the rest $2(N-1)$ samples are treated as negatives. The learned representations are more robust and generalizable since the network is encouraged to learn **augmentation invariance**.

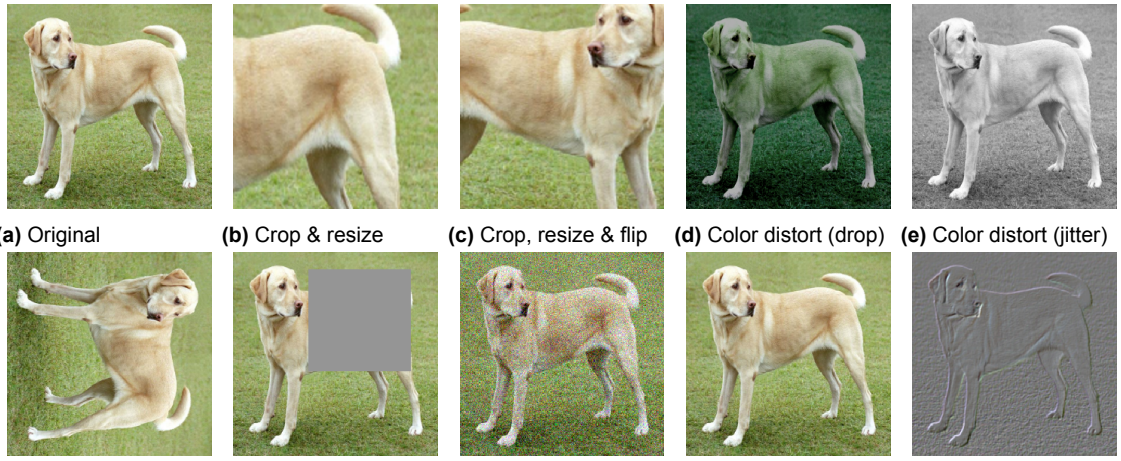


Figure 5.3: Illustrations of data augmentation operators, commonly applied in contrastive SSL. Figure courtesy of T. Chen et al. 2020.

All input images are then encoded by a CNN encoder $h_i = f(x_i)$, followed by an MLP projection head $z_i = g(h_i)$, which maps the output of the encoder to a representation space, where the contrastive loss is applied, as shown in Figure 5.4. Finally, given an input image, the objective of the contrastive loss is to bring its embedding closer to that of its positive pair in the representation space, while repelling the representations of all negative samples, as shown in Figure 5.5. The contrastive loss that SimCLR leverages, is called **NT-Xent** (normalized temperature-scaled cross-entropy loss) (T. Chen et al. 2020) and is a variant of the fundamental **InfoNCE** loss (Oord, Li, and Vinyals 2018). Equation (5.1) defines the loss for a positive pair (i, i^+), with the final loss being the average of all positive pairs in a batch.

$$\ell(i, i^+) = -\log \frac{\exp\left(\mathrm{d}\left[\boldsymbol{z}_i, \boldsymbol{z}_i^+\right]/\tau\right)}{\sum_{j=1}^{2N} \mathbb{1}_{[j \neq i]} \exp\left(\mathrm{d}\left[\boldsymbol{z}_i, \boldsymbol{z}_j\right]/\tau\right)}, \tag{5.1}$$

where τ the temperature scaling factor and d a distance metric, in this case the negative cosine similarity:

$$\mathrm{d}\left[\boldsymbol{z}_i, \boldsymbol{z}_j\right] = \frac{-\boldsymbol{z}_i^\top \boldsymbol{z}_j}{(\|\boldsymbol{z}_i\| \|\boldsymbol{z}_j\|)} \tag{5.2}$$

A key component of Equation (5.1) is the **non-parametric softmax**, initially introduced by Wu et al. 2018, which has been adopted by most contrastive methods in the literature. This name is motivated by removing the need to have a *"parametrized"* linear layer on top of the representation to compute the softmax operator, but instead directly comparing representations with each other.

Finally, the temperature parameter τ controls the *"sharpness"* of the output probability distributions, with higher values of τ leading to *"softer"* or less confident predictions (e.g., $[0.2, 0.2, 0.6]$) and $\tau \to 0$ leading to *"sharper"* predictions (e.g., $[0.01, 0.01, 0.98]$). In practice, lower temperature values can help the network avoid a potential **representation collapse**. A representation collapse occurs when the network learns a *"shortcut"*,

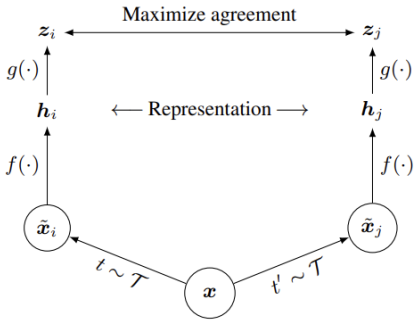


Figure 5.4: A simple framework for contrastive learning of visual representations. Figure courtesy of T. Chen et al. 2020.

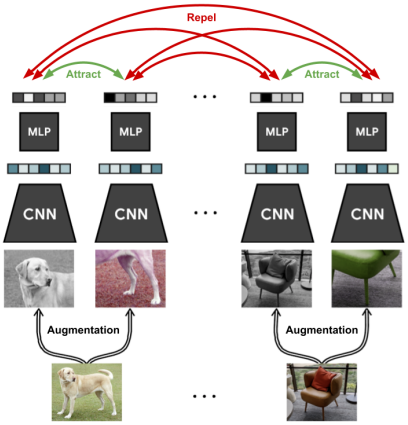


Figure 5.5: The two positive views should be close to each other in the embedding space while being repelled from the representations of other augmented images.

i.e., manages to get incredibly low values for the contrastive loss, by simply outputting the same representation, regardless of the input. Nevertheless, such representations are redundant for any downstream task, which requires the learned features to be discriminative and hold semantic information.

5.1.2. NNCLR

Another popular contrastive learning method is NNCLR (Nearest-Neighbor Contrastive Learning of Visual Representations) (Dwibedi et al. 2021), which builds upon SimCLR, but also introduces a **memory queue** of past representations, as a means of **sampling positives of higher quality**. In particular, despite the profound success of contrastive learning in SSL, these approaches strictly rely on data augmentations for positive generation, which cannot cover all the variance in a given class (e.g., object deformations, different viewpoints). Even more importantly, traditional contrastive learning tries to enforce consistency only at the **instance-level** (each image in a batch corresponds to a unique class), rather than the **class-level**. The possibility of additional positives being present within a batch (e.g., different cat images) is not only overlooked, but additionally such samples are treated as negatives. Consequently, the network could learn different representations for images belonging to the same latent class. NNCLR attempts to alleviate these problems by going **beyond single-instance** positives.

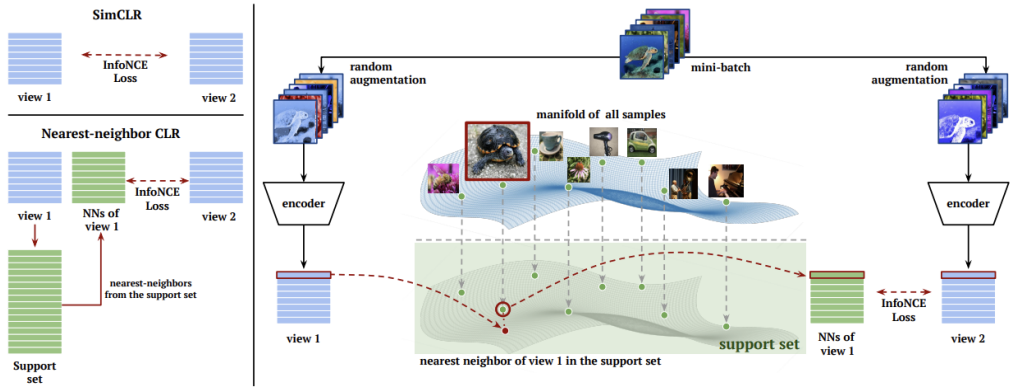


Figure 5.6: Overview of NNCLR training. Figure courtesy of Dwibedi et al. 2021.

As in SimCLR, for each image in a batch we generate two views through random data augmentations, which are then passed through the network to get the embeddings z_i on which the contrastive loss operates. In contrast to SimCLR however, NNCLR enforces consistency (in the contrastive loss) between the **nearest neighbor** of z_i and z_i^+, that is:

$$\ell(i, i^+) = -\log \frac{\exp\left(\mathrm{d}\left[\mathrm{NN}(z_i), z_i^+\right]/\tau\right)}{\sum_{j=1}^{2N} \mathbb{1}_{[j \neq i]} \exp\left(\mathrm{d}\left[\mathrm{NN}(z_i), z_j\right]/\tau\right)}, \tag{5.3}$$

where $\mathrm{NN}(z_i)$ denotes the nearest neighbour of z_i from the memory. The memory is implemented as a **FIFO** (fist in first out) queue of past representations, similarly to He, Fan, et al. 2020, which tries to capture the global distribution of the training set. Figure 5.6 shows an overview of the NNCLR architecture. The nearest neighbor of an image can be an image of the same latent class, which, however, does not necessarily correspond to the same original instance. The positive pairs used by NNCLR have been shown to be more useful to the network, since they incorporate information that a contrastive loss typically ignores. NNCLR led to performance improvements, compared to traditional contrastive SSL approaches, when applied on computer vision benchmarks such as ImageNet.

5.2. Self-Distillation SSL

The working principle of **self-distillation methods** relies on a rather straightforward mechanism. This involves, similarly to contrastive methods, the generation of two different views. However, these two views are fed into two separate encoders, often referred to as **student** and **teacher**. The objective is then to map the student encoder's output to that of the teacher, by relying on a prediction head and typically an asymmetry in the architecture. This is achieved by applying a contrastive loss between the

teacher's projected representations and the student's predicted representations. Various techniques have been developed to prevent representation collapse, with one common approach involving updating the teacher's weights with a running average of the student's weights. This **EMA** (exponential moving average) or **self-distillation** encoder scheme can help maintain diversity in the representations learned by the encoders and promote more meaningful learning. By updating the encoders in this manner, self-distillation methods can achieve better performance, compared to their pure contrastive counterparts, and generate more diverse and informative embeddings for downstream tasks. Two of the most prominent self-distillation SSL approaches are **BYOL** (Grill et al. 2020) and **SimSiam** (Xinlei Chen and He 2021), which will be used to dig deeper into the mechanics of self-distillation.

5.2.1. BYOL

BYOL (Bootstrap Your Own Latent) was the first method to introduce self-distillation as a means of preventing representation collapse. In BYOL, two encoder networks are used together with a predictor to map the output of one encoder to the other. The network responsible for predicting the output is known as the *online* or *student* network, while the network producing the target output is referred to as the *target* or *teacher* network. Different views of the same image, which are created through various image transformations, similarly to those shown on Figure 5.3, including random resizing, cropping, color jittering, and brightness alterations, are given as input to each network.

As the training progresses, the **student** network is updated using **stochastic gradient descent**, just as in normal deep learning training. However, normal SGD training is not applied for the teacher network. Instead of using direct gradient updates, the weights of the **teacher** network are updated with an **exponential moving average (EMA)** of the weights of the student network: $\xi \leftarrow m * \xi + (1 - m) * \theta$, where θ, ξ are the parameters of the student and the teacher, respectively, and m the momentum parameter that controls to what degree the teacher network preserves its history. The slow updates induced by EMA create an **asymmetry** that is crucial to the success of BYOL. A high-level overview of BYOL's architecture is shown in Figure 5.7. The combination of self-distillation and the use of image augmentations leads to a powerful learning mechanism that helps the student network produce meaningful and diverse representations without collapsing to trivial solutions. This approach has proven to be effective in various self-supervised learning tasks and has led to state-of-the-art results in the field.

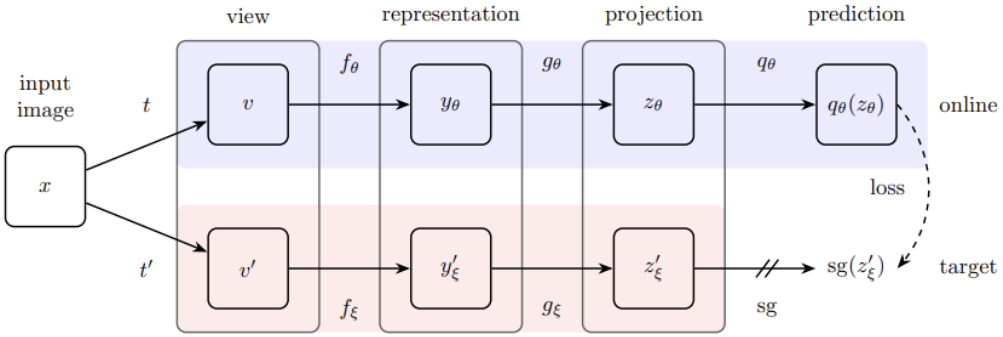


Figure 5.7: BYOL's architecture. Figure courtesy of Grill et al. 2020.

Before elaborating on BYOL's loss function, let us take a closer look on InfoNCE (Oord, Li, and Vinyals 2018), which has been the basis of most modern contrastive loss functions:

$$\mathcal{L}_{iNCE} = -\frac{1}{2N}\sum_{i=1}^{2N}\log\frac{\exp\left(\mathrm{d}\left[\boldsymbol{z}_i,\ \boldsymbol{z}_i^+\right]/\tau\right)}{\sum_{j\neq i}\exp\left(\mathrm{d}\left[\boldsymbol{z}_i,\ \boldsymbol{z}_j\right]/\tau\right)} = \underbrace{-\frac{1}{N}\sum_{i=1}^{N}\mathrm{d}\left[\boldsymbol{z}_i,\ \boldsymbol{z}_i^+\right]/\tau}_{\mathcal{L}_{pos}} + \underbrace{\frac{1}{2N}\sum_{i=1}^{2N}\log\sum_{j\neq i}\exp\left(\mathrm{d}\left[\boldsymbol{z}_i,\ \boldsymbol{z}_j\right]/\tau\right)}_{\mathcal{L}_{neg}} \tag{5.4}$$

InfoNCE can be decomposed into two distinct terms: (i) $\mathcal{L}_{pos}$, only applied between positive pairs $\boldsymbol{z}_i$ and $\boldsymbol{z}_i^+$ to bring their representations together, and (ii) $\mathcal{L}_{neg}$, which pushes each representation away from all other batch representations and has a regularizing effect (preventing representation collapse). BYOL discards the negative term $\mathcal{L}_{neg}$ all together, relying on the self-distillation mechanism to prevent representations from collapsing:

$$\mathcal{L}_{BYOL} = \frac{1}{2N}\sum_{i=1}^{N} \mathrm{d}\left[\boldsymbol{p}_i,\, \boldsymbol{z}_i^+\right] + \mathrm{d}\left[\boldsymbol{z}_i,\, \boldsymbol{p}_i^+\right], \tag{5.5}$$

where $\boldsymbol{p} = q_\theta(g_\theta(f_\theta(\boldsymbol{x})))$ the student's predicted representations and $\boldsymbol{z} = g_\xi(f_\xi(\boldsymbol{x}))$ the teacher's projected representations. The distance metric d used by BYOL is the euclidean distance and the embeddings are l_2 normalized before its application.

5.2.2. SimSiam

The main objective of SimSiam (Xinlei Chen and He 2021) was to understand which components of BYOL are essential for preventing representations from collapsing. It should be noted that contrastive SSL avoided trivial representations by relying on **negative samples**, whereas BYOL relied on **momentum encoder**. SimSiam showed that it is possible to prevent representations from collapsing in **Siamese** (same architecture for student and teacher) networks without relying on any of these approaches. It showed that BYOL's EMA is not necessary in practice, albeit it leads to a small boost in performance. SimSiam shares the weights between the student and the teacher, with a **stop-gradient** operation being applied on the teacher path. An overview of SimSiam's architecture is shown in Figure 5.8. The loss function of SimSiam is very similar to that of BYOL and is described as:

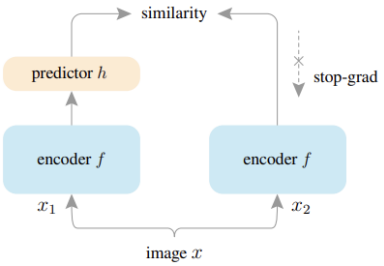


Figure 5.8: SimSiam's simple Siamese architecture. Figure courtesy of Xinlei Chen and He 2021.

$$\mathcal{L}_{BYOL} = -\frac{1}{2N}\sum_{i=1}^{N} \mathrm{sim}\left[\boldsymbol{p}_i,\, \boldsymbol{z}_i^+\right] + \mathrm{sim}\left[\boldsymbol{z}_i,\, \boldsymbol{p}_i^+\right], \tag{5.6}$$

where $\mathrm{sim}\left[\boldsymbol{z}_i, \boldsymbol{z}_j\right] = \frac{-\boldsymbol{z}_i^\top \boldsymbol{z}_j}{(\|\boldsymbol{z}_i\|\|\boldsymbol{z}_j\|)}$ the cosine similarity, which replaces the euclidean distance of BYOL. SimSiam acted as a simple baseline, showing that the Siamese architecture can naturally introduce inductive biases to model **invariance**. In fact, Xinlei Chen and He 2021 argue that similar to how convolutions can model translation invariance via weight-sharing, weight-sharing Siamese networks can model invariance with respect to more complicated data transformations, *, i.e.,* data augmentations.

BECLR adopts the asymmetrical architecture and EMA encoder of BYOL. This is combined with a less biased version of SimSiam's contrastive loss, which also includes a negative loss term. Finally, similar to NNCLR, BECLR exploits past representations to directly modify the contrastive objective and introduce more useful positives in a training batch. However, instead of replacing the teacher's embeddings with that of their nearest neighbours and using a simplistic FIFO memory queue, we leverage a class-cognizant dynamic clustered memory (DyCE). DyCE is iteratively updated with *equipartitioned* batch assignments, converging into a *highly-separable* paritioned latent space and enabling the meaningful positive sampling strategy and batch enhancement of BECLR. This allows BECLR to distill both *instance-* and *class-level* insights within a contrastive learning framework.

5.3. Masked Image Modeling

In the early stages of self-supervised pretraining for computer vision, various prominent approaches applied some form of **degradation to input images** to encourage more robust representations, such as decolorization (R. Zhang, Isola, and Efros 2016), Gaussian noise (Vincent et al. 2008), or patch shuffling (Noroozi and Favaro 2016). Another such technique was called context encoders (Pathak et al. 2016), where large parts of an image were masked out and replaced with white pixels. Then an auto-encoder was trained to inpaint the missing parts. However, this early attempt at **masked**

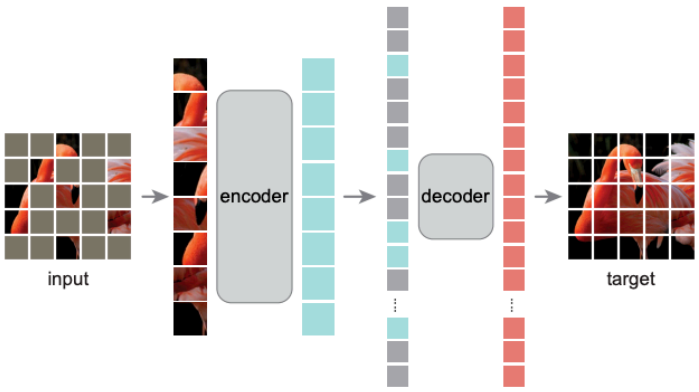


Figure 5.9: High-level overview of masked image modelling.

image modeling (MIM) failed to achieve competitive performance compared to supervised learning in downstream tasks. Figure 5.9 provides a high-level abstraction of MIM.

More recently, BERT (Devlin et al. 2018) revolutionized the NLP world by introducing **masked language modeling (MLM)**, which involved placing text tokens in a transformer language model with learnable mask tokens and training the model to recover the original text. MLM can be considered a form of SSL, which relies on input degradation (masking), and remains extremely popular for training state-of-the-art large language models.

Following the success of MLM, Dosovitskiy et al. 2020 introduced the **Vision Transformer** in the context of MIM and computer vision. They masked-out patches of an image and trained the model to directly predict pixel values, yet found this pretraining strategy ineffective compared to its supervised counterparts. It turns out that directly applying the BERT strategy to images can be challenging, since image patches are eligible for a vastly higher number of possible values than words. Instead, BEiT (Bao et al. 2021) treats MIM as a regression problem, using an autoencoder to map image patches to discrete tokens and then pretraining a vision transformer to predict the discrete token values for masked tokens. BEiT achieved significant downstream task improvement compared to both supervised and self-supervised approaches, yet its performance is dependent on a powerful autoencoder for the creation of the discrete tokens. More recent approaches (He, Xinlei Chen, et al. 2022b; Xie et al. 2022) directly reconstruct masked image patches instead of using discrete image tokens extracted from an encoder, as in BEiT, advancing the downstream performance on several tasks and showcasing that MIM can be an effective SSL technique for computer vision. Finally, some of the most recent and successful approaches to representation learning **iBOT** (J. Zhou et al. 2021) and **DINOV2** (Oquab et al. 2023) employ a mix of MIM and more classical SSL approaches, such as self-distillation. For instance, in iBOT the MIM reconstruction objective is applied in latent space, with the teacher network providing the reconstruction targets, instead of the original image. The MIM objective is combined with a contrastive loss based on self-distillation, as shown on Figure 5.10.

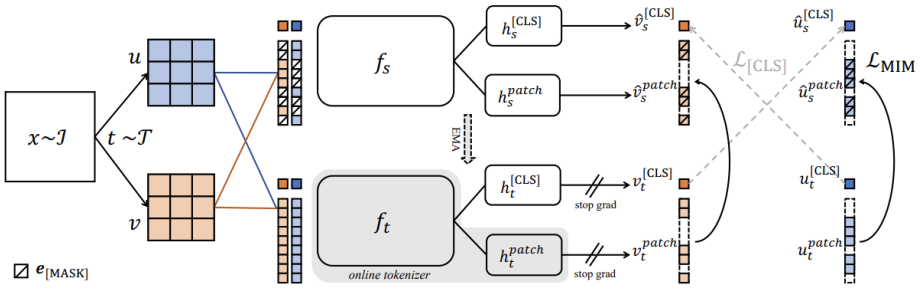


Figure 5.10: Overview of iBOT framework. Figure courtesy of J. Zhou et al. 2021.

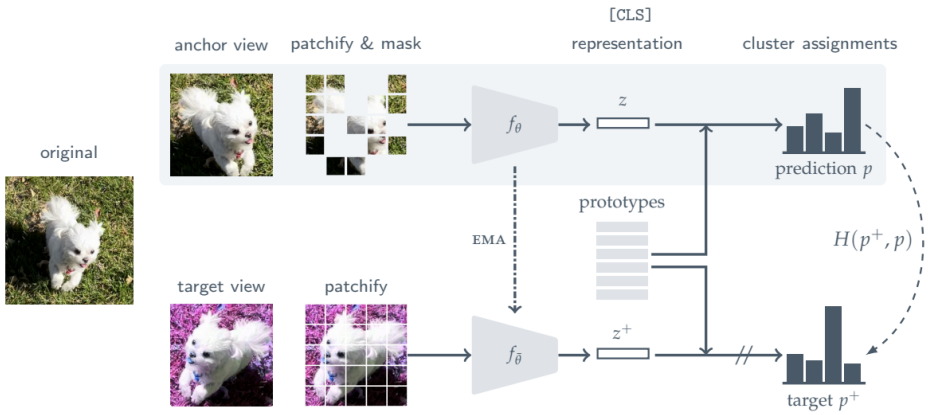


Figure 5.11: Overview of MSN framework. Figure courtesy of Assran, Caron, Misra, Bojanowski, Bordes, et al. 2022.

5.3.1. Masked Siamese Networks

Another powerful SSL approach, which relies on masking, is **Masked Siamese Networks (MSN)** (Assran, Caron, Misra, Bojanowski, Bordes, et al. 2022), which further pushed the state-of-the-art on standard computer vision benchmarks, along with showing a promising performance in extreme low-shot settings (e.g., using only $1-5$ images per class on an ImageNet downstream task). MSN's fundamental idea is to apply mask denoising while avoiding pixel- and token-level reconstruction, but rather relying on self-distillation and a contrastive loss variant. A schematic of the method is illustrated in Figure 5.11. The architecture is similar to that of self-distillation SSL methods, with the distinction that a patch-wise masking step is applied on the anchor (or student) view, while the target (or teacher) view remains unmasked. The objective is for the vision transformer (ViT) encoders to output similar representations for both views. MSN does not directly predict the masked patches, but rather performs an implicit denoising step at the representation level, by making the masked view's predicted prototype assignments consistent with those of the unmasked view. Another advantage of MSN is its computational scaling, since only the unmasked patches are passed through the ViT network, which is even more evident with more aggressive masking ratios.

Unlike standard contrastive approaches, which apply the loss directly to the student and teacher representations, in MSN learning occurs by computing a **soft distribution** (can be viewed as a soft assignment problem) **over a set of stored prototypes** for both views, much like clustering-based SSL approaches (Caron, Misra, et al. 2020; Assran, Caron, Misra, Bojanowski, Joulin, et al. 2021). These assignments of anchor and target representations are calculated using the time-efficient **Sinkhorn-Knopp** algorithm (Cuturi 2013), which solves a relaxed variant of the **Optimal Transport (OT)** problem (Cuturi 2013; Peyré, Cuturi, et al. 2019). Finally, the objective is to bring the assignments of the representations of masked anchor and unmasked target views as close as possible. Several masked anchor views are generated for each image, and the loss is aggregated over all of them.

Given $K > 1$ learnable prototypes $\boldsymbol{q} \in \mathbb{R}^{K \times d}$ and a masked anchor representation $z_{i,m}$, MSN finds an optimal transport plan or *"prediction"* $p_{i,m}$ in the K-dimensional simplex by measuring the cosine similarity to the prototype matrix:

$$p_{i,m} = \text{softmax}(\frac{z_{i,m} \cdot \boldsymbol{q}}{\tau}) \tag{5.7}$$

A standard cross-entropy loss $H(p_i^+, p_{i,m})$ as in Equation (3.8) between the student $p_{i,m}$ and teacher predictions p_i^+ is used to penalize different predictions. Additionally, the mean entropy maximization (ME-MAX) regularizer is utilized to encourage the use of the entire set of prototypes, as in (Joulin and Bach 2012; Assran, Caron, Misra, Bojanowski, Joulin, et al. 2021). By aggregating over the number of

masked views and the number of images in the batch, we get the final loss:

$$\mathcal{L}_{MSN} = \frac{1}{MB} \sum_{i=1}^{B} \sum_{m=1}^{M} H(p_i^+, p_{i,m}) - \lambda H(\bar{p}), \tag{5.8}$$

where $\bar{p} = \frac{1}{MB} \sum_{i=1}^{B} \sum_{m=1}^{M} p_{i,m}$ and λ controls the effect of the ME-MAX regularization. Finally, gradients are computer only for the anchor network, with the target network's parameters being updated with an EMA of the anchor network's parameters, much like the self-distillation approaches.

Similarly to MSN, `BECLR` applies patch-wise masking on the student path and relies on the contrastive loss to enforce an implicit mask denoising step. The motivation for MIM is similar to that of data augmentations in that by making the training task more difficult, the network is encouraged to learn better representations, which are robust to input degradation (like masking).

6

Few-Shot Learning

Deep learning has been crucial for advancing major progress in various scientific fields, including computer vision, speech recognition, natural language processing, and more. However, despite these breakthroughs, deep learning and **artificial intelligence (AI)** in general are largely dependent on the availability of vast amounts of data to allow models to learn meaningful features and generalizations. In stark contrast, **human intelligence** allows learning new abstract concepts or even skills with minimal "data" examples. For example, children would be able to easily recognize cats and distinguish them from other animals after only encountering them a few times. Similarly, humans are capable of detecting patterns and similarities between objects (analogous to how deep networks learn discriminative features) with just one or a few examples and are adept at inferring relationships and abstractions. The motivation behind **few-shot learning (FSL)** is to bridge this gap between AI and human intelligence. In this setting, the model is trained to be able to generalize to new classes and tasks when provided with only a few samples per class. In the context of this work, we focus on the specific few-shot learning instance of **image classification**, where the downstream task involves the prediction of class labels for an unlabeled dataset (query set) based on a rather small labeled dataset (support set) with only a few samples (typically 1 to 5) per evaluated class.

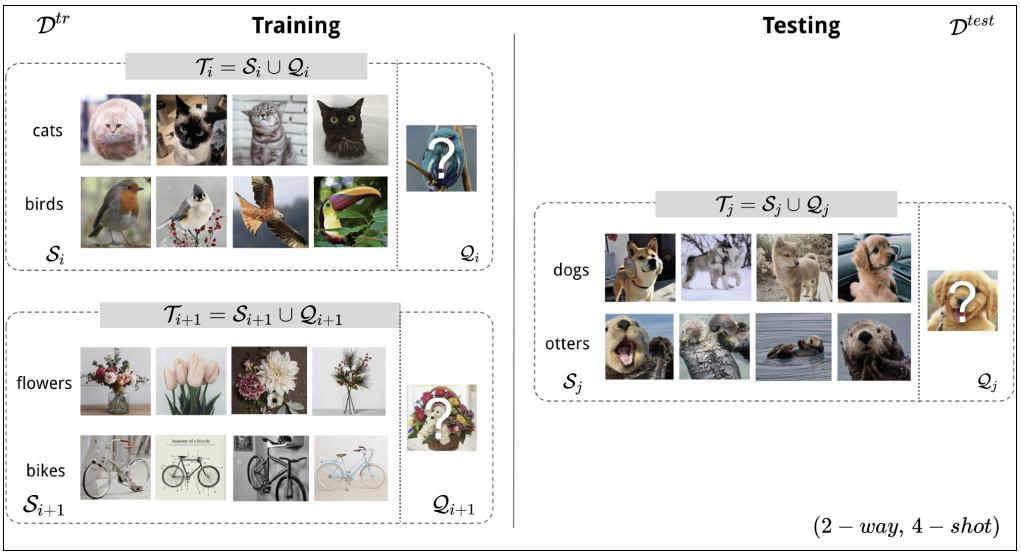


Figure 6.1: Illustration of a (2-way, 4-shot) episodic setting. Each dotted box corresponds to a task or episode $\mathcal{T}_i$, which is comprised of a labelled support $\mathcal{S}_i$ and unlabelled query $\mathcal{Q}_i$ set. Meta learning approaches sample multiple tasks during training, instead of larger conventional batches. During testing, typically 600-2000 tasks are sampled for an unbiased estimate of the performance, but only 1 task $\mathcal{T}_j$ is shown for the sake of simplicity.

Our approach follows the standard practice of few-shot image classification approaches, which consists of two sequential phases: (i) **pretraining** on a large dataset of base classes, followed by (ii) simple **inference** or **fine-tuning** on an unseen smaller dataset, consisting of novel classes. The goal of the pretraining phase is to learn a global feature extractor (i.e., backbone network or encoder) from the base dataset, followed by simply fitting a classifier (inference) or learning a trainable classification layer (supervised fine-tuning) on top of the frozen feature extractor to adopt to the novel classes. The base and novel classes are either mutually exclusive classes of the same dataset (**in-domain**) or originate from different datasets (**cross-domain**). In this paper we tackle the more challenging **unsupervised few-shot learning (U-FSL)** problem, which means that we do not have access to the actual class labels for the base dataset. Hence, we adopt a **self-supervised** (also interchangeably called "unsupervised" in the literature) pretraining, followed by a simple supervised inference phase.

Two primary approaches have emerged to address the few-shot learning problem.

Meta Learning. In this approach, the algorithm is designed to *"learn how to learn"*, with the goal of finding a model that can quickly adapt and generalize to new tasks or classes. Meta learning was one of the first approaches that successfully tackled the challenging FSL task, with various methods falling under this category (Finn, Abbeel, and Levine 2017; Hsu, Levine, and Finn 2018; Antoniou and Storkey 2019; Ji et al. 2019; Khodadadeh, Boloni, and Shah 2019; D. B. Lee et al. 2020; Ye, Han, and Zhan 2022). Meta learning is based on an **episodic** training process, which involves creating synthetic *"tasks"* (or episodes) that mimic the downstream, also episodic, fine-tuning phase. By repeatedly exposing the model to a variety of episodes, it learns to extract generalized patterns and representations.

Transfer Learning. In this approach, the model is initially trained **non-episodically** to learn optimal representations, which can be generalized to novel tasks with minimal updates, during the pretraining phase from a large base dataset. The pre-trained feature extractor, usually trained using a form of **metric learning** (e.g., contrastive learning), captures the underlying structure of the unlabeled data. Then, during the episodic fine-tuning or inference phase, a simple predictor, often a linear layer or classifier, is combined with the pre-trained feature extractor for rapid adaptation to novel classes. The better the feature extractor captures the global structure of the unlabeled data, the easier it will be for the predictor to adapt to novel unseen classes in the testing phase. Several methods have adopted the transfer learning approach to FSL (Dhillon et al. 2019; Medina, Devos, and Grossglauser 2020; Yonglong Tian et al. 2020; W. Chen et al. 2021; Kishorkumar Shirekar, A. Singh, and Jamali-Rad 2022; Lu et al. 2022).

Despite the popularity of both approaches, several recent studies (Dhillon et al. 2019; Medina, Devos, and Grossglauser 2020; Yonglong Tian et al. 2020; Laenen and Bertinetto 2021; Lu et al. 2022) have questioned the suitability of meta-learning on the few-shot classification task and showed that episodic meta-learning training is data-inefficient, failing to adequately exploit the training batch. These studies have shown competitive performance on FSL benchmarks, without using episodic pretraining, but rather following the transfer learning approach. Notably, in the U-FSL setting, the current state-of-the-art (Lu et al. 2022) on the two most popular few-shot classification benchmarks: miniImageNet (Vinyals et al. 2016) and tieredImageNet (Ren et al. 2018) adopts a **self-distillation-based self-supervised pretraining**, inspired from SimSiam (Xinlei Chen and He 2021), and conclusively demonstrates transfer learning's superiority to meta learning in terms of downstream classification performance. Our approach `BECLR` also follows the transfer learning paradigm and manages to further advance the state-of-the-art performance on both benchmarks in the U-FSL regime.

6.1. Problem Formulation

Let us consider a large labeled dataset $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{M}$ of images $\boldsymbol{x_i}$, labels y_i and of size M. This dataset is divided into three disjoint datasets $\left\{\mathcal{D}^{\text{tr}} \cup \mathcal{D}^{\text{val}} \cup \mathcal{D}^{\text{tst}}\right\} \in \mathcal{D}$, referring to the training, validation and test subsets, respectively. The validation set is typically used for model selection, and the test set is for evaluation. The **base** classes contained in the training set are mutually exclusive to the **novel** classes contained in the validation and test sets. The training set $\mathcal{D}^{\text{tr}} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{M^{\text{tr}}}$ is composed of a set of randomly sampled images in the case of **transfer learning**, or a set of randomly sampled tasks $\mathcal{T}_i$ in the case of **meta learning**. Model selection and testing are performed by randomly sampling

tasks $\mathcal{T}_j$ from $\mathcal{D}^{\text{val}}$ and $\mathcal{D}^{\text{tst}}$, respectively. A task $\mathcal{T}_i$ consists of two parts: (i) the support set $\mathcal{S}$, from which the model learns, and (ii) the query set $\mathcal{Q}$, on which the model is evaluated. The support set $\mathcal{S} = \{\boldsymbol{x}_i^s, y_i^s\}_{i=1}^{NK}$ contains K labeled random samples from N different classes, while Q unlabeled samples for each class comprise the query set $\mathcal{Q} = \{\boldsymbol{x}_j^q\}_{j=1}^{NQ}$. In a N-way, K-shot task (also referred to as episode), the number of labeled samples K is relatively small (e.g., 1 or 5). By convention, we denote a few-shot task as: $\mathcal{T}_i = \mathcal{S} \cup \mathcal{Q}$ with (N-way, K-shot). Figure 6.1 shows some (2-way, 4-shot) training and testing tasks.

6.2. Model Agnostic Meta Learning

Meta-learning (Schmidhuber 1987), often referred to as "learning-to-learn", involves improving a learning algorithm by leveraging multiple learning episodes, distinguishing it from conventional machine learning, which utilizes multiple data instances. Meta-learning is comprised of (i) an **inner learning algorithm**, which solves a task (e.g., image classification), and (ii) an **outer algorithm**, which updates the inner parameters to improve a broader and more general objective.

Model-agnostic meta-learning (MAML) (Finn, Abbeel, and Levine 2017) constitutes one of the fundamental few-shot meta-learning-based methods, aiming to improve the learning strategy over multiple training episodes. Its main objective is to find an initial set of model weights that can be **quickly adapted to new tasks** with just a few gradient descent steps. The sensitivity of the loss function to the parameters of new tasks θ_i' is maximized, allowing small changes in the parameters to lead to significant changes in the loss of the task. This allows quicker adaptation to new tasks. As shown in Figure 6.2 given a neural network f_θ to be optimized for FSL, the objective is to find an optimal set of parameters, capable of fast generalization (with only a few steps of gradient descent). Figure 6.3 describes the entire training process, which involves performing a few gradient descent steps on the support set of each randomly sampled task (within a metabatch), resulting in the individual inner updated parameters θ_i' for each task. Finally, the meta-update step calculates the final loss over all the corresponding query sets using their respective inner-updated parameters. The inner updates along with the meta-update comprise one iteration of the MAML algorithm, with multiple iterations carried out until the outer parameters have converged to an optimal value: $\theta \rightarrow \theta^*$.

MAML describes a rather straightforward approach, which has its merits in that it can be applied on any deep learning model and no additional parameters need to be introduced for meta-learning. Additionally, it's optimization is based on the thoroughly applied gradient descent, without the need of more complicated optimization algorithms, and it has found applications on a plethora of domains from image classification to reinforcement learning. Nevertheless, meta-learning has been shown to be **data inefficient** and can become **computationally intensive** because of its utilization of higher-order gradients.

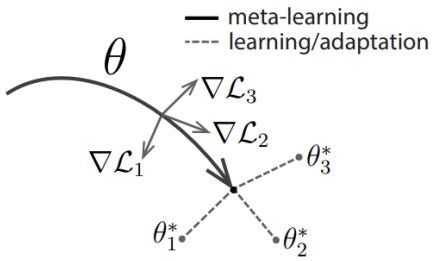


Figure 6.2: MAML optimizes for model parameters θ, capable of quickly adapting to novel tasks. Figure courtesy of Finn, Abbeel, and Levine 2017.

Algorithm 2 MAML for Few-Shot Supervised Learning

Require: $p(\mathcal{T})$: distribution over tasks
Require: α, β: step size hyperparameters
1: randomly initialize θ
2: **while** not done **do**
3: Sample batch of tasks $\mathcal{T}_i \sim p(\mathcal{T})$
4: **for all** $\mathcal{T}_i$ **do**
5: Sample K datapoints $\mathcal{D} = \{\mathbf{x}^{(j)}, \mathbf{y}^{(j)}\}$ from $\mathcal{T}_i$
6: Evaluate $\nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$ using $\mathcal{D}$ and $\mathcal{L}_{\mathcal{T}_i}$ in Equation (2) or (3)
7: Compute adapted parameters with gradient descent: $\theta_i' = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$
8: Sample datapoints $\mathcal{D}_i' = \{\mathbf{x}^{(j)}, \mathbf{y}^{(j)}\}$ from $\mathcal{T}_i$ for the meta-update
9: **end for**
10: Update $\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i'})$ using each $\mathcal{D}_i'$ and $\mathcal{L}_{\mathcal{T}_i}$ in Equation 2 or 3
11: **end while**

Figure 6.3: MAML algorithm for FSL. The support set for each training task is used to compute the inner updates θ', with the query set being evaluated on the inner updated parameters. The loss applied on the task query sets is used for meta-updating θ, through backpropagation. Figure courtesy of Finn, Abbeel, and Levine 2017.

6.3. Prototypical Networks

Prototypical networks (protonets) (Snell, Swersky, and Zemel 2017) constitute another fundamental approach to tackle FSL, which, unlike MAML, belongs to the **metric learning** approaches. The objective of metric learning is to learn a **representation function** capable of mapping objects to a latent embedding space, which is suitable for classification. The latent space should satisfy certain properties, like **preserving relations between input images**, i.e., similar/dissimilar inputs should be closer/farther in the embedding space. In order for proto-nets to avoid overfitting due to the small number of samples and ensure better generalization on novel tasks, the authors argue that the classifier should have a small inductive bias. The objective is to learn a representation function that clusters all inputs of the same class around their single-class prototype and farther from prototypes of other classes in the episode.

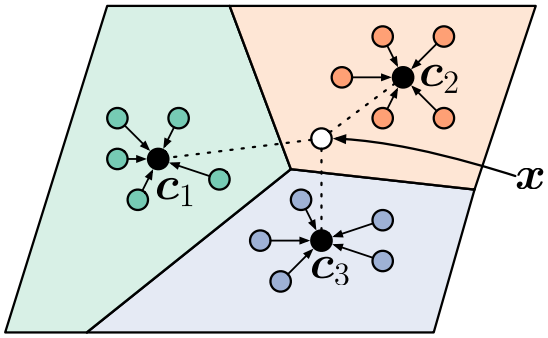


Figure 6.4: Few-shot prototypes c_n are computed as the mean of embedded support examples for each class. Figure courtesy of Snell, Swersky, and Zemel 2017.

A neural network encoder f_θ acts as the representation function mapping the input images to latent space feature vectors, and is trained under the supervised setting. Then, since training is performed episodically, given a (N-way, K-shot) episode, the K embeddings for each of the N classes are averaged to compute the class prototypes. The prototypes for each class $n \in \mathcal{N}$ are supposed to represent the entire class and are defined as the mean feature vector of the embeddings of the support set:

$$\boldsymbol{c}_n = \frac{1}{|\mathcal{S}_n|} \sum_{(\boldsymbol{x}_i, y_i) \in \mathcal{S}_n} f_{\boldsymbol{\theta}}(\boldsymbol{x}_i). \tag{6.1}$$

Given a query input, a differentiable distance metric (in this case euclidean distance) is applied between the query embedding and the class prototype vectors, followed by a softmax in order to calculate the distribution over the support classes:

$$P(y = n|\boldsymbol{x}) = \text{softmax}\left(-\,\text{d}(f_{\boldsymbol{\theta}}(\boldsymbol{x}), \boldsymbol{c}_n)\right) = \frac{\exp(-\,\text{d}(f_{\boldsymbol{\theta}}(\boldsymbol{x}), \boldsymbol{c}_n))}{\sum_{n' \in \mathcal{N}} \exp(-\,\text{d}(f_{\boldsymbol{\theta}}(\boldsymbol{x}), \boldsymbol{c}_{n'}))} \tag{6.2}$$

Finally, the network parameters are updated through gradient descent by minimizing the negative log likelihood $J(\boldsymbol{\theta}) = -\log P_{\boldsymbol{\theta}}(y = k|\boldsymbol{x})$ of the class-conditional probabilities $P(y = n|\boldsymbol{x})$. It should be noted that since the prototypes correspond to the mean of each class, and Euclidean distances are utilized to compute the probability distributions, this method is essentially equivalent to applying a **nearest-mean** linear classifier on the learned representations $f_{\boldsymbol{\theta}}(\boldsymbol{x})$, as shown on Figure 6.4.

6.4. Unsupervised Few-Shot Learning

Until now we have only discussed supervised few-shot learning approaches, which rely on the existence of labels in the training set. Nevertheless, our method addresses the more challenging **unsupervised few-shot learning** objective. Formally, the training set $\mathcal{D}^{\text{tr}} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{M^{\text{tr}}}$ that we defined in Section 6.1 is now replaced by $\mathcal{D}^{\text{tr}} = \{(\boldsymbol{x}_i)\}_{i=1}^{M^{\text{tr}}}$ without access to ground truth labels y. However, most supervised FSL methods adopt an **episodic** training scheme, which in turn depends on the existence of labels to create valid training episodes. This challenge sheds more light on the superiority of **transfer learning** methods, which typically rely on **self-supervised** pre-text tasks and self-distillation, over

meta-learning approaches, which depend on episode generation for training. In the following sections we will further elaborate on some unsupervised few-shot learning methods from earlier approaches, which are based on their MAML and protonets supervised counterparts, to more competitive methods, which apply self-supervised based pretraining and are much closer to our approach.

6.4.1. Unsupervised Meta Learning

CACTUs. One of the first approaches to unsupervised few-shot classification was CACTUs (Hsu, Levine, and Finn 2018), which tries to fully incorporate the training scheme and the MAML objective but adapt it to the unsupervised setting. Hence, it retains the episodic training of MAML. However, in the absence of labels, CACTUs introduces a clustering-based **task creation** process for generating the episodes needed for MAML's meta-training.

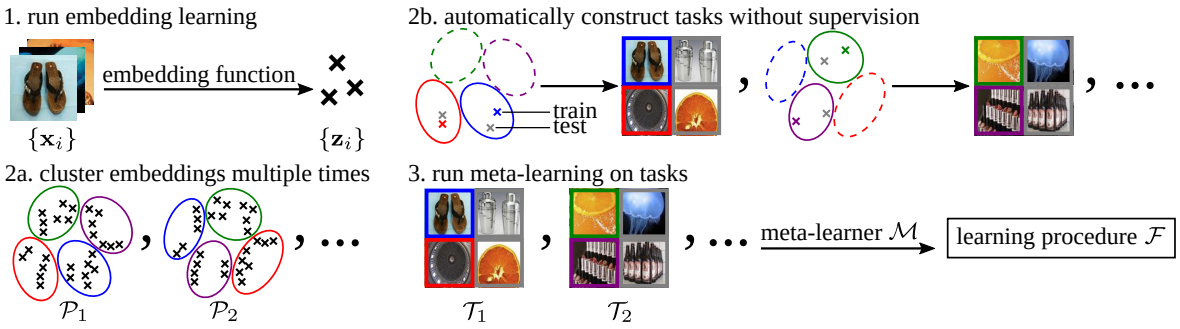


Figure 6.5: Illustration the meta-learning procedure of CACTUs. Figure courtesy of Hsu, Levine, and Finn 2018.

The first step of CACTUs involves the separate training of a representation learning algorithm $\boldsymbol{z}_i = f(\boldsymbol{x}_i)$ on the training set. Hsu, Levine, and Finn 2018 opt for utilizing DeepCluster (Caron, Bojanowski, et al. 2018), BiGAN (Berthelot et al. 2018), ACAI (Donahue, Krähenbühl, and Darrell 2016), and InfoGAN (Xi Chen et al. 2016) for this step, although in practice any self-supervised method can be combined with the CACTUs pipeline. The next step applies multiple iterations of **k-means clustering** (P times) to generate a set of partitions, from which meta learning episodes can be sampled. Subsequently, CACTUs first samples a random partition P, before sampling a random cluster of K elements from the chosen partition P. The whole sampling process is repeated until N clusters have been chosen to form the support set of a (N way, K shot) episode, which in turn can be used in MAML. Similarly, N clusters of Q elements are sampled for the MAML outer update. Figure 6.5 illustrates the entire CACTUs training pipeline. The quality of the representation learning algorithm used in the first step influences the quality of the k-means clustering, which is used to sample the meta-learning episodes.

A major limitation of CACTUs is that its performance depends on the first step of separate representation learning, which, in fact, employs deeper models (e.g., AlexNet (Krizhevsky, Sutskever, and Hinton 2012)) on smaller few-shot benchmarks like miniImageNet (Vinyals et al. 2016). Additionally, several surveys (He, X. Zhang, et al. 2016; Dosovitskiy et al. 2020) have empirically shown that deeper neural networks are capable of learning much better representations than shallower models in most scenarios. Consequently, in the case of CACTUs, a deeper model is used to guide the meta-learning training of a shallower (typically Conv4) model, which can be considered a form of knowledge distillation. However, this approach defeats the purpose of a self-contained end-to-end unsupervised few-shot learning algorithm.

UMTRA. Another early unsupervised approach, which again relies on the inner and outer updates of meta-learning, is unsupervised meta-learning with tasks constructed by random sampling and augmentation (UMTRA) (Khodadadeh, Boloni, and Shah 2019). In contrast to CACTUs, neither does UMTRA depend on a deeper model nor focuses on generating "optimal" episodes for MAML. Instead, like its name suggests, it relies on augmentations of the unlabeled input images to create "pseudo-labels" and in turn create a valid query set.

Figure 6.6 illustrates the complete UMTRA training process. The first step involves the sampling

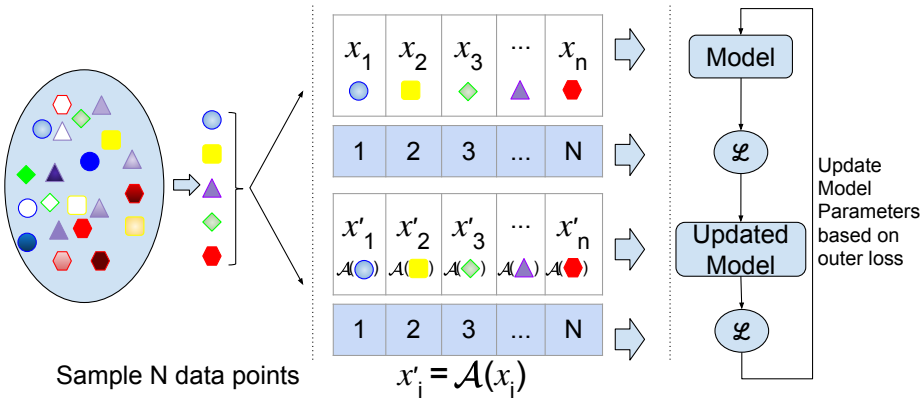


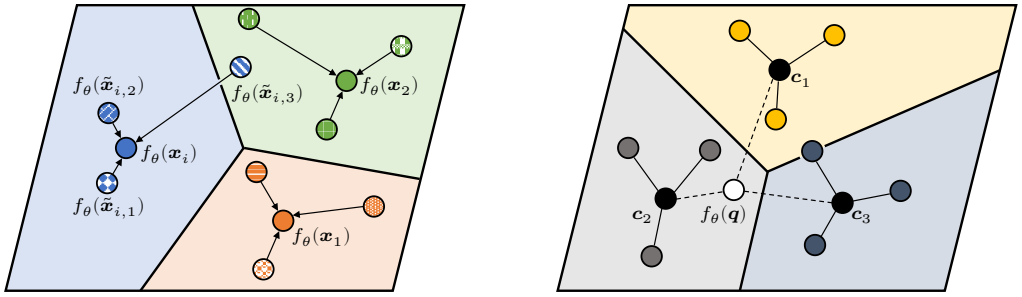
Figure 6.6: Illustration of UMTRA training process. N data points are sampled to create the support set for the inner update, whereas the query set (for the meta update) is formed by applying augmentation function $\mathcal{A}$ on each of the N training images. Figure courtesy of Khodadadeh, Boloni, and Shah 2019.

of N images from the training set $\mathcal{D}^{\text{tr}}$, which are used for the inner update of MAML. Subsequently, an augmentation function is applied to each of the training images $\boldsymbol{x}_i$ to generate query images $\boldsymbol{x}_i' = \mathcal{A}(\boldsymbol{x}_i)$, which are utilized by the MAML outer update. The working principle behind UMTRA is that data augmentations are **class-preserving transformations**, hence the support and query sets, and by extension the inner and outer objectives, contain the same classes, despite the lack of ground-truth labels. Nevertheless, it should be noted that UMTRA makes the latent assumption that in the absence of labels, each image sampled from the training set, along with its augmentations, corresponds to a unique class. Hence, UMTRA operates on the **instance-level**, since each image instance is treated as a unique class. As a result, images within a batch that could correspond to the same "latent" class (e.g., different images of cats) are treated as distinct classes, which could make the network erroneously learn different distributions for such images. `BECLR` alleviates this problem, which is ubiquitous in unsupervised few-shot learning approaches, by ingraining **class-level** information along the default **instance-level** in a contrastive loss.

6.4.2. ProtoTransfer

Despite their success in supervised few-shot learning, meta-learning approaches, such as CACTUs and UMTRA, have been shown to be inefficient in the unsupervised case, as we discussed in the introduction of Chapter 6. Instead, **transfer learning** has been proven to be the most competitive approach for U-FSL. Transfer learning methods methods typically consist of two sequential phases to solve the U-FSL problem: (i) pretraining on a large unlabeled dataset of base classes, followed by (ii) simple inference or fine-tuning on an unseen smaller dataset, consisting of novel classes. The goal of the pretraining phase is to learn a global feature extractor (i.e., backbone network or encoder) from the base dataset, which is leverages standard self-supervised training instead of an episodic training scheme. Finally, the supervised inference or fine-tuning phase involves fitting a classifier (inference) or learning a trainable classification layer (supervised fine-tuning) on top of the frozen feature extractor to adopt to the novel classes. In fact, `BECLR` also adopts this popular training paradigm and is able to outperform meta-learning approaches by a significant margin in both **in-domain** (base and novel classes come from the same dataset but are mutually exclusive) and **cross-domain** (base and novel classes come from different datasets altogether) settings.

One of the first methods, which demonstrated the effectiveness of transfer learning in the U-FSL regime, was ProtoTransfer (Medina, Devos, and Grossglauser 2020), which extends the idea of proto-nets in the absence of ground-truth labels. ProtoTransfer does not rely on a task-based episodic scheme, but instead adopts a contrastive SSL approach, inspired by SimCLR, for its pretraining phase, followed by supervised fine-tuning of a linear layer (MLP). ProtoTransfer's pretraining modules is aptly denoted **ProtoCLR**, since its pretraining loss can be interpreted as a self-supervised version of the prototypical loss (Snell, Swersky, and Zemel 2017) in line with contrastive learning. In ProtoCLR, the original image $\boldsymbol{x}_i$ is augmented Q times and the network is trained through contrastive loss, which



(a) Self-Supervised Prototypical Pretraining (b) Prototypical Fine-Tuning & Inference

Figure 6.7: Self-Supervised Prototypical Transfer Learning. (a): In the embedding space, the original images $\boldsymbol{x}_i$ serve as class prototypes around which their Q augmented views $\tilde{\boldsymbol{x}}_{i,q}$ should cluster. (b): The prototypes $\boldsymbol{c}_k$ are the means of embedded support examples for each class n and initialize a final linear layer for fine-tuning. An embedded query point $\boldsymbol{q}$ is classified via a softmax over the fine-tuned linear layer. Figure courtesy of Medina, Devos, and Grossglauser 2020.

enforces consistency between the original image (acts as the prototype) and all its augmentations.

It should be noted that in the supervised case, proto-nets utilized the support set to calculate the prototypes and were evaluated on the query set. In the unsupervised setting, similar to UMTRA, ProtoTransfer treats each sample instance as its own class. Hence, a training batch of n samples $\{\boldsymbol{x}\}_{i=1\ldots n}$ acts as a 1-shot support set with each sample corresponding to a class prototype. Each prototype $\boldsymbol{x}_i$ is then randomly augmented Q times to form the query samples $\tilde{\boldsymbol{x}}_{i,q}$, on which the loss is evaluated. Consequently, ProtoTransfer is also susceptible to UMTRA's **instance-level** assumption.

The second phase of ProtoTransfer involves a prototypical supervised fine-tuning denoted as **ProtoTune**. In this second stage, ProtoTransfer is evaluated on FSL downstream tasks, which include a labeled support set of few samples per class, from which the network needs to learn to adapt to the novel classes, and an unlabeled query set on which the method is evaluated. ProtoTransfer is based on the linear layer property of protonets, as described in Section 6.3, to initialize a linear classification layer from the support set of each test episode. Then, random subsets of the support set are used for fine-tuning (using a standard cross-entropy loss) the linear layer for some iterations. Finally, the representations of the query set are passed through the encoder network f_θ and the linear classification layer to obtain the final predictions. The entire pipeline of ProtoTransfer is illustrated on Figure 6.7.

6.4.3. PDA-Net

Prototypical contrastive approaches like ProtoTransfer are quite intuitive and able to outperform metalearning-based approaches. However, the state-of-the-art performance in U-FSL is achieved from different approaches. These approaches also follow the **transfer learning** paradigm, which consists of pretraining on an unlabeled base dataset and fine-tuning or evaluating on few-shot tasks. However, instead of relying on protonets, these methods adopt a **self-supervised distillation-based** approach, typically applied on general representation learning, for their pretraining phase, with the objective of attaining the best possible representations from the first phase of transfer learning. The motivation is that the better the feature extractor can capture the global structure and latent semantic relationships of the base dataset, the easier it would be for the classifier to adapt to novel few-shot tasks, when given only a few support samples.

PDA-Net (part discovery and augmentation network) (W. Chen et al. 2021) is one of the most competitive U-FSL approaches, which adopts the instance discrimination pre-text task of contrastive learning for its pretraining. PDA-Net features an asymmetrical architecture, along with an EMA updated teacher encoder, inspired by MoCo (He, Fan, et al. 2020). Its pretraining module is denoted as **PDN** (part discovery network) and tries to enforce consistency (via the contrastive loss) between an input

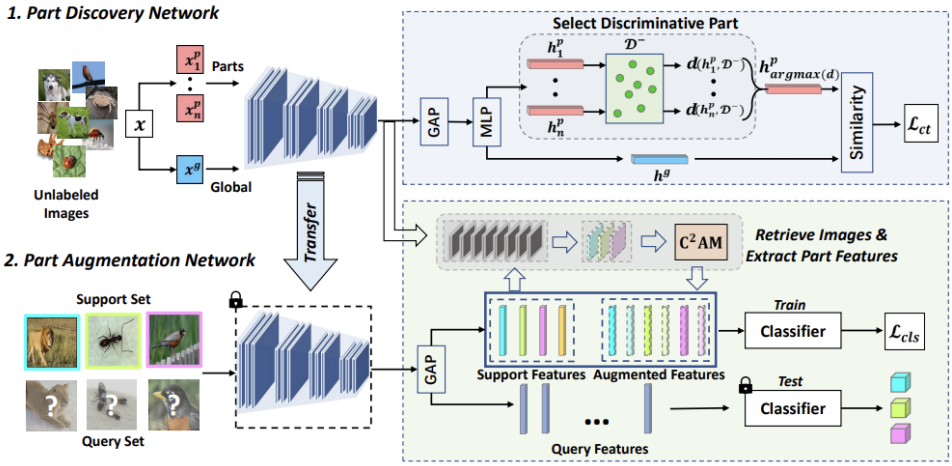


Figure 6.8: The PDA-Net framework. Figure courtesy of W. Chen et al. 2021.

image and its most discriminative part. First, each input image $\boldsymbol{x}$, is cropped into n patches $\{\boldsymbol{x}_i^p\}_{i=1}^n$, which in turn are augmented through random transformations, along with a larger global crop $\boldsymbol{x}^g$. The global and patch views are passed through the CNN encoder network f_θ to acquire global $\boldsymbol{h}^g$ and part $\{\boldsymbol{h}_i^p\}_{i=1}^n$ representations. The next step involves the selection of the **most discriminative part** from $\{\boldsymbol{h}_i^p\}_{i=1}^n$, which is the part representation with the largest average distance from the global views of all other images in a batch $\mathcal{D}^- = \{\boldsymbol{h}_j^g\}_{j=1}^{N^-}$:

$$\boldsymbol{h}^p = \boldsymbol{h}_{i^*}^p, \;\; i^* = \underset{a}{\operatorname{argmax}} \left(\mathrm{d}\left[\boldsymbol{h}_i^p, \; \mathcal{D}^-\right] \right), \tag{6.3}$$

where $\mathrm{d}\left[\boldsymbol{h}_i^p, \; \mathcal{D}^-\right]$ the mean distance between each part image $\boldsymbol{h}_i^p$ and $\mathcal{D}^-$:

$$\mathrm{d}\left[\boldsymbol{h}_i^p, \; \mathcal{D}^-\right] = \frac{1}{|\mathcal{D}^-|} \sum_{\boldsymbol{h}^{g-}} - \mathrm{sim}(\boldsymbol{h}_i^p, \boldsymbol{h}^{g-}) \tag{6.4}$$

Once the most discriminative parts have been selected for each image in the training batch, the InfoNCE (Oord, Li, and Vinyals 2018) contrastive loss is applied to train the network to map the global view and the most discriminative view to similar embeddings in the latent space, as in Equation (5.4) with the distance metric corresponding to the negative cosine similarity.

The second phase of PDA-Net is denoted as **PAN** (part augmentation network), which attempts to augment the support set of testing episodes, by incorporating relevant part-based features from the base dataset (used during pretraining). First, given a test episode, its support set is used to train a linear classifier in a supervised manner. Subsequently, the trained classifier is applied to the entire base dataset, with the N_α images, corresponding to the most confident predictions for each of the k classes present in the support set, being kept. The feature maps of these N_α images for a particular class k are denoted as: $\mathcal{A}_k = \{\boldsymbol{M}_i^k\}_{i=1}^{N_\alpha}$. PAN then applies a class attention map (CAM) mechanism (B. Zhou et al. 2016), denoted as the class competitive attention map (C^{2}AM) as shown in Figure 6.9. C^{2}AM obtains attention maps $\boldsymbol{S}_k$ for all the feature maps $\boldsymbol{M}^k \in \mathcal{A}_k$, indicating their relevance for class k at each spatial location:

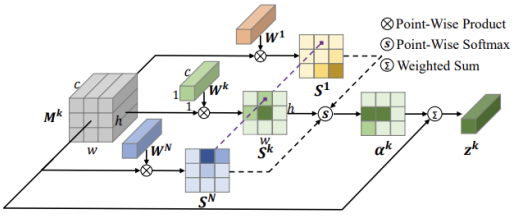


Figure 6.9: Illustration of C^{2}AM. Figure courtesy of W. Chen et al. 2021.

$$\boldsymbol{S}_k(i,j) = \boldsymbol{W}^k \boldsymbol{M}^k(i,j) + b^k, \tag{6.5}$$

where $\boldsymbol{W}^k$, b^k the learnt classifier's weight vector and bias and $\boldsymbol{S}_k(i,j)$, $\boldsymbol{M}^k(i,j)$ the classification score for class k and the feature vector at location (i, j), respectively. Subsequently, a softmax operator is applied on $\boldsymbol{S}_k(i,j)$ to highlight parts that have a higher confidence score only for class k and not for all classes, resulting in revised competitive attention maps $\boldsymbol{a}_k$. Then $\boldsymbol{a}_k$ is combined with the feature maps $\boldsymbol{M}^k$ to produce augmented support features:

$$\boldsymbol{z}^k = \frac{\sum_{i,j} \boldsymbol{a}_k(i,j)\boldsymbol{M}^k(i,j)}{\sum_{i,j} \boldsymbol{a}_k(i,j)} \tag{6.6}$$

Finally, the new features form the augmented support set $\mathcal{A} = \cup_k \mathcal{A}_k$, where $\mathcal{A}_k = \{\boldsymbol{z}_i^k\}_{i=1}^{N_\alpha}$. The augmented support set $\mathcal{A}$ and the original support set $\mathcal{S}$ are then combined to refine the classifier weights. Despite its competitive performance, PDA-Net adds a significant computational overhead since not only does the linear classification layer have to be trained twice for each testing episode, but more importantly, the entire (much larger) base dataset needs to be evaluated by the classifier in order to obtain the augmented support features of PAN. Additionally, this approach exploits information from the vastly larger base dataset, while evaluating on a much smaller few-shot task, which in practice defeats the purpose of a self-contained few-shot learning approach, that only uses the **few** support samples to adapt to the novel classes quickly.

6.4.4. UniSiam

UniSiam (Lu et al. 2022) constitutes another transfer learning approach, which applies self-supervision via self-distillation for its pretraining phase. In contrast to PDA-Net, UniSiam does not rely on elaborate techniques in its pretraining and fine-tuning stages, but instead focuses on representation quality. By adopting SimSiam's (Xinlei Chen and He 2021) asymmetrical architecture and stop-gradient operation on the teacher path, introducing even more aggressive data augmentations, and applying a less biased variant of the popular InfoNCE contrastive loss, UniSiam manages to reach state-of-the-art performance on the most popular U-FSL benchmarks, miniImageNet (Vinyals et al. 2016) and tieredImageNet (Ren et al. 2018). Lu et al. 2022 utilize SimSiam's intuitive architecture, as shown on Figure 6.10, and demonstrate that self-supervision can in fact be a very competitive few-shot learner.

Given a mini-batch of B random samples from the base dataset ($\mathcal{D}_{tr}$), two augmented views are produced from $\boldsymbol{X}$ by applying random image augmentations, which are then fed to the student and teacher networks to get the student's predictions $\boldsymbol{P} = q_\theta(g_\theta(f_\theta(\boldsymbol{X})))$ and teacher's projections $\boldsymbol{Z} = \text{SG}(g_\theta(f_\theta(\boldsymbol{Z})))$, respectively. Then, a contrastive loss is applied to enforce consistency between $\boldsymbol{P}$ and $\boldsymbol{Z}$. The objective of **InfoNCE** contrastive loss is to maximize the **Mutual Information (MI)** between the two augmented views of an image. Despite its popularity, however, recent studies (Poole et al. 2019; Song and Ermon 2019) have shown that it is prone to high bias, especially when the batch size is small and the MI is large. Instead, UniSiam introduces a different contrastive loss, which

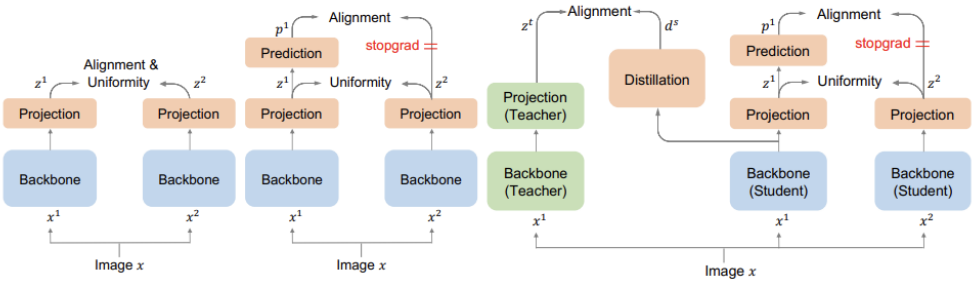


Figure 6.10: (a) SimCLR (T. Chen et al. 2020) architecture for comparison. (b) UniSiam for self-supervised pretraining. (c) UniSiam for self-supervised pretraining with knowledge distillation. Figure courtesy of Lu et al. 2022.

maximizes the same MI objective, but has been shown to be less biased for edge cases:

$$\mathcal{L}_{iMINE} = \underbrace{-\frac{1}{B}\sum_{i=1}^{B} \mathrm{d}\left[Z_i,\, Z_i^+\right]/\tau}_{Alignment} + \underbrace{\log\left(\frac{1}{2B}\sum_{i=1}^{2B}\sum_{j\in Neg(i)} \exp\left(\mathrm{d}\left[Z_i,\, Z_j\right]/\tau\right)\right)}_{Uniformity}, \tag{6.7}$$

where alignment and uniformity correspond to the positive and negative terms of InfoNCE (Equation (5.4)), respectively and d the negative cosine similarity. Thus, each embedding is l_2 normalized before applying a dot product to compute the distance. UniSiam also introduces an asymmetric alignment, which has been shown to produce better results (Xinlei Chen and He 2021) in combination with an asymmetric architecture (prediction only on the student path), yielding the final contrastive loss:

$$\mathcal{L} = \underbrace{-\frac{1}{2B}\sum_{i=1}^{B}\left(P_i \cdot Z_i^+ + P_i^+ \cdot Z_i\right)}_{Asymmetric\ Alignment} + \lambda \underbrace{\log\left(\frac{1}{2B}\sum_{i=1}^{2B}\sum_{j\in Neg(i)} \exp\left(Z_i \cdot Z_j/\tau\right)\right)}_{Uniformity}, \tag{6.8}$$

where λ a weighting hyperparameter and τ the temperature scaling factor. Finally, Lu et al. 2022 also incorporates a simple self-supervised knowledge distillation scheme, as shown in Figure 6.10, which utilizes pretrained deeper models (e.g., ResNet50) to improve the performance of shallower models (e.g., ResNet12).

UniSiam demonstrated that supervised pretraining and self-supervised pretraining try to optimize for different objectives in terms of mutual information, with the former maximizing mutual information between representations and labels ($I(\boldsymbol{Z}; Y)$) and the latter between the representations of 2 views ($I(\boldsymbol{Z}_1; \boldsymbol{Z}_2)$), which is a lower bound of mutual information between representations and data ($I(\boldsymbol{Z}; \boldsymbol{X})$). Hence, self-supervision aims to preserve the raw data information as much as possible in the learned representations, which in expectation would allow for better generalization to the novel classes. This is corroborated by Lu et al. 2022, who empirically demonstrate that self-supervised methods such as UniSiam can outperform supervised methods in the FSL regime, without relying on ground truth labels.

6.4.5. Connection to BECLR

UniSiam was chosen as the basis for our `BECLR` pretraining due to its intuitive architecture, robustness and focus on representation quality, with its novel loss function being adapted by our framework. However, UniSiam, like most U-FSL approaches, is susceptible to two main limitations:

- Despite the profound success of contrastive learning and self-distillation approaches, like PDA-Net and UniSiam, in U-FSL (current state-of-the-art), these approaches strictly rely on data augmentations for positive generation, which cannot cover all the variance in a given class (e.g., object deformations, different viewpoints). Even more importantly, traditional contrastive learning tries to enforce consistency only on at the **instance-level** (each image in a batch corresponds to a unique class), rather than the **class-level**, as it was briefly discussed in Section 6.4.1. The possibility of additional positives being present within a batch (e.g., different cat images) is not only overlooked, but additionally such samples are treated as negatives. Consequently, the network could learn different representations for images belonging to the same latent class.

- Despite a good feature extractor (learned in the pretraining phase) being essential for clustering unseen data, in the U-FSL setting there is a distribution shift between training and testing datasets, i.e., the base and novel classes are mutually exclusive. Downstream standard representation learning tasks, such as image classification, typically rely on supervised fine-tuning training schemes using novel classes to account for this distribution shift (Grill et al. 2020; He, Fan, et al. 2020). However, due to the intrinsic nature of FSL tasks, only a small number of samples from the support set are available, which are insufficient for adapting the network's weights, rendering fine-tuning redundant. Consequently, the final FSL performance is strongly correlated with the sample quality of the support data. In case the support set is not representative of the latent class distribution or contains outlier data, the learned classifier will be biased. This is known as the **sample bias** problem and is accentuated in lower-shot regimes (e.g., 1-shot).

- BECLR introduces a dynamici clustered memory (DyCE), which is used to draw more meaningful positive pairs and, as empirically shown, manages to capture the latent class distribution of the base dataset quite accurately thanks to novel equipartitioned memory updates, which leverage **optimal transport** (OT). DyCE is utilized to incorporate both a **class-level** and the default **instance-level** insights within a contrastive learning framework. Additionally, we introduce OpTA - a supervised inference strategy, which applies an OT-based feature alignment between the support prototypes and query set embeddings. OpTA successfully pushes the support set prototypes closer to the query set (and, by extension, ground-truth) class distributions. Consequently, it restricts the effects of **sampling bias** and results in a significant performance boost in downstream classification performance, especially in the 1-shot setting (where the sampling bias is most significant). Importantly, this performance boost cannot be achieved only by pre-training, rendering OpTA an efficient module, which could be applied on top of existing (not necessarily contrastive) U-FSL approaches. By tackling these two limitations, BECLR is able to push the state-of-the-art in U-FSL benchmarks by a significant margin, especially in the 1-shot setting (up to 14% increase in accuracy performance).

7

Optimal Transport

When faced with a decision-making problem, humans often opt for the **"shortest path"** approach, which involves the least amount of effort (or highest "reward") possible. Similarly, in statistics and machine learning problems, we are interested in finding a meaningful distance metric between probability distributions, capable of capturing the "shortest path" approach, while maintaining attractive qualities, such as **symmetry** and **triangle inequality**. Nevertheless, divergences (i.e., weaker "distance" notions often applied between probability distributions) more often than not fail to satisfy these properties. We will use the popular **Kullback-Lieibler (KL)** divergence (Kullback and Leibler 1951) as an example to illustrate the shortcomings of such methods, and how **Optimal Transport (OT)** constitutes a better alternative. The KL divergence is denoted as:

$$D_{KL}(P||Q) = \int p(x) log(\frac{p(x)}{q(x)})\, dx \tag{7.1}$$

where P, Q denote two probability distributions. Despite being one of the most useful and fundamental metrics in information theory, KL-divergence suffers from two main shortcomings: (i) not being a symmetric operation (i.e., $D_{KL}(P||Q) \neq D_{KL}(Q||P)$ and (ii) taking infinite values, when the support of P, Q do not match. Although the first limitation can be alleviated by a symmetric variant of the KL-divergence (Solomon Kullback 1997), the second one can be rather important, as sketched by Figure 7.1. In particular, the KL-divergence is infinite ($D_{KL}(P||Q) = +\infty$) for all 3 distribution pairs, denoting that they are equally dissimilar, which is rather counterintuitive. By smoothing the probability distributions so that their supports match, it is possible to alleviate the problem, but in certain cases it can be quite challenging to choose a suitable bandwidth parameter for the smoothing kernel.

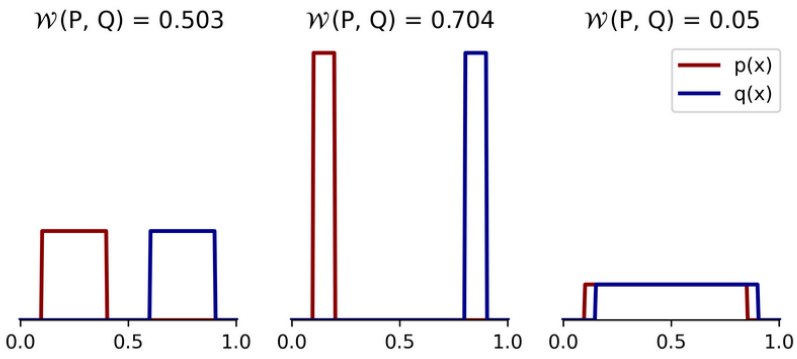


Figure 7.1: These density functions are infinitely far apart according to KL-divergence. On the contrary, their Wasserstein distance is finite and matches our intuition.

In contrast to KL-divergence, optimal transport can yield an alternative distance metric between distributions, which is not susceptible to either of the aforementioned limitations. In particular, OT defines an optimal **transport plan** (often denoted as **Wasserstein distance**, or even "Earth Mover's distance" in the literature) between two probability distributions, which is both symmetric and finite for distributions with different support. As shown in Figure 7.1, the Wasserstein distance is indeed finite in all cases, with its values matching our intuition: i.e., the middle and right panels correspond to the largest and smallest distance, respectively. Additionally, OT satisfies the triangular inequality, has no trainable parameters, maintains differentiability, and under certain conditions can ensure equipartitioned assignments. All these attractive qualities have made it rather popular in different pipelines, with applications in imaging (J. Lee, Bertrand, and Rozell 2020), generative deep networks (Adler and Lunz 2018), and even biological data analysis (Schiebinger et al. 2019).

`BECLR` leverages OT in both self-supervised **pretraining** (as part of the memory updating in `DyCE`) and supervised **inference** (for the feature alignment of `OpTA`) stages. During pretraining, we use OT to update our `DyCE` memory modules in a class-cognizant manner by finding optimal assignments between batch embeddings and memory prototypes, while ensuring equipartitioned assignments to the memory prototypes. The equipartitioning prevents the creation of dominant large clusters in the memory space, and along with the better encoder representations (as the training progresses) allows the memory embedding space to evolve from containing volatile and cluttered clusters (upon initialization) to comprising of highly separable ones. For the inference stage, given a test FSL task, we apply OT in order to "transport" the support set prototypes into the domain of the query set embeddings (denoted as the `OpTA` add-on). The transportation step essentially increases the spread of the support set prototypes in the query set's domain, reducing the distribution shift between support and query set and by extension the sampling bias problem. Subsequently, we simply fit a logistic classifier from the transported support prototypes, which is used to make the final predictions on the unlabelled query set.

7.1. Continuous Optimal Transport

Despite the fact that modern optimal transport is typically applied in statistics and machine learning, optimal transport theory was originally grounded in physical intuition. In particular, in the late 18^{th} century, Gaspard Monge[1] was the first to formulate the optimal transport problem in its simplest form, i.e., "how to transport soil between excavations with minimal transport expenses?". Let us consider a toy example of the Monge problem shown in Figure 7.2 to get a better intuition about OT. The objective is to find the most efficient transportation plan, which moves all the dirt from the dirt piles to fill all the available holes (if we consider the piles as prototypes and the holes as individual elements, it becomes clear that OT can act as an assignment or clustering algorithm). The piles and holes correspond to two probability distributions, and we assume that their total volume is equal.

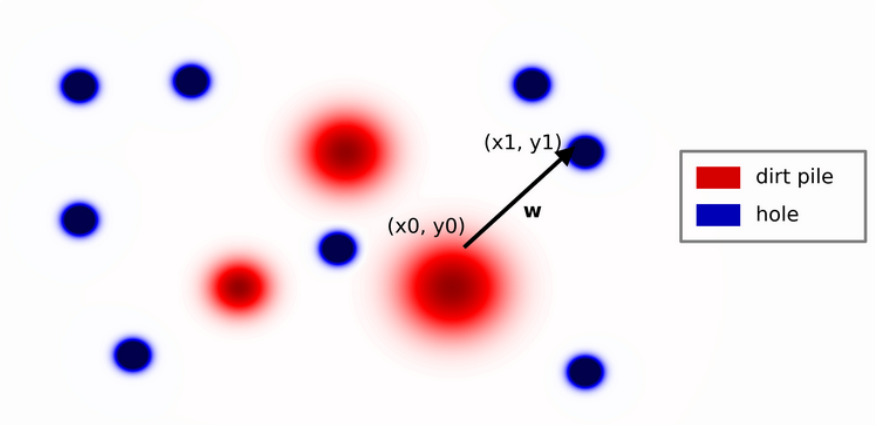


Figure 7.2: 2D optimal transport example. The dirt piles (red) need to fill all the holes (blue). The arrow schematizes w units of dirt being transported from location (x_0, y_0) to (x_1, y_1). A complete transport plan specifies transport paths like this over all pairs of locations.

[1] https://mathshistory.st-andrews.ac.uk/Biographies/Monge/

The first step involves defining the **transportation cost** of moving a single unit of dirt from (x_0, y_0) to (x_1, y_1). Equation (7.2) defines the transportation cost, when the Euclidean distance is used as a distance metric (although alternative metrics are also used in practice).

$$C(x_0, y_0, x_1, y_1) = (x_0 - x_1)^2 + (y_0 - y_1)^2 \tag{7.2}$$

Next we need to define the transportation plan , which tells us how many units of dirt to move from (x_0, y_0) to (x_1, y_1), for instance:

$$\pi(x_0, y_0, x_1, y_1) = w, \tag{7.3}$$

where the dirt pile (x_0, y_0) needs to contain at least w units of dirt, and the hole (x_0, y_0) needs to be able to fit at least w units of dirt. Only positive values are allowed for transportation plans. Additionally, we are allowed to transport dirt from a single pile to multiple holes, and a single hole can receive dirt from multiple piles. Finally, the transport plan needs to satisfy the following conditions:

$$\begin{aligned} \int\int \pi(x_0, y_0, x, y)dx\, dy &= r(x_0, y_0), \;\; \forall\, (x_0, y_0) \in dirt\ piles \\ \int\int \pi(x, y, x_1, y_1)dx\, dy &= c(x_1, y_1), \;\; \forall\, (x_1, y_1) \in holes, \end{aligned} \tag{7.4}$$

where r, c density functions encoding dirt volume. Intuitively, the first constraint ensures that all available dirt is transported for each pile, and the second constraint ensures that all holes are filled. Finally, given a transport plan π function, we can define the total cost to be minimized:

$$total\ cost = \int\int\int\int C(x_0, y_0, x_1, y_1)\pi(x_0, y_0, x_1, y_1)dx_0\, dy_0\, dx_1\, dy_1 \tag{7.5}$$

In practice, a transport plan π can be interpreted as a probability distribution. Specifically, if the two probability distributions are defined in some space $\mathcal{X}$, π can be viewed as a probability distribution defined in $\mathcal{X} \times \mathcal{X}$, where $\times$ is the Cartesian product.

7.2. Discrete Optimal Transport

Despite the intuitive aspect of the 2D toy example, continuous OT can be intractable for most real-world applications (Peyré, Cuturi, et al. 2019). Nevertheless, by discretizing the problem into atomic elements, we can compute reasonable estimates of optimal transport. `BECLR`, like most modern approaches, does in fact apply the discretized version of optimal transport; hence we will describe the full notation and mathematical formulations of discrete optimal transport.

7.2.1. Assignment Problem

In its simplest form, discrete OT can be considered as an assignment problem between sets, that is: "which is the best configuration among all possible configurations?" This simplest form of the problem is quite restrictive and really hard to solve, in that the **source** and **target** sets are required to be of exactly the same size. Each set can be represented as a histogram (or vector) $\boldsymbol{r}$, that belongs to the probability simplex - the components of the vector sum up to 1:

$$\boldsymbol{r} \in \left\{ x = (x_1, ..., x_N) \in \mathbb{R}^N : \sum_{i=1}^{N} x_i = 1 \right\} \tag{7.6}$$

We can denote $\mathbf{C}_{i,j}$ the cost of moving an element from i to j, then the total cost to be minimized is $\sum_{i=1}^{N} \mathbf{C}_{i,\sigma(i)}$, where σ denotes a **permutation** of the set $\{1, \dots, N\}$, and can be viewed as an assignment of the i^{th} bin of the source probability histogram r to the j^{th} bin of the target probability histogram c. In this setting, OT corresponds to a **combinatorial** problem, which can be summarized as:

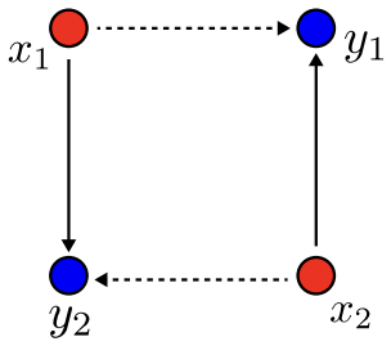


Figure 7.3: Non-unique assignments. The other solution is dashed. Figure courtesy of Peyré, Cuturi, et al. 2019.

How can we assign every element $i \in \{1, \dots, N\}$ to elements $j \in \{1, \dots, N\}$ in order to minimize $\sum_{i=1}^{N} \mathbf{C}_{i,\sigma(i)}$?

The result of this search is the **optimal assignment** or **Wasserstein distance**. It should be noted that this version of OT is very difficult to solve as N grows large, since there are $N!$ possible solutions and the optimal assignment is not unique, as illustrated in Figure 7.3

7.2.2. Working with Asymmetric Distributions

The restriction of equal-sized histograms is very strong and is not representative of real-world problems. Therefore, in this setting several points x_i can be mapped to the same y_i (much like the piles and holes in the Monge problem), but the conservation constraint of mass or volume still needs to be held.

In this case, the assignment between source and target histograms is no longer a permutation but rather a **surjective**[2] mapping T. If points $\{x_1, \dots, x_n\}$ have weights $\boldsymbol{r} = (r_1, \dots, r_n)$ and points $\{y_1, \dots, y_m\}$ have weights $\boldsymbol{c} = (c_1, \dots, c_m)$, π must verify:

$$\forall j \in \{1, \dots m\}, \quad c_j = \sum_{i:\pi(x_i)=y_j} r_i \tag{7.7}$$

This equation describes the mass conservation constraint. Even with this formulation, OT is not easier to solve since it remains an assignment problem.

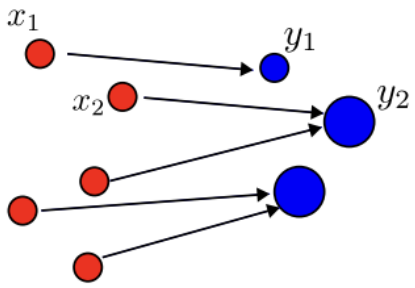


Figure 7.4: Optimal transport with asymmetric distributions. Figure courtesy of Peyré, Cuturi, et al. 2019.

7.2.3. The Kantorovich Relaxation

Another fundamental idea for modern OT was proposed in 1942 by Kantorovich 1942, who proposed the **relaxation**[3] of the deterministic aspect of transportation. In this relaxed variant of the initial assignment problem, the source points x_i are no longer required to map to a single target point y_i, that is, x_i can be fragmented into smaller pieces, which are distributed among multiple y_i. This property is known as **mass splitting**. This new formulation is much more suitable for real-world applications, for instance logistic problems, and was summarized by Hitchcock 1941 as:

When several factories supply a product to a number of cities, we seek the least costly method of distribution. Due to freight rates and other matters, the cost of a ton of product to a particular city will vary according to which factory supplies it, and will also vary from city to city.

In this setting our previous formulation needs to change, in that the permutation function σ is replaced by a coupling matrix $\boldsymbol{\pi} = \boldsymbol{\pi}_{ij} \in \mathbb{R}_{+}^{n \times m}$, which can be viewed as a possible transport plan and corresponds to an arrow from factory i to city j in Figure 7.5. The collection of all possible assignments can be denoted as:

$$\boldsymbol{\Pi}(\boldsymbol{r}, \boldsymbol{c}) = \left\{ \boldsymbol{\pi} \in \mathbb{R}_{+}^{n \times m} \mid \boldsymbol{\pi} \mathbf{1}_m = \boldsymbol{r}, \boldsymbol{\pi}^{\top} \mathbf{1}_n = \boldsymbol{c} \right\}, \tag{7.8}$$

where $\boldsymbol{r}$, $\boldsymbol{c}$ are the probability simplexes. $\boldsymbol{\Pi}(\boldsymbol{r}, \boldsymbol{c})$ contains all nonnegative $n \times m$ matrices for which all rows sum up to $\boldsymbol{r}$ and all columns sum up to $\boldsymbol{c}$ and is essentially a collection of **transport plans** (coupling matrix) of which some are better than others. This formulation is symmetric, i.e., if $\pi \in \boldsymbol{\Pi}(\boldsymbol{r}, \boldsymbol{c})$, then $\pi^T \in \boldsymbol{\Pi}(\boldsymbol{c}, \boldsymbol{r})$. Given a cost matrix $\boldsymbol{C}$, the cost of mapping $\boldsymbol{r}$ to $\boldsymbol{c}$ using a transport plan (or joint probability) $\boldsymbol{\pi}$ can be quantified as $\langle \boldsymbol{\pi}, \boldsymbol{C} \rangle_F$, we can now formulate the problem in a much cleaner fashion.

$$\boldsymbol{\pi}^{\star} = \min_{\pi \in \boldsymbol{\Pi}(r,c)} \sum_{i,j} \boldsymbol{C}_{i,j} \pi_{i,j} = \min_{\boldsymbol{\pi} \in \boldsymbol{\Pi}(r,c)} \langle \boldsymbol{\pi}, \boldsymbol{C} \rangle_F, \tag{7.9}$$

[2] A surjective function is a function f that maps an element x to every element y, i.e., there is a x such that $f(x) = y$.
[3] Relaxation denotes an easier to solve approximate problem in relation to the more difficult original problem.

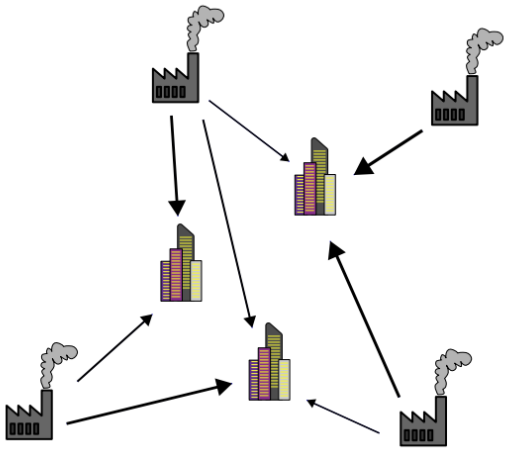


Figure 7.5: Factories with different supply capacities have to deliver goods to cities with various demands.

where $\boldsymbol{\pi}^\star$ is the optimal transport plan and $\langle \cdot, \cdot \rangle_F$ is the Frobenius dot product. When the cost matrix is based on a valid distance metric, the optimum $\boldsymbol{\pi}^\star$ is known as **Wasserstein distance**, and defines a distance metric between probability distributions. This formulation of OT can now be formulated as a **linear program**, where the constraints are a set of $m + n$ equality constraints.

7.2.4. Entropic Regularization

Despite the Kantorovich relaxation significantly reduced the computational requirements of OT, it is *still* not easy to solve, with most convex solvers (for linear programs) having polynomial complexities with exponents larger than 2, and sometimes exponential worst-case complexities. Further regularization of OT was proposed by Hitchcock 1941, who added an additional regularization term to the objective function, further relaxing the problem. First, we need to define the entropy of a coupling matrix $\boldsymbol{\pi}$:

$$\boldsymbol{H}(\boldsymbol{\pi}) = -\sum_{ij} \boldsymbol{\pi}_{ij} \log \boldsymbol{\pi}_{ij}. \tag{7.10}$$

According to information theory, the entropy of a random variable can be viewed as the level of "uncertainty" inherent in the possible outcomes of the variable. Hence, a matrix of assignments (such as $\boldsymbol{\pi}$) corresponding to **low entropy** values would be *sparser*, since its nonzero values would only be concentrated in a few points with **high confidence**. Conversely, a matrix with high entropy will be *smoother*, with the maximum entropy achieved with a uniform distribution of values between its elements. We can now formulate OT with entropic regularization:

$$\boldsymbol{\pi}^\star = \min_{\boldsymbol{\pi} \in \boldsymbol{\Pi}(r,c)} \langle \boldsymbol{\pi}, \boldsymbol{C} \rangle_F - \varepsilon \mathbb{H}(\boldsymbol{\pi}), \tag{7.11}$$

where ε the regularisation coefficient, which controls the level of regularization. When its value increases, the resulting coupling matrix will be smoother, and as $\varepsilon \to 0$ the coupling matrix will be sparser (or sharper) and the solution will be closer to that of the original relaxed OT problem in Equation (7.9). Figure 7.6 shows the effect of decreasing the regularization strength for a simple 1D optimal transport problem. The intuition behind entropy regularization is similar to the temperature-normalized cross-entropy (NT-Xent) discussed in Section 5.1.1. Finally, with the addition of the entropic regularisation, the optimal transport problem has become **convex**. Therefore, there is a unique optimal solution $\boldsymbol{\pi}^\star$ with the following form:

$$\forall (i,j) \in \{1, ..., n\} \times \{1, ..., m\}, \quad \boldsymbol{\pi}_{i,j} = \mathbf{u}_i \boldsymbol{K}_{i,j} \mathbf{v}_j, \quad \boldsymbol{\pi} = \operatorname{diag}(\mathbf{u}) \boldsymbol{K} \operatorname{diag}(\mathbf{v}), \tag{7.12}$$

where $\boldsymbol{K}_{i,j} = \exp\left(-\boldsymbol{C}_{i,j}/\varepsilon\right)$ calculated with $\boldsymbol{C}$ and $\mathbf{u}$ and $\mathbf{v}$ are unknown scaling variables. The formulation in Equation (7.12) is really important because now we have an **explicit formula** for an optimal transport plan.

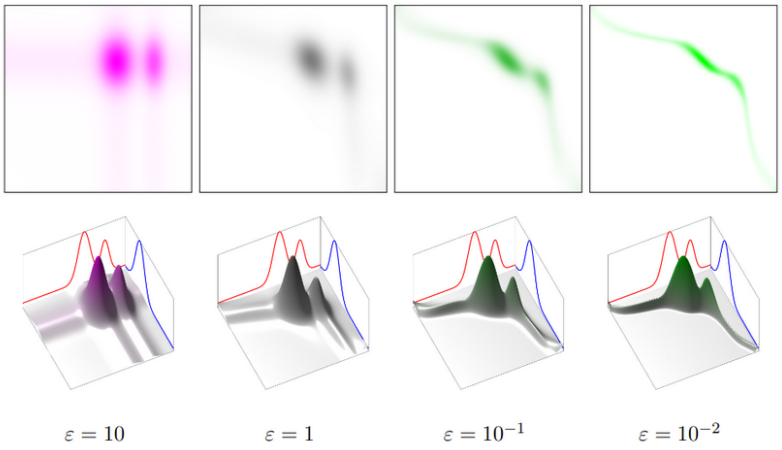


Figure 7.6: Effect of entropic regularization on optimal transport. Figure courtesy of Peyré, Cuturi, et al. 2019.

7.2.5. Sinkhorn-Knopp Algorithm

Equation (7.12) describes a **matrix scaling** problem, where the objective is to find two diagonal scaling matrices $\mathbf{u}$ and $\mathbf{v}$, which when multiplied by $\boldsymbol{K}$ give $\boldsymbol{\pi}$. It is important to note that $\boldsymbol{\pi}$ is a **doubly stochastic matrix**, and, by extension, the product $\operatorname{diag}(\mathbf{u})\boldsymbol{K}\operatorname{diag}(\mathbf{v})$ is also doubly stochastic. To compute these diagonal matrices, we utilize **Sinkhorn's algorithm** (Cuturi 2013), which involves sequentially updating $\mathbf{u}$ and $\mathbf{v}$ alternately through the following equations:

$$\mathbf{u}^{(k+1)} = \frac{\boldsymbol{r}}{\boldsymbol{K}\mathbf{v}^{(k)}} \tag{7.13}$$

$$\mathbf{v}^{(k+1)} = \frac{\boldsymbol{c}}{\boldsymbol{K}^{\top}\mathbf{u}^{(k+1)}} \tag{7.14}$$

This iterative method ensures that all rows and columns of $\boldsymbol{\pi}$ sum up to 1. Cuturi 2013 showed that not only does this iterative process converge, but does so at a linear rate. Sinkhorn's algorithm is the most popular algorithm for solving the entropic regularized optimal transport problem, hence the Wasserstein distance is also often denoted as **Sinkhorn's distance**.

A significant advantage of the Sinkhorn-Knopp algorithm is its differentiability, which allows seamless backpropagation if its iterations, making it well suited for deep learning models. This property has sparked renewed interest in the community, particularly after a paper by Cuturi 2013, which showcased the efficiency and scalability of Sinkhorn updates as approximations to optimal transport. Consequently, the Sinkhorn-Knopp algorithm has found practical applications in various areas. For example, successful self-supervised learning methods such as SWaV (Caron, Misra, et al. 2020) and SeLa (Asano, Rupprecht, and Vedaldi 2019), along with supervised a few shot learning methods such as PT-MAP (Y. Hu, Gripon, and Pateux 2021), have made use of the Sinkhorn-Knopp algorithm and OT.

8

Conclusions and Future Directions

In this work, we introduce a novel self-supervised pretraining methodology (coined as BECLR) that ingrains both instance- and class-level insights within a contrastive learning framework. BECLR employs a dynamic clustered memory (DyCE) module, for providing a meaningful positive sampling strategy and enhancing the original contrastive batch. We explore the effects of equipartitioned assignments via optimal transport for updating DyCE and maintaining a highly-separable latent memory space. We accentuate the sample bias problem in U-FSL and propose an intuitive and effective OT-based feature alignment (OpTA) inference strategy to alleviate its effects. Our extensive evaluation results and qualitative experiments corroborate the efficacy of our design choices in BECLR, DyCE and OpTA on a variety of both in-domain and cross-domain U-FSL tasks. We demonstrate that BECLR sets a new state-of-the-art on the two most widely adopted few-shot classification benchmarks: miniImageNet and tieredImageNet, as well as on miniImagenet $\rightarrow \{$CUB, CropDiseases, EuroSAT$\}$.

As future work, we plan to demonstrate the applicability of BECLR on generic representation learning and additional computer vision downstream tasks (beyond FSL) by training deeper models (e.g., deeper vision transformer architectures would allow us to only pass unmasked patches to the networks and offer significant computation gains) on larger datasets, such as ImageNet. This would render BECLR as an all-inclusive holistic self-supervised learning approach toward representation learning and computer vision. Another potential future direction could be the exploration of the effects of combining our current contrastive framework with a clustering/assignment-based loss term, which would operate on the separable partitions of DyCE and enforce consistency between student and teacher on a clustering-level. Furthermore, our OT-based feature alignment OpTA between support and query sets could find applications on supervised FSL, and in particular, within a meta-learning episodic pretraining framework, where we would first align the support and query embeddings before applying the loss term. Finally, in this work, we have not explicitly targeted the large distribution shift between base and novel classes in the cross-domain setting. A potential future work could involve an additional add-on component, which would bridge this distribution shift (similar to how OpTA accounted for the distribution shift between support and query sets). This module could potentially be combined with our current BECLR, DyCE and OpTA to further advance the state-of-the-art in cross-domain U-FSL settings.

Bibliography

[1] Frank L Hitchcock. “The distribution of a product from several sources to numerous localities”. In: *Journal of mathematics and physics* 20.1-4 (1941), pp. 224–230 (cit. on pp. 55, 56).

[2] L Kantorovich. “On the transfer of masses (in Russian)”. In: *Doklady Akademii Nauk*. Vol. 37. 1942, p. 227 (cit. on p. 55).

[3] Warren S McCulloch and Walter Pitts. “A logical calculus of the ideas immanent in nervous activity”. In: *The bulletin of mathematical biophysics* 5 (1943), pp. 115–133 (cit. on pp. 23, 24).

[4] S Kullback and RA Leibler. “10.1214/aoms/1177729694”. In: *Ann. Math. Stat* 22 (1951), pp. 79–86 (cit. on p. 52).

[5] Herbert Robbins and Sutton Monro. “A stochastic approximation method”. In: *The annals of mathematical statistics* (1951), pp. 400–407 (cit. on p. 25).

[6] Kunihiko Fukushima. “Cognitron: A self-organizing multilayered neural network”. In: *Biological cybernetics* 20.3-4 (1975), pp. 121–136 (cit. on p. 24).

[7] Charles A Micchelli. *Interpolation of scattered data: distance matrices and conditionally positive definite functions*. Springer, 1984 (cit. on p. 24).

[8] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. “Learning representations by back-propagating errors”. In: *nature* 323.6088 (1986), pp. 533–536 (cit. on p. 26).

[9] Jürgen Schmidhuber. “Evolutionary principles in self-referential learning, or on learning how to learn: the meta-meta-... hook”. PhD thesis. Technische Universität München, 1987 (cit. on p. 43).

[10] Yann LeCun, Bernhard Boser, et al. “Backpropagation applied to handwritten zip code recognition”. In: *Neural computation* 1.4 (1989), pp. 541–551 (cit. on p. 28).

[11] Wei Zhang et al. “Parallel distributed processing model with local space-invariant interconnections and its optical architecture”. In: *Applied optics* 29.32 (1990), pp. 4790–4797 (cit. on p. 28).

[12] Sepp Hochreiter. “Untersuchungen zu dynamischen neuronalen Netzen”. In: *Diploma, Technische Universität München* 91.1 (1991), p. 31 (cit. on p. 31).

[13] Kurt Hornik. “Approximation capabilities of multilayer feedforward networks”. In: *Neural networks* 4.2 (1991), pp. 251–257 (cit. on p. 24).

[14] George Cybenko. “Approximation by superpositions of a sigmoidal function”. In: *Mathematics of Control, Signals and Systems* 5 (1992), pp. 455–455 (cit. on p. 24).

[15] Shun-ichi Amari. “Backpropagation and stochastic gradient descent method”. In: *Neurocomputing* 5.4-5 (1993), pp. 185–196 (cit. on p. 26).

[16] Jane Bromley et al. “Signature verification using a” siamese” time delay neural network”. In: *Advances in neural information processing systems* 6 (1993) (cit. on p. 33).

[17] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. “Learning long-term dependencies with gradient descent is difficult”. In: *IEEE transactions on neural networks* 5.2 (1994), pp. 157–166 (cit. on p. 31).

[18] Christopher M Bishop. *Neural networks for pattern recognition*. Oxford university press, 1995 (cit. on p. 31).

[19] Yann LeCun, Larry Jackel, et al. “Comparison of learning algorithms for handwritten digit recognition”. In: *International conference on artificial neural networks*. Vol. 60. 1. Perth, Australia. 1995, pp. 53–60 (cit. on p. 28).

[20] Solomon Kullback. *Information theory and statistics*. Courier Corporation, 1997 (cit. on p. 52).

[21] Tom B Brown et al. “Language Models are Few-Shot Learners. 2020. doi: 10.48550”. In: *arxiv* (2005), pp. 5–7 (cit. on p. 32).

[22] Sumit Chopra, Raia Hadsell, and Yann LeCun. “Learning a similarity metric discriminatively, with application to face verification”. In: *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR’05)*. Vol. 1. IEEE. 2005, pp. 539–546 (cit. on p. 33).

[23] Raia Hadsell, Sumit Chopra, and Yann LeCun. “Dimensionality reduction by learning an invariant mapping”. In: *2006 IEEE computer society conference on computer vision and pattern recognition (CVPR’06)*. Vol. 2. IEEE. 2006, pp. 1735–1742 (cit. on p. 33).
[24] Brian D Ripley. *Pattern recognition and neural networks*. Cambridge university press, 2007 (cit. on p. 31).
[25] Pascal Vincent et al. “Extracting and composing robust features with denoising autoencoders”. In: *Proceedings of the 25th international conference on Machine learning*. 2008, pp. 1096–1103 (cit. on p. 37).
[26] Jia Deng et al. “Imagenet: A large-scale hierarchical image database”. In: *2009 IEEE conference on computer vision and pattern recognition*. Ieee. 2009, pp. 248–255 (cit. on p. 32).
[27] Trevor Hastie et al. *The elements of statistical learning: data mining, inference, and prediction*. Vol. 2. Springer, 2009 (cit. on p. 32).
[28] Armand Joulin and Francis Bach. “A convex relaxation for weakly supervised classifiers”. In: *arXiv preprint arXiv:1206.6413* (2012) (cit. on p. 39).
[29] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “Imagenet classification with deep convolutional neural networks”. In: *Advances in neural information processing systems* 25 (2012) (cit. on pp. 28, 45).
[30] Marco Cuturi. “Sinkhorn distances: Lightspeed computation of optimal transport”. In: *Advances in neural information processing systems* 26 (2013) (cit. on pp. 39, 57).
[31] Karen Simonyan and Andrew Zisserman. “Very deep convolutional networks for large-scale image recognition”. In: *arXiv preprint arXiv:1409.1556* (2014) (cit. on p. 31).
[32] Kaiming He and Jian Sun. “Convolutional neural networks at constrained time cost”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 5353–5360 (cit. on p. 31).
[33] Christian Szegedy et al. “Going deeper with convolutions”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 1–9 (cit. on p. 31).
[34] Xi Chen et al. “Infogan: Interpretable representation learning by information maximizing generative adversarial nets”. In: *Advances in neural information processing systems* 29 (2016) (cit. on p. 45).
[35] Jeff Donahue, Philipp Krähenbühl, and Trevor Darrell. “Adversarial feature learning”. In: *arXiv preprint arXiv:1605.09782* (2016) (cit. on p. 45).
[36] Kaiming He, Xiangyu Zhang, et al. “Deep residual learning for image recognition”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778 (cit. on pp. 31, 45).
[37] Mehdi Noroozi and Paolo Favaro. “Unsupervised learning of visual representations by solving jigsaw puzzles”. In: *European conference on computer vision*. Springer. 2016, pp. 69–84 (cit. on p. 37).
[38] Deepak Pathak et al. “Context encoders: Feature learning by inpainting”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 2536–2544 (cit. on p. 37).
[39] Oriol Vinyals et al. “Matching networks for one shot learning”. In: *Advances in neural information processing systems* 29 (2016) (cit. on pp. 42, 45, 49).
[40] Richard Zhang, Phillip Isola, and Alexei A Efros. “Colorful image colorization”. In: *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part III 14*. Springer. 2016, pp. 649–666 (cit. on p. 37).
[41] Bolei Zhou et al. “Learning deep features for discriminative localization”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 2921–2929 (cit. on p. 48).
[42] Michael Eickenberg et al. “Seeing it all: Convolutional network layers map the function of the human visual system”. In: *NeuroImage* 152 (2017), pp. 184–194 (cit. on p. 28).
[43] Chelsea Finn, Pieter Abbeel, and Sergey Levine. “Model-agnostic meta-learning for fast adaptation of deep networks”. In: *International conference on machine learning*. PMLR. 2017, pp. 1126–1135 (cit. on pp. 42, 43).
[44] Jake Snell, Kevin Swersky, and Richard Zemel. “Prototypical networks for few-shot learning”. In: *Advances in neural information processing systems* 30 (2017) (cit. on pp. 44, 46).

[45] Jonas Adler and Sebastian Lunz. “Banach wasserstein gan”. In: *Advances in neural information processing systems* 31 (2018) (cit. on p. 53).
[46] David Berthelot et al. “Understanding and improving interpolation in autoencoders via an adversarial regularizer”. In: *arXiv preprint arXiv:1807.07543* (2018) (cit. on p. 45).
[47] Mathilde Caron, Piotr Bojanowski, et al. “Deep clustering for unsupervised learning of visual features”. In: *Proceedings of the European conference on computer vision (ECCV)*. 2018, pp. 132–149 (cit. on p. 45).
[48] Jacob Devlin et al. “Bert: Pre-training of deep bidirectional transformers for language understanding”. In: *arXiv preprint arXiv:1810.04805* (2018) (cit. on p. 38).
[49] Kyle Hsu, Sergey Levine, and Chelsea Finn. “Unsupervised learning via meta-learning”. In: *arXiv preprint arXiv:1810.02334* (2018) (cit. on pp. 2, 42, 45).
[50] Yoshua Bengio Ian Goodfellow and Aaron Courville. “Deep learning: The MIT Press, 2016, 800 pp, ISBN: 0262035618”. In: *Genetic programming and evolvable machines* 19.1-2 (2018), pp. 305–307 (cit. on pp. 26, 29, 30, 32).
[51] Ilya Kuzovkin et al. “Activations of deep convolutional neural networks are aligned with gamma band activity of human visual cortex”. In: *Communications biology* 1.1 (2018), p. 107 (cit. on p. 28).
[52] Chris Olah, Alexander Mordvintsev, and Ludwig Schubert. “Feature visualization: How neural networks build up their understanding of images”. In: *distill* (2018) (cit. on p. 30).
[53] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. “Representation learning with contrastive predictive coding”. In: *arXiv preprint arXiv:1807.03748* (2018) (cit. on pp. 34, 36, 48).
[54] Mengye Ren et al. “Meta-learning for semi-supervised few-shot classification”. In: *arXiv preprint arXiv:1803.00676* (2018) (cit. on pp. 42, 49).
[55] Zhirong Wu et al. “Unsupervised feature learning via non-parametric instance discrimination”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 3733–3742 (cit. on p. 34).
[56] Antreas Antoniou and Amos Storkey. “Assume, augment and learn: Unsupervised few-shot meta-learning via random labels and data augmentation”. In: *arXiv preprint arXiv:1902.09884* (2019) (cit. on pp. 2, 42).
[57] Yuki Markus Asano, Christian Rupprecht, and Andrea Vedaldi. “Self-labelling via simultaneous clustering and representation learning”. In: *arXiv preprint arXiv:1911.05371* (2019) (cit. on p. 57).
[58] Guneet S Dhillon et al. “A baseline for few-shot image classification”. In: *arXiv preprint arXiv:1909.02729* (2019) (cit. on pp. 2, 42).
[59] Dan Hendrycks et al. “Using self-supervised learning can improve model robustness and uncertainty”. In: *Advances in neural information processing systems* 32 (2019) (cit. on p. 33).
[60] Zilong Ji et al. “Unsupervised few-shot learning via self-supervised training”. In: *arXiv preprint arXiv:1912.12178* (2019) (cit. on p. 42).
[61] Siavash Khodadadeh, Ladislau Boloni, and Mubarak Shah. “Unsupervised meta-learning for few-shot image classification”. In: *Advances in neural information processing systems* 32 (2019) (cit. on pp. 2, 42, 45, 46).
[62] Adam Paszke et al. “Pytorch: An imperative style, high-performance deep learning library”. In: *Advances in neural information processing systems* 32 (2019) (cit. on pp. 29, 31).
[63] Gabriel Peyré, Marco Cuturi, et al. “Computational optimal transport: With applications to data science”. In: *Foundations and Trends® in Machine Learning* 11.5-6 (2019), pp. 355–607 (cit. on pp. 39, 54, 55, 57).
[64] Ben Poole et al. “On variational bounds of mutual information”. In: *International Conference on Machine Learning*. PMLR. 2019, pp. 5171–5180 (cit. on p. 49).
[65] Geoffrey Schiebinger et al. “Optimal-transport analysis of single-cell gene expression identifies developmental trajectories in reprogramming”. In: *Cell* 176.4 (2019), pp. 928–943 (cit. on p. 53).
[66] Jiaming Song and Stefano Ermon. “Understanding the limitations of variational mutual information estimators”. In: *arXiv preprint arXiv:1910.06222* (2019) (cit. on p. 49).
[67] Mathilde Caron, Ishan Misra, et al. “Unsupervised learning of visual features by contrasting cluster assignments”. In: *Advances in neural information processing systems* 33 (2020), pp. 9912–9924 (cit. on pp. 39, 57).

[68] Ting Chen et al. “A simple framework for contrastive learning of visual representations”. In: *International conference on machine learning*. PMLR. 2020, pp. 1597–1607 (cit. on pp. 2, 32–34, 49).
[69] Alexey Dosovitskiy et al. “An image is worth 16x16 words: Transformers for image recognition at scale”. In: *arXiv preprint arXiv:2010.11929* (2020) (cit. on pp. 38, 45).
[70] Jean-Bastien Grill et al. “Bootstrap your own latent-a new approach to self-supervised learning”. In: *Advances in neural information processing systems* 33 (2020), pp. 21271–21284 (cit. on pp. 32, 36, 50).
[71] Kaiming He, Haoqi Fan, et al. “Momentum contrast for unsupervised visual representation learning”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 9729–9738 (cit. on pp. 2, 35, 47, 50).
[72] Siavash Khodadadeh, Sharare Zehtabian, et al. “Unsupervised meta-learning through latent-space interpolation in generative models”. In: *arXiv preprint arXiv:2006.10236* (2020) (cit. on p. 2).
[73] Dong Bok Lee et al. “Meta-gmvae: Mixture of gaussian vae for unsupervised meta-learning”. In: *International Conference on Learning Representations*. 2020 (cit. on p. 42).
[74] John Lee, Nicholas P Bertrand, and Christopher J Rozell. “Unbalanced optimal transport regularization for imaging problems”. In: *IEEE Transactions on Computational Imaging* 6 (2020), pp. 1219–1232 (cit. on p. 53).
[75] Carlos Medina, Arnout Devos, and Matthias Grossglauser. “Self-supervised prototypical transfer learning for few-shot classification”. In: *arXiv preprint arXiv:2006.11325* (2020) (cit. on pp. 2, 42, 46, 47).
[76] Ishan Misra and Laurens van der Maaten. “Self-supervised learning of pretext-invariant representations”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 6707–6717 (cit. on p. 32).
[77] Christoph Molnar. *Interpretable machine learning*. Lulu. com, 2020 (cit. on p. 31).
[78] Martin Popel et al. “Transforming machine translation: a deep learning system reaches news translation quality comparable to human professionals”. In: *Nature communications* 11.1 (2020), p. 4381 (cit. on p. 32).
[79] Yonglong Tian et al. “Rethinking few-shot image classification: a good embedding is all you need?” In: *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XIV 16*. Springer. 2020, pp. 266–282 (cit. on pp. 2, 42).
[80] Mahmoud Assran, Mathilde Caron, Ishan Misra, Piotr Bojanowski, Armand Joulin, et al. “Semi-supervised learning of visual features by non-parametrically predicting view assignments with support samples”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 8443–8452 (cit. on p. 39).
[81] Hangbo Bao et al. “Beit: Bert pre-training of image transformers”. In: *arXiv preprint arXiv:2106.08254* (2021) (cit. on p. 38).
[82] Wentao Chen et al. “Few-shot learning with part discovery and augmentation from unlabeled images”. In: *arXiv preprint arXiv:2105.11874* (2021) (cit. on pp. 2, 42, 47, 48).
[83] Xinlei Chen and Kaiming He. “Exploring simple siamese representation learning”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2021, pp. 15750–15758 (cit. on pp. 2, 32, 36, 37, 42, 49, 50).
[84] Zitian Chen, Subhransu Maji, and Erik Learned-Miller. “Shot in the dark: Few-shot learning with no base-class labels”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 2668–2677 (cit. on p. 2).
[85] Debidatta Dwibedi et al. “With a little help from my friends: Nearest-neighbor contrastive learning of visual representations”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 9588–9597 (cit. on p. 35).
[86] Priya Goyal, Mathilde Caron, et al. “Self-supervised pretraining of visual features in the wild”. In: *arXiv preprint arXiv:2103.01988* (2021) (cit. on p. 32).
[87] Yuqing Hu, Vincent Gripon, and Stéphane Pateux. “Leveraging the feature distribution in transfer-based few-shot learning”. In: *International Conference on Artificial Neural Networks*. Springer. 2021, pp. 487–499 (cit. on p. 57).

[88] Steinar Laenen and Luca Bertinetto. “On episodes, prototypical networks, and few-shot learning”. In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 24581–24592 (cit. on pp. 2, 42).
[89] Grace W Lindsay. “Convolutional neural networks as a model of the visual system: Past, present, and future”. In: *Journal of cognitive neuroscience* 33.10 (2021), pp. 2017–2031 (cit. on p. 28).
[90] Jinghao Zhou et al. “ibot: Image bert pre-training with online tokenizer”. In: *arXiv preprint arXiv:2111.07832* (2021) (cit. on p. 38).
[91] Mahmoud Assran, Mathilde Caron, Ishan Misra, Piotr Bojanowski, Florian Bordes, et al. “Masked siamese networks for label-efficient learning”. In: *European Conference on Computer Vision*. Springer. 2022, pp. 456–473 (cit. on p. 39).
[92] Yalin Bastanlar and Semih Orhan. “Self-supervised contrastive representation learning in computer vision”. In: *Artificial Intelligence Annual Volume 2022*. IntechOpen, 2022 (cit. on p. 32).
[93] Lee Chen, Kuilin Chen, and Kuilin Chi-Guhn. “Unsupervised Few-shot Learning via Deep Laplacian Eigenmaps”. In: *arXiv preprint arXiv:2210.03595* (2022) (cit. on p. 2).
[94] Priya Goyal, Quentin Duval, et al. “Vision models are more robust and fair when pretrained on uncurated images without supervision”. In: *arXiv preprint arXiv:2202.08360* (2022) (cit. on p. 33).
[95] Kaiming He, Xinlei Chen, et al. “Masked autoencoders are scalable vision learners”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2022, pp. 16000–16009 (cit. on p. 32).
[96] Kaiming He, Xinlei Chen, et al. “Masked autoencoders are scalable vision learners”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2022, pp. 16000–16009 (cit. on p. 38).
[97] Ojas Kishorkumar Shirekar, Anuj Singh, and Hadi Jamali-Rad. “Self-Attention Message Passing for Contrastive Few-Shot Learning”. In: *arXiv e-prints* (2022), arXiv–2210 (cit. on pp. 2, 42).
[98] Rayan Krishnan, Pranav Rajpurkar, and Eric J Topol. “Self-supervised learning in medicine and healthcare”. In: *Nature Biomedical Engineering* 6.12 (2022), pp. 1346–1352 (cit. on p. 33).
[99] Yuning Lu et al. “Self-supervision can be a good few-shot learner”. In: *European Conference on Computer Vision*. Springer. 2022, pp. 740–758 (cit. on pp. 2, 42, 49, 50).
[100] Zhenda Xie et al. “Simmim: A simple framework for masked image modeling”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 9653–9663 (cit. on p. 38).
[101] Han-Jia Ye, Lu Han, and De-Chuan Zhan. “Revisiting unsupervised meta-learning via the characteristics of few-shot tasks”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 45.3 (2022), pp. 3721–3737 (cit. on p. 42).
[102] Randall Balestriero et al. “A cookbook of self-supervised learning”. In: *arXiv preprint arXiv:2304.12210* (2023) (cit. on p. 32).
[103] Wentao Hu et al. “Meta-DM: Applications of Diffusion Models on Few-Shot Learning”. In: *arXiv preprint arXiv:2305.08092* (2023) (cit. on p. 2).
[104] Huiwon Jang, Hankook Lee, and Jinwoo Shin. “Unsupervised Meta-learning via Few-shot Pseudo-supervised Contrastive Learning”. In: *arXiv preprint arXiv:2303.00996* (2023) (cit. on p. 2).
[105] Maxime Oquab et al. “Dinov2: Learning robust visual features without supervision”. In: *arXiv preprint arXiv:2304.07193* (2023) (cit. on p. 38).
[106] O Ciga, T Xu, and AL Martel. “Self supervised contrastive learning for digital histopathology. arXiv 2020”. In: *arXiv preprint arXiv:2011.13971* () (cit. on p. 33).