N° d'ordre NNT : 2023STET009

THESE de DOCTORAT DE L'UNIVERSITE JEAN MONNET SAINT-ETIENNE
membre de
l'UNIVERSITE DE LYON

Ecole Doctorale N° 488
Sciences Ingénierie Santé

Spécialité de doctorat :
Discipline : Informatique

Soutenue publiquement le 29/03/2023, par :

Quentin Bouniot

Towards Few-Annotation Learning in Computer Vision: Application to Image Classification and Object Detection tasks

---

Devant le jury composé de :

Céline Hudelot	Professeure	CentraleSupélec	Rapporteuse
Nicolas Thome	Professeur	Sorbonne Université	Rapporteur
Diane Larlus	Chercheuse	Naver Labs Europe	Examinatrice
Devis Tuia	Professeur Associé	EPFL	Examinateur
David Filliat	Professeur	ENSTA	Examinateur
Ievgen Redko	Chercheur	Huawei	Invité
Romaric Audigier	Chercheur	CEA-List	Encadrant
Angélique Loesch	Chercheuse	CEA-List	Encadrante
Amaury Habrard	Professeur	Université Jean Monnet	Directeur de thèse

Abstract

In this thesis, we develop theoretical, algorithmic and experimental contributions for Machine Learning with limited labels, and more specifically for the tasks of Image Classification and Object Detection in Computer Vision. In a first contribution, we are interested in bridging the gap between theory and practice for popular Meta-Learning algorithms used in Few-Shot Classification. We make connections to Multi-Task Representation Learning, which benefits from solid theoretical foundations, to verify the best conditions for a more efficient meta-learning. Then, to leverage unlabeled data when training object detectors based on the Transformer architecture, we propose both an unsupervised pretraining and a semi-supervised learning method in two other separate contributions. For pretraining, we improve Contrastive Learning for object detectors by introducing the localization information. Finally, our semi-supervised method is the first tailored to transformer-based detectors.

Résumé

Nous développons dans cette thèse des contributions théoriques, algorithmiques et expérimentales pour l'apprentissage automatique avec peu d'annotations, et plus spécifiquement pour les tâches de classification d'images et de détection d'objets en vision par ordinateur. Dans une première contribution, nous nous intéressons à combler le fossé entre la théorie et la pratique concernant les algorithmes de méta-apprentissage les plus répandus, utilisés pour de la classification avec peu d'exemples. Nous établissons des liens avec l'apprentissage de représentation multi-tâches, qui bénéficie de bases théoriques solides, afin de vérifier les meilleures conditions amenant à un méta-apprentissage plus efficace. Ensuite, afin d'exploiter les données non étiquetées pour entraîner des détecteurs d'objets basés sur l'architecture Transformer, nous proposons à la fois un pré-entraînement non supervisé et une méthode d'apprentissage semi-supervisée dans deux autres contributions distinctes. Pour le pré-entraînement, nous améliorons l'apprentissage contrastif des détecteurs d'objets en introduisant les informations de localisation. Enfin, notre méthode semi-supervisée est la première adaptée aux détecteurs d'objets utilisant des modèles type Transformers.

Remerciements

Je voudrais avant tout remercier le service du SIALV et en particulier le labo LVA pour m'avoir accueilli pendant ces dernières années. J'ai eu la chance de travailler avec des personnes remarquables et bienveillantes qui ont été d'une aide précieuse. Je tiens également à remercier tous les autres doctorants avec qui j'ai eu la chance de partager des moments de discussions enrichissants, notamment mes co-bureaux Julien et Rémi. Vous avez été une source de motivation quotidienne.

Je voudrais exprimer ma gratitude envers mes encadrants du CEA, Romaric et Angélique, qui m'ont guidé et encadré depuis 4 ans ! On a commencé à travailler ensemble avant la thèse et je vous remercie d'avoir cru en moi. Je vais maintenant voler de mes propres ailes, mais je garderai toujours vos précieux conseils en tête. Je souhaite d'ailleurs beaucoup de courage à Angélique pour son nouveau poste de cheffe et de maman. Je voudrais également remercier Ievgen, qui a certes quitté l'encadrement officiellement à la moitié de la thèse, mais qui est resté de bon conseil tout du long ! Même avec la distance, en France ou à l'autre bout de l'Europe, tu m'as accompagné et es resté disponible. Un grand merci à mon directeur de thèse Amaury, qui malgré les difficultés et la distance a toujours été présent et disponible pour moi. Je n'ai pas pu venir à Saint-Etienne aussi souvent que je l'aurais voulu à cause du contexte sanitaire, mais je te suis reconnaissant de m'avoir fait confiance toi aussi.

Je voudrais aussi exprimer ma gratitude envers les membres du jury, je suis honoré d'avoir pu présenter mon travail devant vous. J'ai une pensée pour Diane et Devis qui ont pris de leur temps pour évaluer mon travail même à distance. David qui a pris la responsabilité de la présidence du jury. Enfin, je souhaite exprimer toute ma reconnaissance envers les rapporteurs Céline et Nicolas, qui ont consacré du temps à la lecture de mon manuscrit de thèse. J'ai été touché par leurs retours pertinents et constructifs.

Je ne peux pas terminer sans remercier toute ma famille et mes amis, de Tahiti et de Métropole, qui ont été d'un soutien inconditionnel tout du long. Merci à Papa, Maman, Alexandre, Raphaël, Tiapo, Sylvie et Christophe, Mamie, Gilles et Nathalie, Alain et Cathy, sans oublier Karine et Frédéric. Je sais que vous ne comprenez pas tout ce que je fais, mais je vous remercie de votre soutien. Merci à mes amis de Tahiti, Matthias, Jonathan et Kim. Enfin, évidemment, je voudrais exprimer ma gratitude envers ma compagne Anne-Maëlle, qui m'a soutenu et conseillé tout les jours. Ça a été essentiel et je ne serais pas arrivé au bout sans toi.

Contents

Notations

a	A scalar (integer or real)
$\mathbf{a}$	A vector
$\mathbf{A}$	A matrix
$\mathcal{A}$	A set
$\mathbb{A}$	A space
$\mathbb{R}$	The space of real numbers
$\mathbf{a}_i$	Element i of vector $\mathbf{a}$, with indexing starting at 1
$\mathbf{A}_{(i,j)}$	Element i, j of matrix $\mathbf{A}$
$[[N]]$	The set of all integers from 1 to N: $[[N]] := \{1, \ldots, N\}$
θ	Parameters or *weights* of the model
θ^*	Optimal, or ideal parameters
$\hat{\theta}$	Empirical parameters
$\mathbf{A}^\top, \mathbf{a}^\top$	Transpose of matrix $\mathbf{A}$ or vector $\mathbf{a}$
$\|\mathbf{x}\|_p$	L_p norm of $\mathbf{x}$
$\|\mathbf{x}\|$	L_2 norm of $\mathbf{x}$
$\|\mathbf{X}\|_F$	Frobenius norm of $\mathbf{X}$
$\langle \mathbf{x}, \mathbf{y} \rangle$	Scalar product between $\mathbf{x}$ and $\mathbf{y}$
$P(\mathbf{x})$	Probability distribution over $\mathbf{x}$
$\mathbf{x} \sim P$	Random variable $\mathbf{x}$ has distribution P
$\mathbb{E}_{\mathbf{x} \sim P}[f(\mathbf{x})]$	Expectation of $f(\mathbf{x})$ with $\mathbf{x}$ following distribution P
$\mathcal{N}(\mu, \Sigma)$	Gaussian distribution with mean μ and covariance Σ
$\mathbb{1}_{\text{condition}}$	function that is 1 if condition is true, 0 otherwise

Acronyms

ML	Machine Learning
DL	Deep Learning
CV	Computer Vision
SL	Supervised Learning
SSL	Semi-Supervised Learning
Self-SL	Self-Supervised Learning
UL	Unsupervised Learning
MSL	Many-Shot Learning
FSL	Few-Shot Learning
FAL	Few-Annotation Learning
OD	Object Detection
FSC	Few-Shot Classification
FSOD	Few-Shot Object Detection
SSOD	Semi-Supervised Object Detection
MLP	Multi-Layer Perceptron
CNN	Convolutional Neural Network
MTR	Multi-Task Representation Learning
EMA	Exponential Moving Average
COCO	Microsoft Common Objects in Context dataset
ILSVRC	ImageNet Large-Scale Visual Recognition Challenge dataset
PASCAL VOC	PASCAL Visual Object Classes challenge dataset
p.p.	Percentage point
IoU	Intersection over Union, measures the overlap of predicted bounding box to the ground truth

AP	Average Precision metric
AP_{50}	AP calculated at IoU threshold of 0.5
AP_{50-95}	AP calculated for IoU threshold from 0.5 to 0.95 with step size of 0.05
mAP	mean Average Precision metric, mean over all the class APs

Introduction 1

Contents

This chapter provides an introduction to the main problems addressed in this thesis, as well as the presentation of the structure and contributions of this manuscript.

1.1 Context and Background

1.1.1 Context

Learning something new in real life does not necessarily mean going through a lot of examples in order to capture the essence of it. Even though it is said that it takes 10,000 hours to *master* a new skill [Gladwell, 2009], it is also true that it only takes 20 hours to *learn* it [Kaufman, 2013]. Humans are able to build upon prior experience and have the ability to adapt, allowing to combine previous observations with only little evidence for fast learning. This is particularly the case for recognition tasks, for which we are often capable of differentiating between two distinct objects after having seen only a few examples of them.

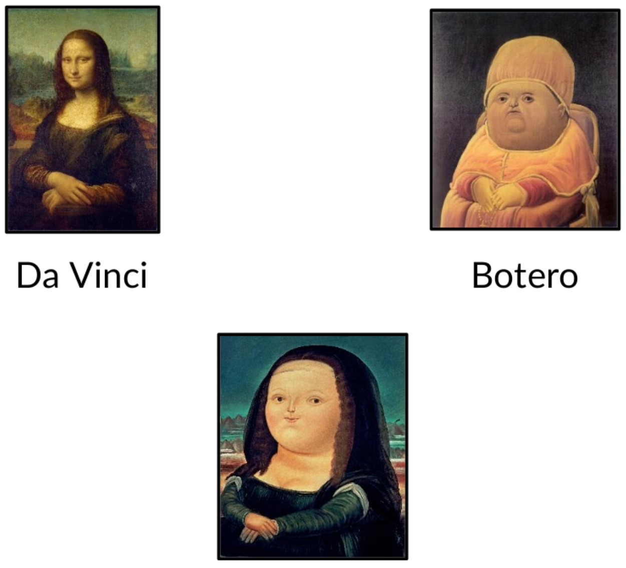


Figure 1.1. *Who is the painter ? Illustration of the human capacity to* learn from limited examples. *The top left painting is the* Mona Lisa *by* Leonardo Da Vinci *and the top right one is the* Pope Leo X after Raphael *by* Fernando Botero*. The bottom painting is* Mona Lisa, Age Twelve*, guessing the painter is left as an exercise to the reader.*

We give an illustrative example from paintings samples presented in Figure 1.1. It is quite obvious that one would easily guess and recognize the painter of the last painting below after having seen just one example of each painter's styles above. This idea has found its application in *machine learning* in a more general *few-shot learning* paradigm that wants to mimic the human capability to quickly learn how to solve a new problem. The above example is a direct illustration of *one-shot learning*.

1.1.2 Background

Learning models, *i.e.* estimating parameters from data, has long proven to be an effective approach in the literature for solving different kinds of applications, ranging from computer vision to natural language processing. This is what is commonly referred as *Machine Learning*.

Ever since AlexNet [Krizhevsky, 2009], the rise of Deep Learning (DL) [LeCun et al., 2015, Goodfellow et al., 2016] models for *Representation Learning* has led to near-human performance on many vision tasks. Current best performing architectures are based on Convolutional Neural Networks (CNN) [LeCun et al., 1989], such as the popular ResNet architecture [He et al., 2016], and more recently on Transformers [Vaswani et al., 2017, Dosovitskiy et al., 2021, Carion et al., 2020].

Training DL models consists in optimizing an *objective*, or *loss* function *w.r.t.* to some experience, or previously acquired *information*. Learning algorithms considered throughout this thesis are allowed to experience an entire *dataset*. A dataset is a collection of many examples, also called *data samples*. These examples are usually measurements, and more specifically in the case of Computer Vision, they most commonly take the form of *images*, annotated or not.

1.1.3 Learning frameworks

Training paradigms can be broadly separated into either *Supervised Learning (SL)*, *Unsupervised Learning (UL)*, or *Semi-supervised Learning (SSL)* depending on the amount of annotated information available during the learning process.

Supervised Learning (SL) The most common form of machine learning, deep or not, is *supervised learning* [Shalev-Shwartz and Ben-David, 2014, Mohri et al., 2018, LeCun et al., 2015]. During training, SL algorithms are shown examples associated with a *label*, or *target*. The term *supervised learning* comes from the view that an instructor, or teacher, is providing *supervision* through the *labels* [Goodfellow et al., 2016]. Indeed, the training optimizes supervised loss functions that take into account the additional information contained in the labels.

Unsupervised learning (UL) Unlike SL, UL algorithms [Shalev-Shwartz and Ben-David, 2014, Mohri et al., 2018, LeCun et al., 2015] experience a dataset *without labels*, *i.e.* without additional information. The objective is to learn useful properties about the *structure* of the input dataset, by, *e.g.* learning the entire probability distribution that generated the dataset, either explicitly or implicitly [Goodfellow et al., 2016]. This problem is generally seen as more difficult than SL, but also potentially more impactful since it requires less information on the data provided by the user [LeCun et al., 2015]. The recent *Self-Supervised learning (Self-SL)* paradigm is a particular case of UL, in which the data provides its own supervision to learn a practical representation.

Semi-Supervised Learning (SSL) In addition to data, SSL algorithms [Mohri et al., 2018] are supervised with labels, but not for all examples. This paradigm can be seen as halfway between supervised and unsupervised learning [Chapelle et al., 2009]. The dataset can be usually divided into *labeled data*, for which labels are provided, and *unlabeled data*, the labels of which are not known.

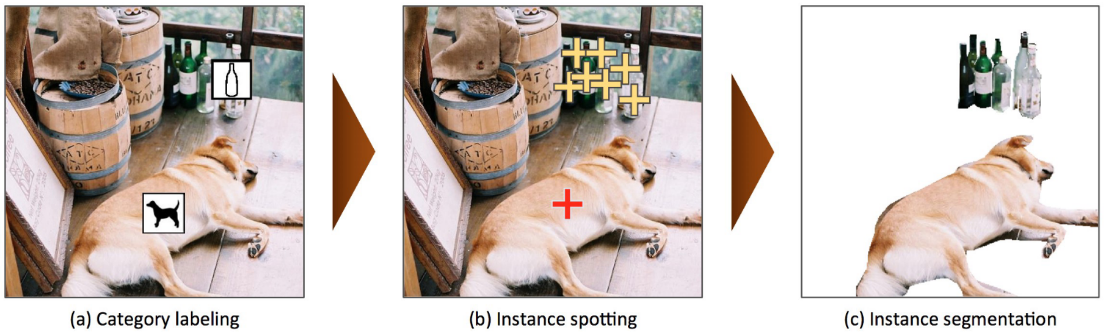


Figure 1.2. *Illustration of the annotation pipeline from the COCO dataset [Lin et al., 2014]. The pipeline is split into 3 primary tasks: (a) labeling the categories (class of object) present in the image, (b) locating and marking all instances of the labeled categories, and (c) segmenting each object instance.*

When having access to *large datasets*, SSL is often a good compromise between the two other frameworks in practice. Resulting models can achieve strong performance while requiring limited supervision. Even though SL is the most common form of DL, it is also the most restrictive one for practical use, as it requires to have access to *a large number of labels*.

1.2 Challenges and Objectives

1.2.1 The Problem of Labeling Data

Supervised Learning requires a lot of labeled data to be effective. The ImageNet dataset [Russakovsky et al., 2015], one of the most popular datasets for *image recognition*, contains from 1.2M (*ILSVRC 2012* or *ImageNet-1k*) to 14M labeled images (*ImageNet-21k*). However, the process of annotating data can be hard, time-consuming and thus, expensive. For a highly accurate dataset, humans must verify and manually annotate each collected example. Then, to reduce human errors, the process is repeated independently by multiple persons and the final label of each sample is taken as the convincing majority of labels given. Each year from 2010 to 2017, ImageNet employed 20,000 to 30,000 people to annotate the dataset through the Amazon Mechanical Turk (AMT)[1] service, paying a few cents for each image labeled[2].

Furthermore, more *dense* computer vision tasks, *i.e.* problems that require a deeper understanding of images and thus more complex annotations per image, take more

[1]https://www.mturk.com/
[2]https://www.nytimes.com/2012/11/20/science/for-web-images-creating-new-technology-to-seek-and-find.html

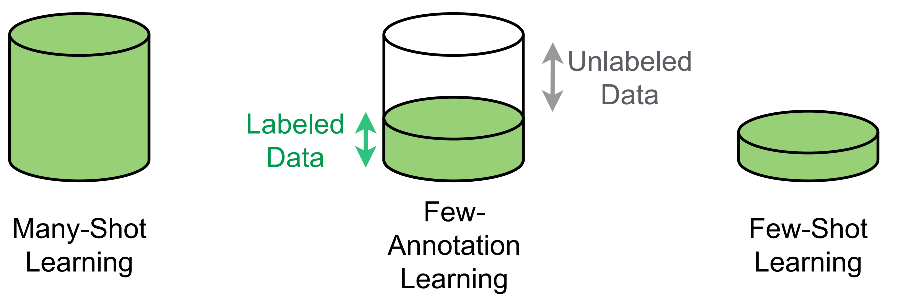


Figure 1.3. *Illustration of the different possible* training data conditions *encountered in practice, depending on the quantity of target data* (the size of the cylinder) *and labels* (in green) *available.*

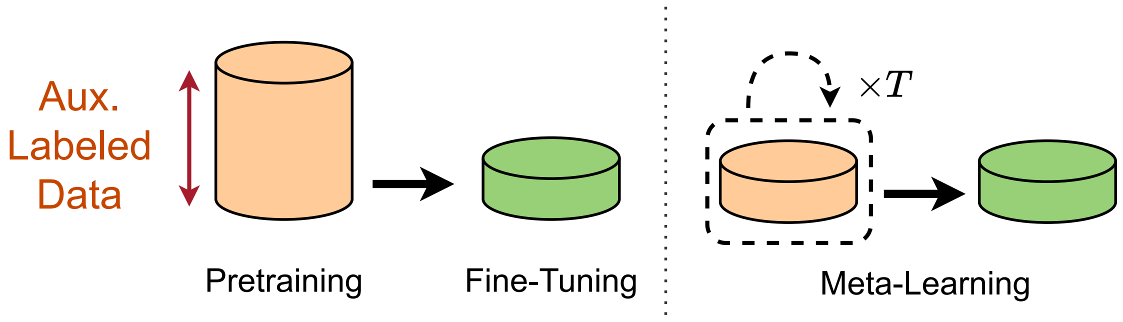
time to annotate. Figure 1.2 is an illustration of the full annotation process for the *MS COCO (COCO)* dataset [Lin et al., 2014]. COCO is one of the most popular large-scale dataset used for the *Object Detection (OD)* task, which consists in locating and classifying every instance of object in an image. The dataset is a collection of over 200k labeled images for a total of about 2.5M object instances. As we can see in Figure 1.2, the annotation pipeline is separated in three tasks, (a) *Category labeling*, (b) *Instance spotting* and (c) *Instance segmentation*. Here again, multiple AMT workers were asked to manually annotate and verify images, for a total of over 85,000 worker hours [Lin et al., 2014], *i.e.* over *9 worker years*.

The success stories of the above two examples of dataset labeling heavily rely on a *collaborative annotation process* to reduce the overall cost. However, not all datasets can be annotated in the same way. While anyone can label images of dogs and cats, data from more complex domains might require precious time from qualified workers. For instance, labeling medical or borehole images requires insights from health specialists and geophysicists, increasing even more the cost of annotation. In addition, in some cases, sensitive data or private datasets cannot be shared with anyone. For all these reasons, we often do not have access in practice to large-scale labeled datasets.

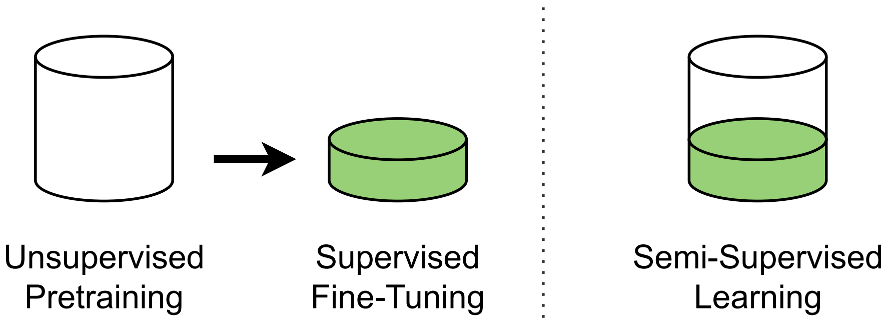
1.2.2 Training data conditions in practice

We can separate learning problems in practice in three categories, depending on the amount of *target data*, *i.e.* data linked to the task that we want to solve, but also depending on the *labels* available on this data. We give an illustration of these different categories in Figure 1.3.

Many-Shot Learning (MSL) The most common category, *MSL*, corresponds to learning problems for which we have access to *a lot of annotated data*. They usually arise when the data is easy to acquire *and* to annotate. We will not delve into the



(a) *Few-Shot Learning (FSL)*



(b) *Few Annotation Learning (FAL)*

Figure 1.4. *Illustration of commonly used approaches for* ***(a)*** FSL *and* ***(b)*** FAL *problems.* Auxiliary (Aux.) labeled data *is represented in* orange.

details of this setting as it represents the easiest learning conditions, given that we can directly apply *supervised algorithms*.

Few-Shot Learning (FSL) The hardest category of problems, *FSL*, is the most scarce in terms of target data *and* labels available. These problems have access to only few examples, or *shots*, to learn from, usually because of a difficult data collection. In this setting, *auxiliary data* can be used to compensate for the lack of *available target data*. We illustrate approaches commonly described in the literature in Figure 1.4a. Models are either pretrained on the auxiliary data, with supervision or not, and then refined on the target data *(left)*, or trained with *Meta-Learning* (that will be presented more in depth in Chapter 3) on auxiliary data by constructing a large number T of small tasks, to be able to adapt faster to the few data of the target task *(right)*.

Few-Annotation Learning (FAL) An intermediate category, *FAL*, includes problems for which we have a lot of target data but *few annotated ones*. While data gathering for these problems may be simple, labeling may be too costly. We give an illustration in Figure 1.4b of the possible approaches in this setting to leverage unlabeled

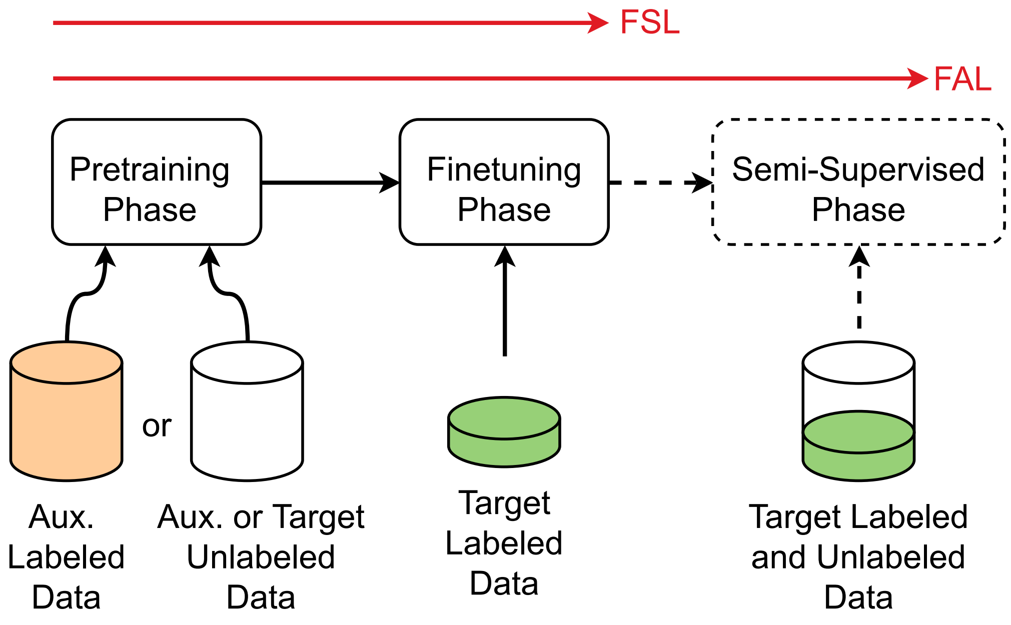


Figure 1.5. *A tentative general pipeline encompassing approaches of FSL and FAL, coping with limited target labels. The semi-supervised phase* (dotted lines) *is possible if unlabeled target data are also available (FAL).*

data on the target task to better learn on the few labeled data available. Methods in this setting either rely on an *unsupervised pretraining* phase on the unlabeled data followed by a *supervised fine-tuning* phase on the labeled data *(left)*, or *semi-supervised learning* to leverage unlabeled data along with labeled data during training *(right)*.

In this thesis, we are specifically interested in problems with limited labels available and explore both settings of FSL and FAL. In this case, from approaches in the literature, we can draw a general learning pipeline that distinguishes three different phases, illustrated in Figure 1.5, which are not always present. First, a *pretraining phase* either unsupervised or supervised on auxiliary labeled data, learns a general representation. Then, the *finetuning phase* adapts the representation learned to the target labeled data. Additionally, a *semi-supervised phase* can complete the training with both labeled and unlabeled target data.

While there is a large literature in MSL, research in FSL and, even more, in FAL, have been much more sparsely addressed. Yet, they represent the most common settings in industry because of the cost of data annotation, and sometimes even its collection. This discrepancy between research and industry interests motivated the work presented in this thesis.

1.2.3 Issues addressed

Throughout this thesis, we are interested in applications to *Computer Vision problems*, with a focus on *Image Classification* and *Object Detection* tasks. As detailed above, while the easiest learning conditions and the most effective methods require having access to a large amount of labeled images, learning with scarcely annotated data remains a challenging problem even though it corresponds to the most common setting in practice. We address the issues of *learning with limited labels* from different perspectives:

- On the one hand, our objective is to better understand the inner workings of learning algorithms in FSL settings from a *theoretical point of view*. Conditions and guarantees for a given algorithm to learn well are given by theoretical analyses and are expressed by *learning bounds*. These bounds often take the following form [Vapnik, 1999]:

$$\text{True Risk} - \text{Empirical Risk} \leq \frac{\text{Model Complexity}}{F(\text{Number of training data})}, \tag{1.1}$$

  where the *True* and *Empirical Risk* correspond to the expected value of the loss function, respectively computed on the *full data distribution* and the available *dataset*, the *Model Complexity* is a measure of the *representation capacity* of the model used, and F is a function of the *number of training data* used. However, in FSL where the number of training data available is small, this last specific term can hinder the training process. Through connections to frameworks with solid theoretical foundations, we aim to find the best training conditions for this setting and investigate this problem in Chapter 4, with the settings considered illustrated in Figure 1.6a.

- On the other hand, we want to improve *Object Detection* methods from an *algorithmic point of view* in FAL settings. The specificities of Object Detection have been barely addressed in the context of FAL, even though it represents a task with many practical applications. We are specifically interested in *unsupervised pretraining* and *semi-supervised learning* methods, investigated respectively in Chapter 5 and Chapter 6, with their respective settings illustrated in Figure 1.6b and Figure 1.6c.

In the following section, we discuss the overall structure of the manuscript and our contributions addressing the issues introduced above.

1.3 Outline and Contributions

We present in this section an overview of this thesis and the diverse associated publications.

1.3.1 Structure of the manuscript

This manuscript is organized as follows. The present Chapter 1 is meant as a motivation to the work presented in the following chapters. We introduced the different learning paradigms and the main settings encountered in practice. We illustrated the *time-cost problem of manual data labeling* with examples of the annotation process from popular datasets. This problem is at the heart of all the questions that will interest us throughout this thesis, since it dictates the amount of target annotated data available.

In Chapter 2, we delve more in depth into the different concepts in Machine Learning that will be important to understand this work. We present the learning process and theoretical notions with simple models, then deeper models and their specificities. Finally, since we will focus on using images as our data throughout this thesis, we introduce the popular tasks of *Image Classification* and *Object Detection*, and their respective recent literature.

In Chapter 3, we review related work in learning with limited labels. These works span the topics of *Data Augmentation*, *Meta-Learning*, *Transfer Learning* and *Semi-supervised Learning*. We introduce recent work in each topic, and conclude with the research directions outlined for this thesis.

Then, the following chapters present our contributions, each positioned *w.r.t.* the general training pipeline from Figure 1.5, in Figures 1.6a, 1.6b and 1.6c.

Chapter 4 is intended to bridge the gap between Multi-Task Representation Learning theory and practices in Meta-Learning for FSL. We investigate the behavior and theory behind popular meta-learning algorithms through the lens of theoretical assumptions from recent learning bounds for Multi-Task Representation Learning. This leads us to better understand these methods and improve their performance by forcing them to follow the conditions for a more efficient learning in FSL.

Chapter 5 focuses on pretraining strategies for learning Object Detector with limited labels. In this chapter, we propose a novel pretraining method for recent transformer-based object detectors, based on a Proposal-Contrastive Learning paradigm. Throughout the chapter we advocate for consistency in the level of information learned during pretraining, which forms the foundations of our method.

In Chapter 6, we propose a novel semi-supervised method tailored to transformer-based object detectors. We are interested in improving performance with scarcely

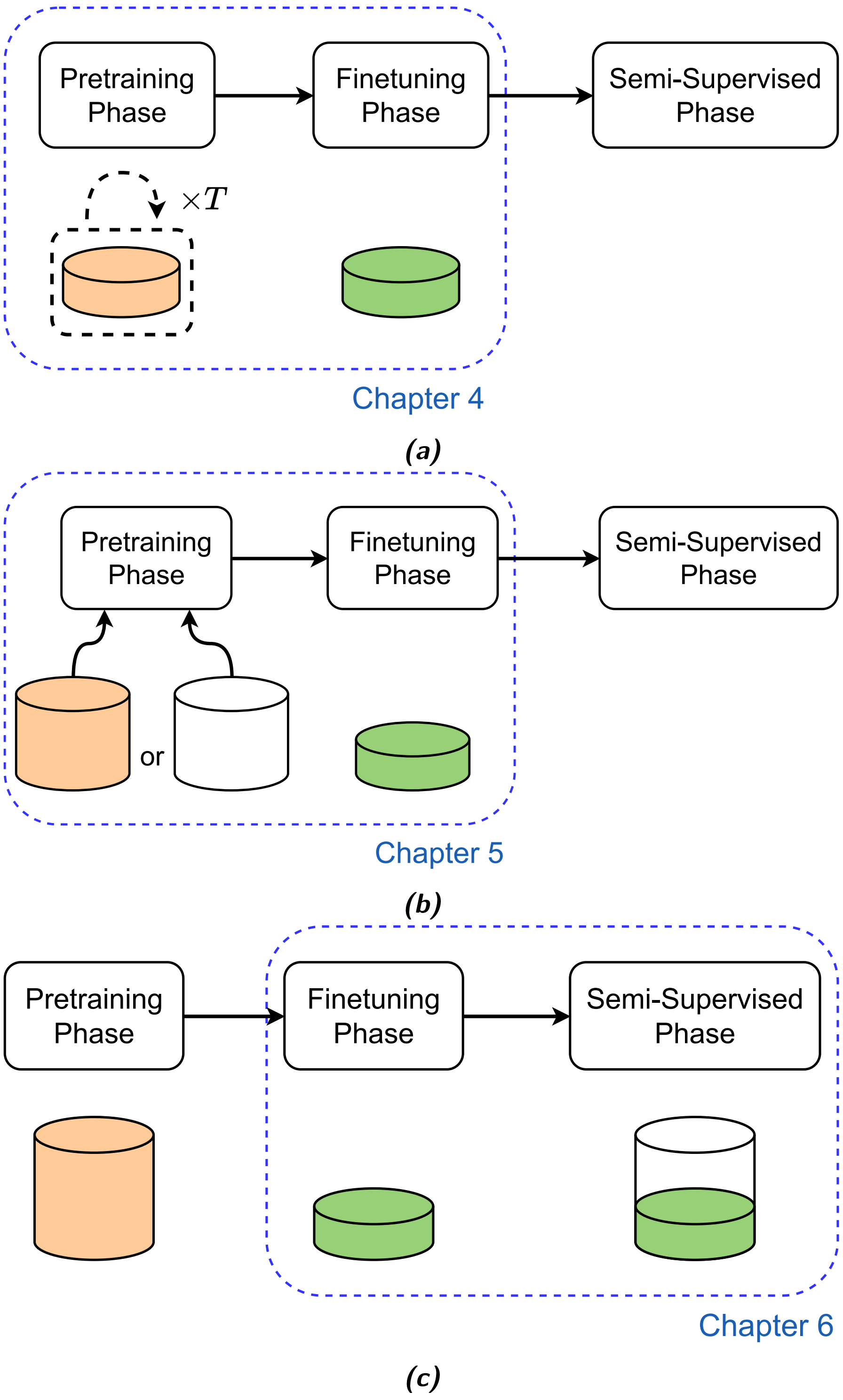


Figure 1.6. *Illustration of the position of our contributions presented in* ***(a)*** *Chapter 4,* ***(b)*** *Chapter 5 and* ***(c)*** *Chapter 6, in the general training pipeline from Figure 1.5.*

labeled data but with access to target unlabeled data, after fine-tuning on the target labeled data. We found that direct transpositions of popular practices with CNN can be counterproductive when using transformer-based detectors, and investigate ways to learn these latter models more efficiently in SSL for object detection with limited labels.

Finally, Chapter 7 concludes this thesis with perspectives on possible future research directions and a discussion on broader impacts of this line of work.

1.3.2 Scientific contributions

The contributions presented in this thesis led to the following peer-reviewed and published works:

- International Conferences:

  1. **Quentin Bouniot**, Ievgen Redko, Romaric Audigier, Angélique Loesch, Amaury Habrard. "Improving Few-Shot Learning through Multi-task Representation Learning Theory". In *European Conference on Computer Vision (ECCV)*, Oct. 2022. [Bouniot et al., 2022]
  2. **Quentin Bouniot**, Angélique Loesch, Romaric Audigier, Amaury Habrard. "Towards Few-Annotation Learning for Object Detection: Are Transformer-based Models More Efficient ?". In *Winter Conference on Applications of Computer Vision (WACV)*, Jan. 2023. [Bouniot et al., 2023b]
  3. **Quentin Bouniot**, Romaric Audigier, Angélique Loesch, Amaury Habrard. "Proposal-Contrastive Pretraining for Object Detection from Fewer Data". In *International Conference on Learning Representatons (ICLR)*, May 2023. [Bouniot et al., 2023a]

- International Workshop:

  4. **Quentin Bouniot**, Ievgen Redko, Romaric Audigier, Angélique Loesch, Amaury Habrard. "Putting Theory to Work : From Learning Bounds to Meta-Learning Algorithms". In *Neural Information Processing Conference (NeurIPS) Workshop on Meta-Learning (MetaLearn)*, Dec. 2020. [Bouniot et al., 2020]

- National Conference:

  5. **Quentin Bouniot**, Ievgen Redko, Romaric Audigier, Angélique Loesch, Amaury Habrard. "Vers une meilleure compréhension des méthodes de méta-apprentissage à travers la théorie de l'apprentissage de représentations

multi-tâches". In *Conférence sur l'Apprentissage Automatique (CAp)*, June 2021. [Bouniot et al., 2021]

- International Communication:

  6. **Quentin Bouniot** & Ievgen Redko, "Understanding Few-Shot Multi-Task Representation Learning Theory". In *International Conference on Learning Representations (ICLR) Blog Track*, May 2022. [Bouniot and Redko, 2022]

Chapter 4 is based on [Bouniot et al., 2022, Bouniot et al., 2020, Bouniot et al., 2021, Bouniot and Redko, 2022], Chapter 5 on [Bouniot et al., 2023a] and Chapter 6 on [Bouniot et al., 2023b].

The Basics of Learning from images

2

Contents

This chapter presents the preliminary knowledge to understand this thesis. The main objective is to describe basic notions and challenges in Machine Learning *and* Deep Learning, *with a focus on using* images *as our input data. We begin by explaining the theoretical context of Machine Learning problems in Section 2.1, then we move to Deep Learning, with a justification of the necessity of deeper models as well as a presentation of the specificities and techniques in Section 2.2. Finally, we give a general overview of the field of Computer Vision with a focus on the Image Classification and Object Detection tasks in Section 2.3.*

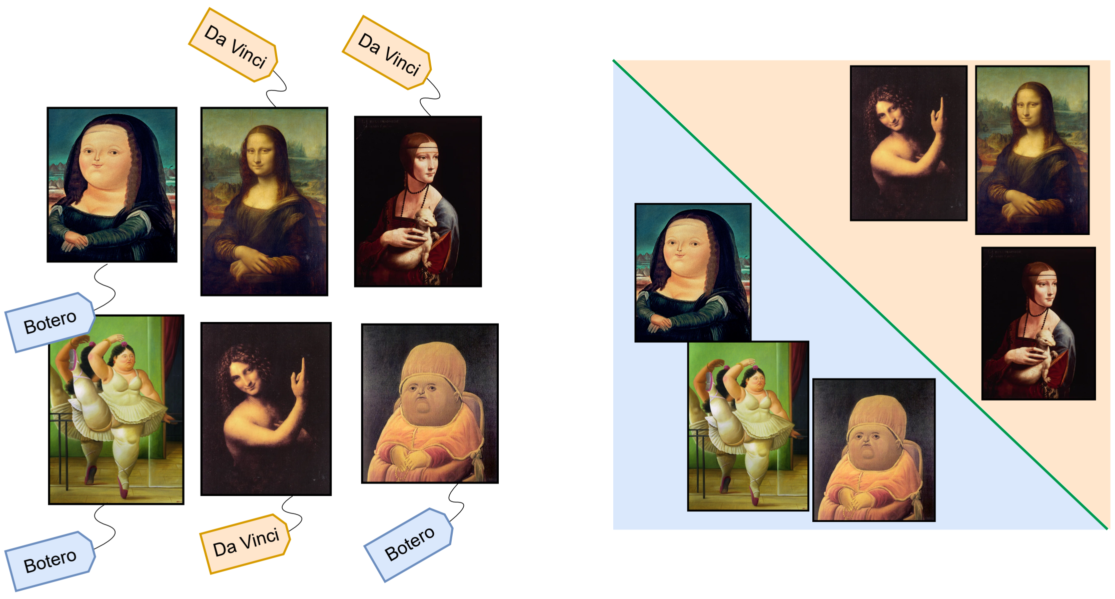


Figure 2.1. *We illustrate our problem using images of famous paintings, and ground-truth labels of the corresponding painters* (left). *We want to find a* model (right, the green line) *that can correctly predict the painters, which can be easily done by a human.*

2.1 Machine Learning

2.1.1 Objective

In this section, we want to solve a task for which we have a dataset containing a *ground truth label* associated to each input samples. We illustrate the problem with images in Figure 2.1. Using Machine Learning (ML), the goal is to *learn a model* that can correctly predict the label given an input data. This model is generally a *function*, potentially complex depending on the problem to solve, and this function can be represented by the set of its parameters, that we estimate from the data during the learning process. Throughout this thesis we will be specifically interested in the case of images. However, input data could correspond to any vector of values.

2.1.2 Learning Process

We formally describe the process of learning from data [Vapnik, 1999, Goodfellow et al., 2016] in this section. Suppose that we have N *training data samples* represented as d_1-dimensional input vectors from an input space $\mathbb{X}$: $\{\mathbf{x}_i\}_{i=1}^{N} \in \mathbb{X}^N \subseteq \mathbb{R}^{N\times d_1}$ with their associated *ground-truths*, or *labels*, represented as d_2-dimensional vectors from an output space $\mathbb{Y}$: $\{\mathbf{y}_i\}_{i=1}^N \in \mathbb{Y}^N \subseteq \mathbb{R}^{N\times d_2}$, corresponding to N observations of random vectors $\mathbf{x}$ and $\mathbf{y}$ that were drawn independently from fixed but unknown true data distributions $P(\mathbf{x})$ and $P(\mathbf{y}|\mathbf{x})$. We group these observations into matrices $\mathbf{X} \in \mathbb{R}^{N\times d_1}$ and $\mathbf{Y} \in \mathbb{R}^{N\times d_2}$ to form the *training set* $\mathcal{D}_{\text{train}}$, such that

$$\mathcal{D}_{\text{train}} := (\mathbf{X}, \mathbf{Y}) = ((\mathbf{x}_1, \mathbf{y}_1), \dots, (\mathbf{x}_N, \mathbf{y}_N)) \sim P(\mathbf{x}, \mathbf{y}), \tag{2.1}$$

with the joint probability distribution $P(\mathbf{x}, \mathbf{y}) = P(\mathbf{y}|\mathbf{x})P(\mathbf{x})$, called the *data-generating distribution*. We typically make the assumption that the data samples are *independent and identically distributed (i.i.d.)*. We also further assume that we have a *testing set* of data $\mathcal{D}_{\text{test}}$, obtained by the same underlying distribution $P(\mathbf{x}, \mathbf{y})$, that will be used to evaluate our future model and is never used during the training process that we describe in the next section.

2.1.3 Risk Minimization

The goal of Machine Learning is to find a function $f_\theta : \mathbb{X} \to \mathbb{Y}$, or *model*, that gives a good approximation of the labels associated to entries of $\mathbb{X}$ *w.r.t.* $P(\mathbf{y}|\mathbf{x})$. f_θ is parameterized by a set of *weights* $\theta \in \Theta$. The model f_θ is selected by a *learning algorithm* from a set of predefined models, called the *hypothesis space*: $\mathcal{H} \subseteq \{f_\theta | f_\theta : \mathbb{X} \to \mathbb{Y}, \theta \in \Theta\}$.

The learning is driven by a *loss function* $\mathcal{L}$, which provides a measure of the quality of a prediction given by the model outputs $f_\theta(\mathbf{x})$ *w.r.t.* the corresponding ground-truth $\mathbf{y}$, and the *true risk* $\mathcal{R}$ given by the expected value of the loss:

$$\mathcal{R}(\theta) := \int \mathcal{L}(\mathbf{y}, f_\theta(\mathbf{x}))dP(\mathbf{x}, \mathbf{y}). \tag{2.2}$$

Then, the *optimal parameters* θ^* can be found by minimizing the risk functional. However, since the *true* data-generating distribution is unknown, the only available information is contained in the training set $\mathcal{D}_{\text{train}}$ defined in Equation (2.1). Thus, in practice, we compute the *empirical risk* $\mathcal{E}$ from the training set as follows:

$$\mathcal{E}(\theta) := \frac{1}{N}\sum_{i=1}^{N} \mathcal{L}(\mathbf{y}_i, f_\theta(\mathbf{x}_i)). \tag{2.3}$$

Finally, the *empirical parameters* $\hat{\theta}$ are obtained by *Empirical Risk Minimization (ERM)*:

$$\hat{\theta} := \arg\min_{\theta\in\Theta} \mathcal{E}(\theta). \tag{2.4}$$

The induction principle of ERM assumes that the *empirical model* $f_{\hat{\theta}}$, which minimizes $\mathcal{E}$, results in a *true risk* close to its minimum as well [Vapnik, 1999]. Finally, the process of minimizing $\mathcal{E}$ is solved as an *optimization problem*, and it is referred to as *training a model*.

2.1.4 Learning Bounds

The soundness of the ERM principle largely depends on the *uniform convergence* of the empirical risk $\mathcal{E}$ to the true risk $\mathcal{R}$, given $\epsilon > 0$:

$$P\{\sup_{\theta\in\Theta} |\mathcal{R}(\theta) - \mathcal{E}(\theta)| > \epsilon\} \to 0 \qquad \text{when} \qquad N \to \infty, \tag{2.5}$$

and the *rate of convergence*, *i.e.* how fast $f_{\hat{\theta}}$ converges towards f_{θ^*} in terms of number of samples N, which is also called the *sample efficiency*. The more sample-efficient a learning algorithm is, the more interesting in practice it becomes.

Rates of convergence for a given class of function $\mathcal{H}$ are obtained by theoretical analysis, but mainly depend on the *capacity* $C(\mathcal{H})$ of the models. Informally, a model's capacity indicates its ability to fit a wide variety of functions, and can be quantified in different ways, with the most known ones being the *Vapnik-Chervonenkis dimension* [Vapnik and Chervonenkis, 2015] and the *Rademacher complexity* [Bartlett and Mendelson, 2002]. However, these are mainly theoretical quantities useful for upper-bounding purposes. Accurately measuring the capacity of a class of model in practice is still an open problem. Statistical learning theory [Vapnik, 1999, Hastie et al., 2009, Shalev-Shwartz and Ben-David, 2014, Mohri et al., 2018] gives us *learning bounds* providing justification of the learning process, as well as guarantees on the rate of convergence and *generalizability* of the models. We discuss the latter in the next section.

2.1.5 Generalization Problem

The question of the *generalizability* of a model arises when evaluating on the testing data $\mathcal{D}_{\text{test}}$. The ability of a model to perform well on unseen data is called *generalization*.

When training a model, we can compute an *error measure* on the training set, that we call the *training error*, and this error is usually reduced during the optimization process of the ERM. However, we want our model to be effective on the full underlying distribution of inputs $P(\mathbf{x})$ to be useful in practice, not only on the training observations $\mathbf{X}$. Thus, we also measure the *test error* on $\mathcal{D}_{\text{test}}$ and want it to be low as well, to estimate the *generalization error* on unseen data. The need for generalizability is what separates machine learning from optimization.

The *capacity* of a model has a great influence on its generalization capabilities, as can be seen in Figure 2.2. A model with low capacity will likely *underfit*, *i.e.* not predict accurate labels, which leads to a high training *and* test error. On the other hand, a model with a too big capacity will likely *overfit*, *i.e.* memorize training data leading to a low training error but a high test error.

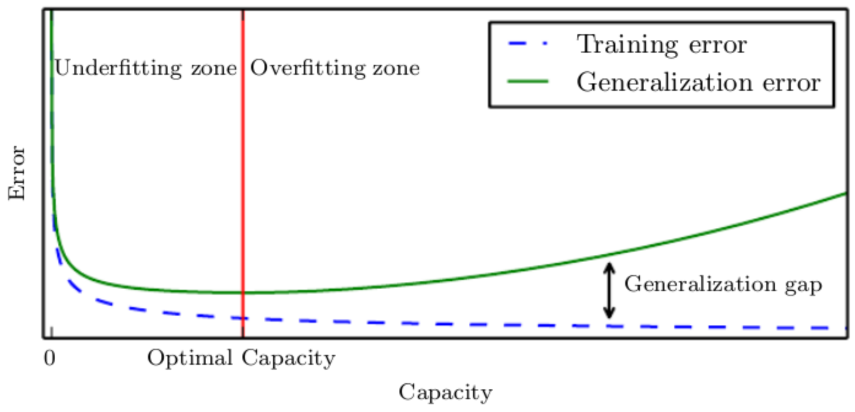


Figure 2.2. *Illustration of the relationship between capacity and error from [Goodfellow et al., 2016]. Since we only have access to training data, training and test error behave differently. At the left end of the graph, training error and generalization error are both high. This is the underfitting regime. As we increase capacity, training error decreases, but the gap between training and generalization error increases. Eventually, the size of this gap outweighs the decrease in training error, and we enter the overfitting regime, where capacity is too large, above the optimal capacity.*

To avoid overfitting to the training data, we can either restrict the hypothesis space to low capacity models, or more elegantly guide the optimization process towards low capacity models through *regularization*, which we explain in the next section.

2.1.6 Regularization

Regularization consists in defining a *penalty function*, called *regularizer*, to introduce in the learning problem, that will express a preference for specific functions in the hypothesis space. For instance, a common regularizer used is the *weight decay* [Krogh and Hertz, 1992] Ω:

$$\Omega(\theta) := \|\theta\|_2^2, \tag{2.6}$$

defined as the squared L_2 norm of the parameters, that influences the optimization towards small weights and reduces overfitting. We can introduce the weight decay regularization in the training process by considering

$$\mathcal{J} := \mathcal{L} + \lambda\Omega, \tag{2.7}$$

as our regularized loss function for the ERM, which is then called *Regularized ERM (RERM)*. In the above equation, we add the parameter λ to control the strength of the regularization.

2.1.7 Types of Models

The most simple Machine Learning algorithms are based on linear models, such that $\mathcal{H} = \{f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + b, \mathbf{w} \in \mathbb{R}^{d_1}, b \in \mathbb{R}\}$, with the most influential approach being the Support Vector Machine (SVM) [Boser et al., 1992, Cortes and Vapnik, 1995]. These models are easy to learn since the training can be easily transformed into a *convex optimization problem* thanks to a formulation based on a maximum margin principle:

$$\hat{\mathbf{w}}, \hat{b} = \arg\min_{\mathbf{w},b} \|\mathbf{w}\|^2, \quad \text{subject to} \quad y_i(\mathbf{w}^\top \mathbf{x}_i - b) \geq 1, \quad \forall i \in \{1, \dots, N\}. \tag{2.8}$$

To overcome the limitation of the representational capacity of linear models, which is most apparent when dealing with high dimensional data such as images, one key innovation from SVM is the *kernel trick* [Boser et al., 1992]. The main idea comes from observing that linear models can be rewritten as a dot product between examples:

$$\mathbf{w}^\top \mathbf{x} + b = b + \sum_{i=1}^{N} \alpha_i \mathbf{x}^\top \mathbf{x}_i. \tag{2.9}$$

The kernel trick consists in using a non-linear function ϕ to represent the data samples $\mathbf{x}$ by $\phi(\mathbf{x})$, and replacing the dot product by a *kernel function* $k(\mathbf{x}, \mathbf{x}_i) = \langle \phi(\mathbf{x}), \phi(\mathbf{x}_i) \rangle$. The kernel trick is powerful since it allows to learn a model that is *non-linear* as a function of $\mathbf{x}$ using *efficient convex optimization methods*, but the mapping ϕ and the associated kernel are usually handcrafted or generic. The computations also become dependent on the number of training examples, which can be highly expensive when the training dataset is large. *Deep Learning* represents another family of methods that allow to overcome these limitations by learning efficiently non-linear models through *backpropagation* and *stochastic gradient descent*.

2.2 Deep Learning

Simple machine learning algorithms generally struggle with high-dimensional data, such as natural images in practice, since the number of possible configurations for the model increases exponentially with the number of dimensions. This is known as the *curse of dimensionality* in Machine Learning [Goodfellow et al., 2016]. These high-dimensional data often lie in specific and complicated regions of the representation space. For instance, sampling random values from a uniform distribution for each pixel will result in an image very different from a real photo, which is illustrated in Figure 2.3.

All these problems motivated the design of more complex models to generalize to central applications such as computer vision. The strategy of Deep Learning is to *learn* a non-linear mapping ϕ providing a representation of the data, instead of using generic or handcrafted functions with the kernel trick.

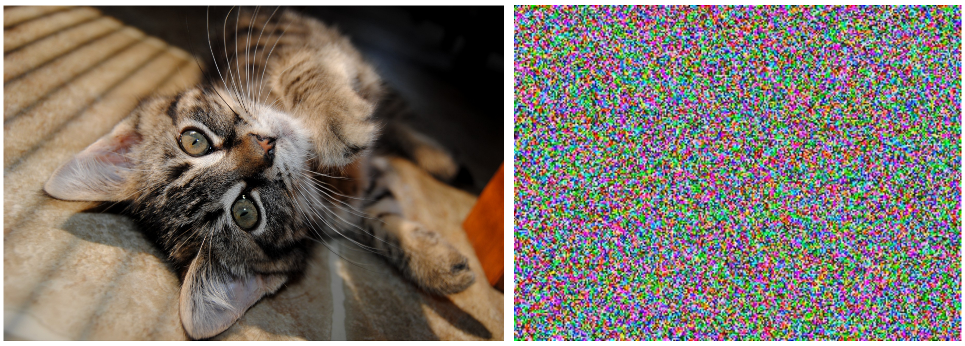


Figure 2.3. *Although there is a non-zero probability to generate a real photo by randomly sampling values for pixels, we never actually observe this in practice. This suggests that natural images lie in specific and narrow regions of the image space, which can be difficult to represent with a simple model.*

2.2.1 Simple Neural Networks

Deep Learning architectures are composed of several layers of *artificial neurons*. By stacking layers and adding more units within layers, deep neural networks can represent complex, *non-linear* functions.

2.2.1.1 Artificial Neurons

A neuron is represented as a single *parametric function* $f_{\mathbf{w}} : \mathbb{X} \to \mathbb{R}$ which takes inputs $\mathbf{x} \in \mathbb{X}$ and applies an *activation function* $\varphi : \mathbb{R} \to \mathbb{R}$:

$$f_{\mathbf{w}}(\mathbf{x}) = \varphi\left(\mathbf{w}^\top \mathbf{x}\right). \tag{2.10}$$

Artificial neurons were popularized by the *Perceptron algorithm* [Rosenblatt, 1958] for supervised binary classification, which used a *step function* as activation: $\varphi = \mathbb{1}_{\mathbb{R}_+}$. They are at the basis of most deep learning architectures.

2.2.1.2 Multilayer Perceptrons

Multilayer Perceptrons (MLP) are the simplest deep learning models, composed of multiple *layers* of neurons. The number of neurons in a layer determine its *width*, *i.e.* its dimensionality. The first and last layers are respectively the *input layer* and *output layer*, and the dimensionality of the input and output data respectively define their widths. All the other layers are called *hidden layers*, and their respective widths

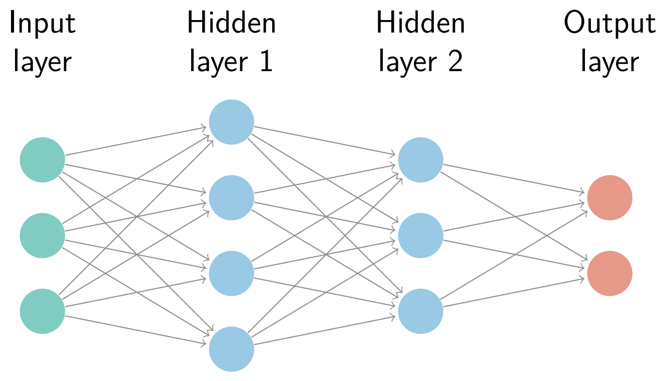


Figure 2.4. *Illustration of an MLP with two hidden layers such that input and output dimensions are respectively* $d_1 = 3$ *and* $d_2 = 2$*.*

are hyperparameters specified by design. Each layer describes a function, and the full model f_θ can be seen as the composition of all the functions:

$$f_\theta = f^{(d)}_{\theta^{(d)}} \circ \cdots \circ f^{(1)}_{\theta^{(1)}}, \tag{2.11}$$

where $f^{(i)}_{\theta^{(i)}}$ corresponds to the function describing the i-th layer. The resulting parameters θ of the whole model are defined as the combination of parameters $\theta^{(i)}$ of each layer. The number of layers gives the *depth* d of the model. An illustration of an MLP with two hidden layers is given in Figure 2.4. A specificity of this architecture is that for a given layer, each neuron is connected to every node of the next layer, leading to many connections, and these layers are also called *fully connected*. Non-linearities can be introduced between layers through *activation functions*, but they make the optimization of most loss functions a non-convex problem. Popular choice of activation functions for hidden layers are the Rectified Linear Unit (ReLU) [Jarrett et al., 2009] functions.

2.2.2 Gradient-Based Learning

2.2.2.1 Loss Functions and Predictive Modeling Problems

An important aspect of the design of neural networks is the choice of the loss function, which is tightly coupled with the choice of the activation function of the output layer. Suppose that the network provides a set of *hidden features* $\mathbf{h}$, from the last hidden layer. Depending on the *prediction problem* that the network must perform, the role of the output layer is to provide an additional transformation to the features $\mathbf{h}$ to follow the correct framing of the problem. The loss function then computes the error depending on the predictions and labels.

Regression Problems In a *regression problem*, the network must predict an n-dimensional vector $\mathbf{y} \in \mathbb{R}^n$, which can be seen as the mean of a Gaussian distribution. *Affine functions* are often used in this case:

$$\hat{\mathbf{y}} = \theta^{(d)\top}\mathbf{h} + \mathbf{b}. \tag{2.12}$$

The error between the output and ground-truth can then be measured using the *Mean Squared Error (MSE)* loss function:

$$\mathcal{L}_{MSE}(\mathbf{y}, \hat{\mathbf{y}}) = \|\mathbf{y} - \hat{\mathbf{y}}\|_2^2. \tag{2.13}$$

Binary Classification Problems In a *binary classification problem*, the model has to classify samples as belonging to one of two possible classes. The two classes are often labeled 0 and 1 for simplicity, such that $y \in \{0, 1\}$. These problems can be cast as predicting a Bernoulli distribution: $\hat{y} = P(y = 1)$. A popular choice is then to use a *logistic sigmoid activation function* defined by:

$$\sigma : \mathbb{R} \to \mathbb{R} \tag{2.14}$$

$$\sigma(x) = \frac{1}{1 + e^{-x}}, \tag{2.15}$$

and the output $\hat{y}$ is then computed as:

$$\hat{y} = \sigma\left(\theta^{(d)\top}\mathbf{h} + b\right). \tag{2.16}$$

It can be seen as the combination of a *linear layer* to compute $z = \theta^{(d)\top}\mathbf{h} + b$, also called *logit*, with a *sigmoid activation* to convert z into a probability. The usual loss function is then the *Binary Cross Entropy (BCE)*:

$$\mathcal{L}_{BCE}(y, \hat{y}) = -(y \cdot \log \hat{y} + (1 - y) \cdot \log \hat{y}). \tag{2.17}$$

Multi-Class Classification Problems In a *multi-class classification problem*, the model must predict a class for an input sample over C possible classes. For a sample with ground-truth class c, the target label is usually either the class number $y = c$ or a vector $\mathbf{y} \in \{0, 1\}^C$ such that $\mathbf{y}_c = 1$, also called *one-hot label vector*. These problems can be represented as predicting a probability distribution over a discrete variable with C possible values, also called a Multinoulli distribution. The output is a vector $\hat{\mathbf{y}}$ such that $\hat{\mathbf{y}}_c$ represents the predicted probability of class c: $\hat{\mathbf{y}}_c = P(y = c)$. A *softmax activation function* is often used for these problems:

$$\mathrm{softmax} : \mathbb{R}^C \to \mathbb{R}^C \tag{2.18}$$

$$\mathrm{softmax}(\mathbf{x})_i = \frac{\exp(\mathbf{x}_i)}{\sum_{j=1}^{C} \exp(\mathbf{x}_j)}. \tag{2.19}$$

The output $\hat{\mathbf{y}}$ is obtained such that:

$$\forall i \in [1, \ldots, C], \quad \hat{\mathbf{y}}_i = \text{softmax}\left(\theta^{(d)\top}\mathbf{h} + \mathbf{b}\right)_i. \tag{2.20}$$

Similarly to binary classification, the output layer is the combination of a linear layer to compute $\mathbf{z} = \theta^{(d)\top}\mathbf{h} + \mathbf{b}$ with a *softmax* activation. The *Cross Entropy (CE)* loss function is used for these problems:

$$\mathcal{L}_{CE}(\mathbf{y}, \hat{\mathbf{y}}) = -\sum_{i=1}^{C} \mathbf{y}_i \cdot \log \hat{\mathbf{y}}_i. \tag{2.21}$$

While the *affine*, *sigmoid* and *softmax* activation functions described above are the most common, neural networks can generalize to almost any kind of output layer. The total cost function used to train a neural network will often combine one of a loss function described above with additional regularization terms, such as the *weight decay* discussed in Section 2.1.6.

2.2.2.2 Optimization by Backpropagation

Deep Learning models are usually trained iteratively using *gradient-based* optimizers [Bottou et al., 1998, Kingma and Ba, 2015, Loshchilov and Hutter, 2019] on the training data. The loss functions being non-convex because of non-linear activation functions, the optimization problems can have several local and global minima. The optimization process aims to drive the loss function to a very low value, but has no convergence guarantees and is very sensitive to initialization.

The optimization algorithms typically compute the updates of the parameters based on an expected value of the loss function computed on randomly sampled subsets, or *minibatch*, of the training data. We usually call *batch size* the size of each minibatch in the training process. In practice, the choice of the batch size is important and is driven by several factors. On the one hand, large minibatch leads to more accurate gradients, and more different data seen for the same number of training iterations. On the other hand, a small batch size induces a noise that has also a regularization effect during training. However, the batch size is ultimately limited by the hardware capacities.

Stochastic Gradient Descent (SGD) [Bottou et al., 1998] and its variants, with the popular Adam [Kingma and Ba, 2015], are the most used optimization algorithms in machine learning in general and deep learning in particular. A crucial parameter in these algorithms is the *learning rate*, *i.e.* the step size in the direction given by the gradient. In practice, it is necessary to define a function, called *scheduler*, to decay the learning rate over the course of the training process. Popular choices of scheduler include linear, cosine or step functions.

Computing an analytical expression for the gradient of the model can be straightforward, but numerically evaluating it can be computationally expensive. The *back-propagation* algorithm [Rumelhart et al., 1986] is a simple procedure to efficiently compute an approximation of the gradients. The general idea of the algorithm is to recursively apply the *chain rule of calculus* to compute the derivatives of the model as a function formed by the composition of other functions. In practice, back-propagation is integrated and computed by the deep learning frameworks implementing *automatic differentiation*, such as Pytorch[1] or Tensorflow[2].

However, gradient-based learning methods with backpropagation also suffers from instabilities in the training process. During each training iteration, the updates of the weights are scaled depending on the current weights. With deep architectures, the scaling accumulates between layers because of backpropagation, and updates can become either very small *(vanishing gradients)* or very large *(exploding gradients)* [Hochreiter et al., 2001]. Vanishing gradients effectively prevent the weights from changing their values, while exploding gradients can result in an unstable training.

2.2.3 Deeper Architectures

2.2.3.1 Convolutional Neural Networks

Convolutional Neural Networks [LeCun et al., 1989], or *CNNs*, are a specialized kind of MLP for data with a grid-like topology and local information. CNNs are composed of *convolutional layers* which apply *convolution* on the input using different filters, or *kernels*, and pass the result to the next layer. For instance, with two-dimensional input I and kernel K, the convolution operation is defined as:

$$(I * K)(i,j) = \sum_m \sum_n I(m,n)K(i-m, j-n). \tag{2.22}$$

We usually assume that these kernels are 0 everywhere but a finite set of points. See Figure 2.5 for an illustration of the convolution operation. CNNs have the advantage of working with inputs of variable sizes and implement by design three important concepts:

Sparse connectivity: with a kernel smaller than the input, there are fewer parameters and connections than in MLP layers.

Parameter sharing: the same parameters in the kernel are used for different locations in the input feature map.

[1] https://pytorch.org/
[2] https://www.tensorflow.org/

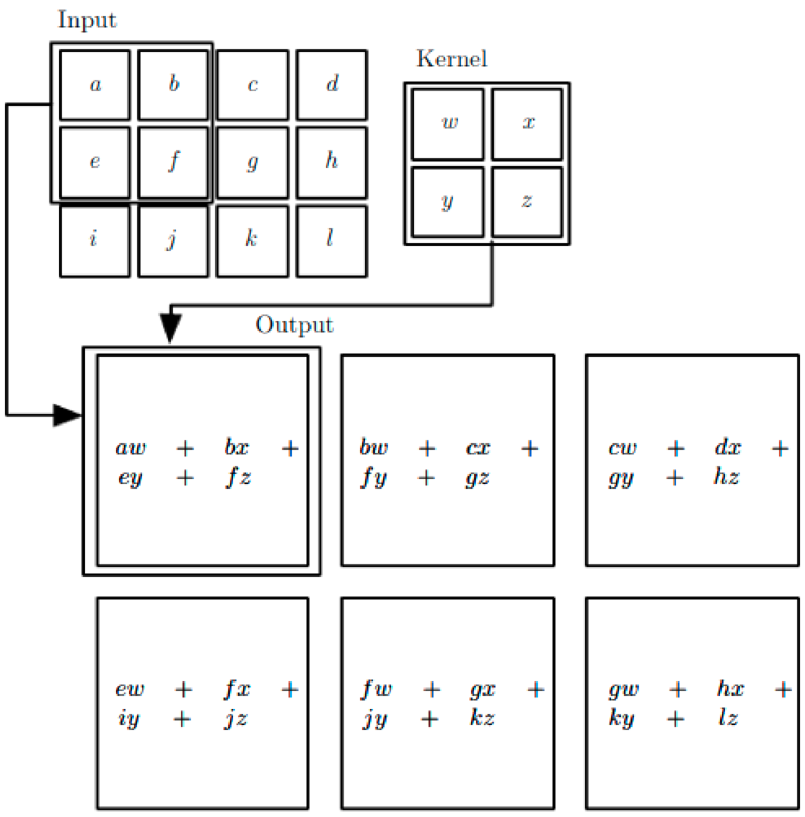


Figure 2.5. *An illustration of the 2D convolution operation used in CNNs from [Goodfellow et al., 2016]. The input and output are also called* feature maps*. A* kernel *matrix slides across the input feature map to compute the output feature map.*

Equivariance to translation: the convolution operation is invariant to a translation in the inputs since the kernel is translated at different location in the input feature map.

The CNN architectures have been at the heart of many advances in deep learning, their widespread application leading to deep CNNs reaching state of the art in many vision problems starting with AlexNet [Krizhevsky et al., 2017], with the architecture illustrated in Figure 2.6. Among the most popular ones are the Residual Networks or *ResNet* [He et al., 2016], that implement *residual connections* illustrated in Figure 2.7, allowing to go deeper in the architectures design and represent even more complex spaces, also called *embeddings*. Notably, residual connections allow to limit the problem of *vanishing gradients* [He et al., 2016]. They are now an important building block used in almost all recent architectures, notably the Transformers [Vaswani et al., 2017] detailed hereafter.

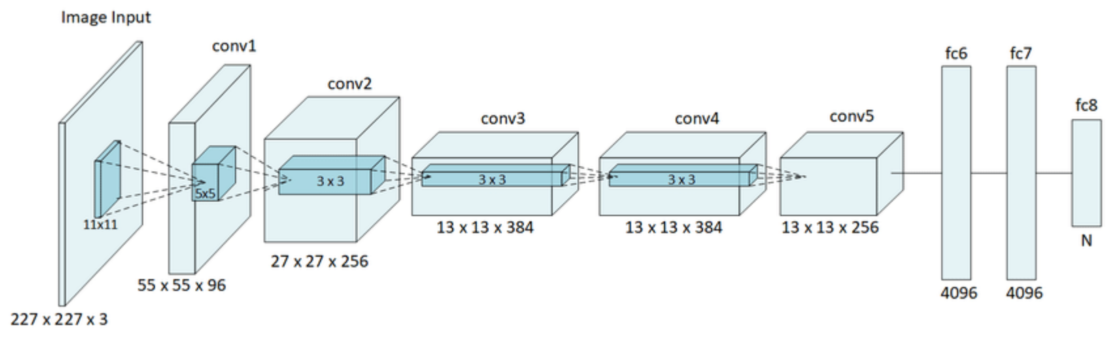


Figure 2.6. *Illustration of the AlexNet architecture, from [Krizhevsky, 2009]. Dimensions of the resulting features are written below each layers. The architecture is composed of 5 convolutional layers* (conv) *followed by 3 fully-connected layers* (fc)*.*

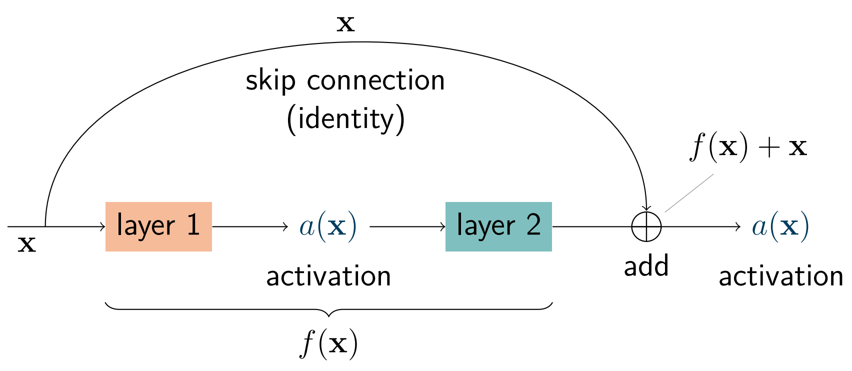


Figure 2.7. *Illustration of residual connection, or skip connection, inspired by [He et al., 2016]. This is an important building block of the ResNet architecture.*

2.2.3.2 Transformers

The *Transformer* architecture [Vaswani et al., 2017], illustrated in Figure 2.8, was originally introduced for Natural Language Processing (NLP), but has been recently used and adapted to Computer Vision tasks as well [Carion et al., 2020, Dosovitskiy et al., 2021]. Transformer are designed to represent *set-to-set functions*, *i.e.* they take a *fixed number of tokens* as input and outputs *the same number of processed tokens*. However, there is no particular association between the input tokens and the output tokens.

Before being used by the encoder and decoder, input and output tokens are converted to *embedding vectors* through simple *representation functions* such as a linear projection [Dosovitskiy et al., 2021]. For the model to make use of the order of the input sequence, information about the position of each token are injected through

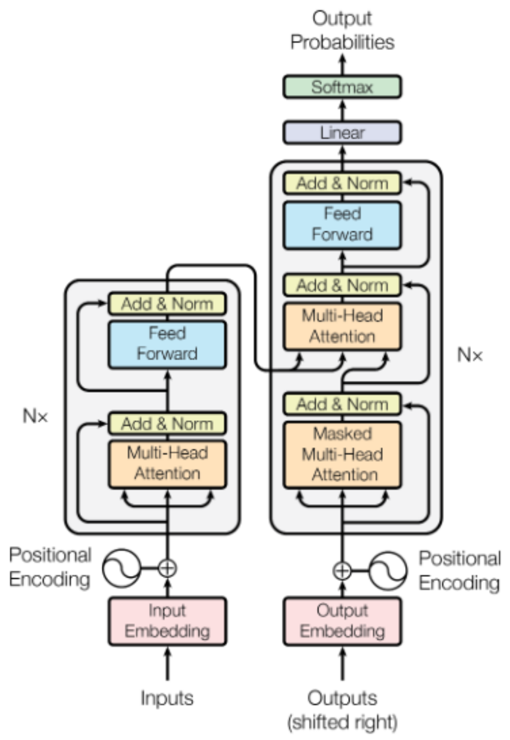


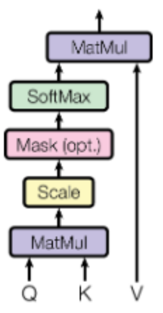
Figure 2.8. *The Transformer encoder-decoder architecture from [Vaswani et al., 2017] used for NLP tasks. The left and right halves respectively represent the* Transformer encoder *and* Transformer decoder *architectures.*

positional encoding. There are many choices of positional encodings, learned or fixed [Gehring et al., 2017]. The *original* Transformer architecture [Vaswani et al., 2017] is decomposed into two parts, an *encoder* and a *decoder*.

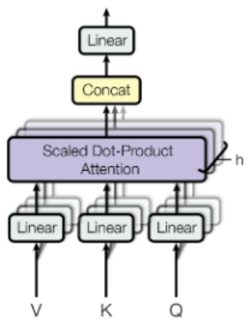
The **encoder** is made of a stack of several (N) identical layers, with each layer split into two sub-layers: a *multi-head self-attention*, illustrated in Figure 2.9b, and a *position-wise MLP*. Each sub-layers are surrounded by residual connections [He et al., 2016], followed by layer normalization [Ba et al., 2016]. The inputs of the layer are projected into three matrices called *Key* (K), *Value* (V), and *Query* (Q). The *attention operation* corresponds to a *scaled dot-product* between these three matrices:

$$Attention(Q, K, V) = \mathrm{softmax}(\frac{QK^\top}{\sqrt{d}})V, \tag{2.23}$$

with d the dimension of the keys K. The main objective of *attention* is to focus on the important elements of the task by allowing for an adequate weighting of the inputs. An illustration of the sequencing of operations in attention can be found in Figure 2.9a.



***(a)** Scaled Dot-product Attention*



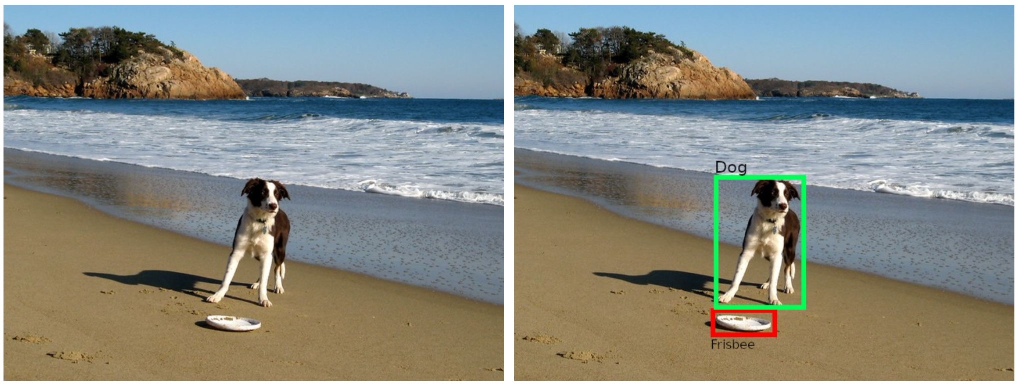
***(b)** Multi-head Attention*

***Figure 2.9.** Illustration of the sequencing of **(a)** the Scaled Dot-product Attention and **(b)** the Multi-head Attention, from [Vaswani et al., 2017]. These operations are computed between the keys K, queries Q and values V matrices.*

The projection and computation of the attention are repeated on several sub-layers *in parallel*, hence the name *multi-head*. Using different K, Q and V matrices for each head, allows considering different possible representations for the same inputs. The term *self-attention* is also used in the literature, since the attention function is computed between the input vectors. The outputs of the Multi-head attention layer is then sent to the *position-wise MLP*. In each layers of the encoder and decoder, the *position-wise MLP* is applied to each position separately but identically. While the architectures of the MLP are the same across different positions, they use different parameters.

The **decoder** part is also made of a stack of N identical layers. In addition to the two sub-layers in each encoder layer, the decoder inserts a third sub-layer, which performs multi-head attention over the output of the encoder stack. The multi-head attention is also modified by a *masking*, to prevent positions from attending to subsequent positions.

For Computer Vision, Transformers have been successfully adapted for Image Classification [Dosovitskiy et al., 2021] but also for Object Detection [Carion et al., 2020], and are now strong contenders to CNNs. We present these computer vision tasks in the next section.



(a) *Class: Dog* ***(b)*** *Object 1: Dog / Object 2: Frisbee*

Figure 2.10. *An illustration of the difference between* ***(a)*** *Image Classification and* ***(b)*** *Object Detection tasks. In Image Classification, the most probable class among a set of predefined class is predicted for the whole image, ignoring the fact that the image could represent multiple different objects. In Object Detection, all objects among all known class of objects appearing in an image must be identified and located through a bounding box.*

2.3 Deep Computer Vision

2.3.1 Image Classification

In Computer Vision, the task of *Image Classification* consists in predicting a class for a whole image, among multiple predefined classes. Classification is one of the go-to problems for evaluating and comparing Deep Learning models, thanks to several benchmarks from black and white handwritten digits in *MNIST* [LeCun et al., 1998] in the early days of Deep Learning, to tiny natural images in *CIFAR* [Krizhevsky, 2009], and now large-scale image recognition challenges with *ImageNet* [Russakovsky et al., 2015]. These benchmarks have allowed to quantitatively compare results and improvements between different methods and relatively to baselines. With the advent of large-scale datasets such as the full ImageNet dataset [Deng et al., 2009], containing 14M images divided into 21k classes [Ridnik et al., 2021], the Image Classification task on ImageNet serves as the main initialization of models before training on other downstream tasks [Ren et al., 2015, He et al., 2019]. Improving performance on the classification tasks often serves as a proxy metric for general applicability [Kornblith et al., 2019].

However, this task has its limitations. It does not take into account multiple different objects in a single image, nor their locations, illustrated in Figure 2.10. The model only learns general information from the images, rather than local information as it is done in more *dense tasks*, *i.e.* tasks that require a deeper understanding of the image, such as Object Detection.

2.3.2 Object Detection

The goal of *Object Detection* is to locate and classify every single object appearing in an image, from a pool of known classes objects. The *locating task* is done by predicting a *Bounding Box* around the object. Early *Object Detectors* were based on the computation of local features such as SIFT [Lowe, 2004] or HOG [Dalal and Triggs, 2005] along with more classic ML paradigms, but they are now largely outperformed by Deep Learning models. These more recent object detectors can be separated into three categories, of which we give a brief review below. We refer the reader to [Jiao et al., 2019, Zaidi et al., 2022] for more thorough surveys of modern deep-learning based object detection methods. They all use a *backbone* network as a *feature extractor*, typically a ResNet [He et al., 2016], then inserts specific additional networks. The backbone is usually pretrained on Image Classification or an unsupervised task [He et al., 2019], before *finetuning* the full network, *i.e.* updating the parameters through training, on the Object Detection downstream task. Illustrations of typical object detection models are given in Figures 2.11 and 2.12.

Similarly to Image Classification, Object Detection has several large scale benchmark datasets to compare improvements between methods. The most notorious are the *PASCAL VOC* [Everingham et al., 2010] and *MS COCO* [Lin et al., 2014] benchmarks.

Two-stage Detectors These networks [Girshick et al., 2014, Girshick, 2015, Ren et al., 2015, He et al., 2015, Dai et al., 2016, Lin et al., 2017a], illustrated in Figure 2.11 (a), have a separate module to generate *Region Proposals* [Uijlings et al., 2013]. They find an arbitrary number of objects proposals in the input image during the first stage, and then classify and refine their location in the second. The most representative detector from this category is Faster R-CNN [Ren et al., 2015], in which the first stage proposes candidate object bounding boxes using a *Region Proposal Network (RPN)*, and the second stage extracts features from each candidate box by the RoI (Region of Interest) pooling operation for the following classification and regression tasks. As these methods have two separate steps, they generally take more time to generate and process the region proposals, but they have strong detection performance.

One-stage Detectors These detectors [Redmon et al., 2016, Redmon and Farhadi, 2017, Liu et al., 2016, Lin et al., 2017b, Law and Deng, 2018, Zhou et al., 2019, Tian et al., 2019] classify and locate in a single shot using a dense sampling. They either predefine boxes, or *anchors*, of various scales and aspect ratios in the feature maps, that are then refined to locate and classify objects [Redmon et al., 2016, Redmon

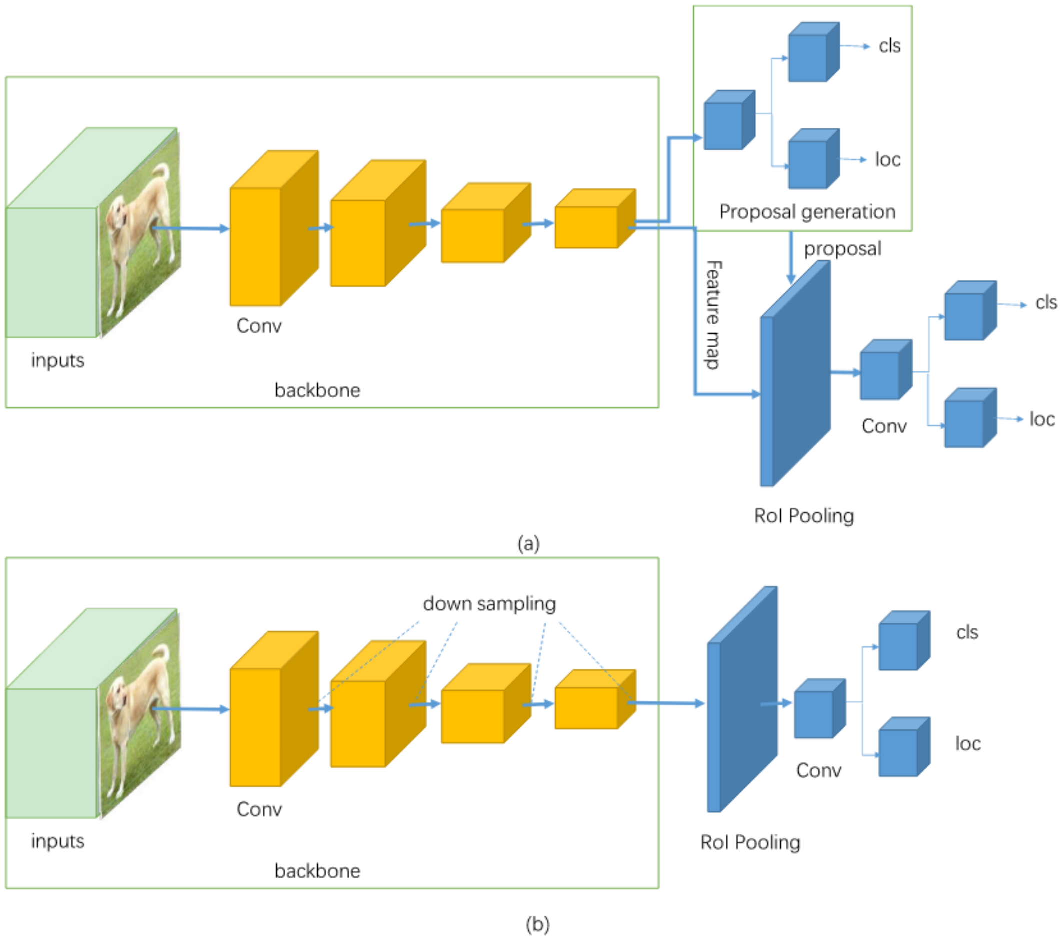


Figure 2.11. *Illustration from [Jiao et al., 2019] of the basic architectures of (a)* two-stage detectors *and (b)* one-stage detectors, *based on deep convolutional networks. The* cls *and* loc *outputs correspond respectively to the class and location predictions of objects. The detectors are composed of a* backbone network (in yellow), *with additional* detection-specific layers (in blue).

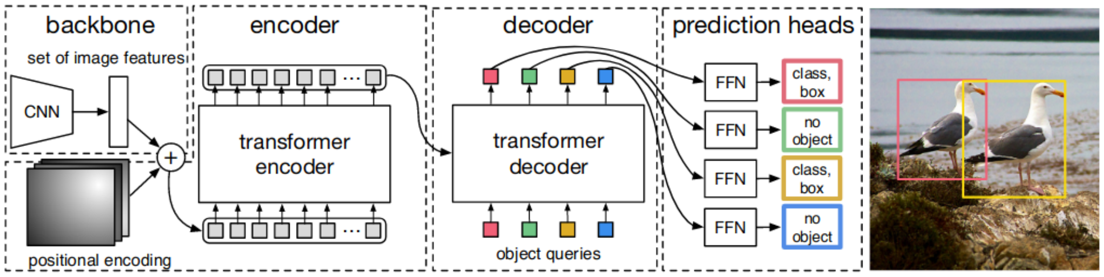


Figure 2.12. *Illustration from [Carion et al., 2020] of DETR, the original architecture based on Transformers for end-to-end object detection. The model is also composed of a* backbone network (top left)*, followed by detection-specific* transformer encoders and decoders*.*

and Farhadi, 2017, Liu et al., 2016], or predict bounding boxes according to reference points in the image [Law and Deng, 2018, Zhou et al., 2019, Tian et al., 2019]. One-stage methods have a much simpler design than two-stage ones, and have the benefit of real-time inference speed at the cost of lower detection performance.

Transformer-based Detectors Since the introduction of *Detection Transformers* (DETR) [Carion et al., 2020], transformer-based detectors [Carion et al., 2020, Zhu et al., 2021, Dai et al., 2021a, Meng et al., 2021, Wang et al., 2021e, Liu et al., 2022a, Yang et al., 2022, Li et al., 2022] have been an increasingly popular architecture. They allow end-to-end detection with a simpler overall architecture and without the need for hand-crafted heuristics such as the Non-Maximal Suppression (NMS) used in Faster R-CNN, or the predefined boxes locations in one-stage detectors. However, Transformers being set-to-set functions, these detectors rely on the Hungarian algorithm [Munkres, 1957] to find the optimal matching between predicted object proposals and the ground-truths, which can be costly to compute at each iterations.

In the next chapter, we now focus on the main problem of interest of this thesis, *learning from few labels*. Some methods and problems presented in this chapter will be revisited according to this setting.

Learning with few labels: an Overview 3

Contents

This chapter aims at discussing related work in the context of learning with limited labels. The main goal is to give a general overview of the recent background related to this topic while allowing us to present our contributions in the following chapters. The organization of the chapter is as follows. First, we present previous work on data augmentation *in Section 3.1, to alleviate label scarcity by increasing diversity. Then we focus on learning frameworks adapted to the limited data setting,* Meta-Learning *in Section 3.2, specifically designed for FSL,* Transfer Learning by Pretraining *in Section 3.3 which includes discussion on both* pretraining *and* fine-tuning *strategies, and finally* Semi-Supervised Learning with few annotations *in Section 3.4, that leverages unlabeled data along with few labeled data. In each section, we discuss applications for image classification and object detection separately.*

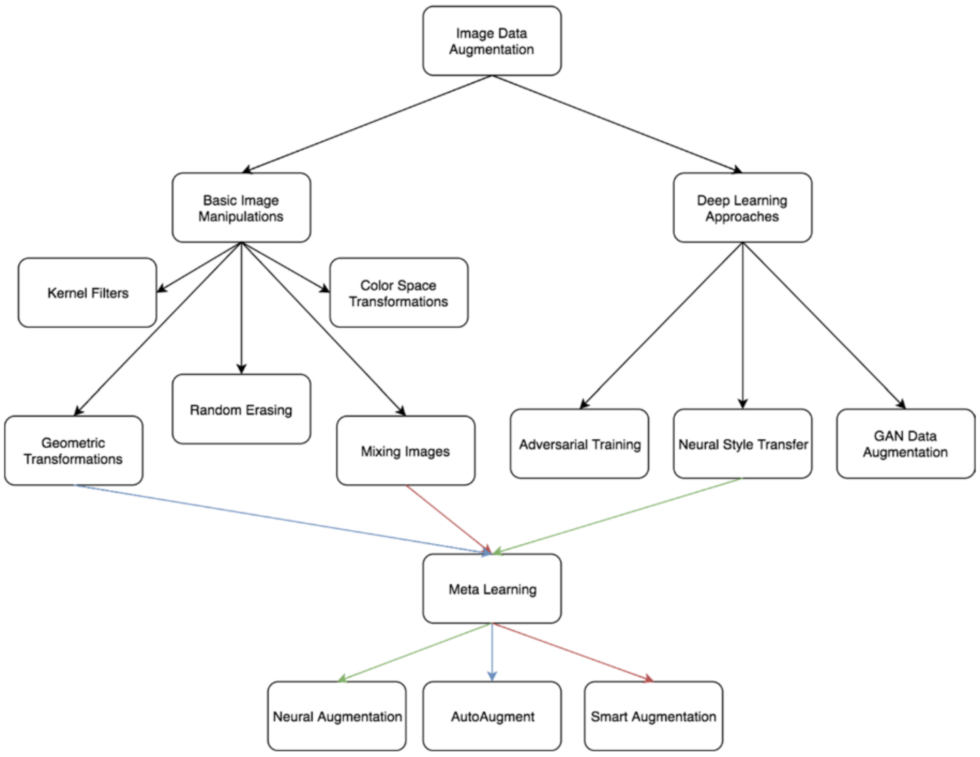


Figure 3.1. *A general taxonomy of data augmentations techniques from [Shorten and Khoshgoftaar, 2019].*

3.1 Data Augmentation with few images

The most direct way to deal with the problem of having too few samples, is to artificially increase the diversity in the training dataset through *data augmentation procedures*. Data augmentation has been a staple since the early days of deep learning [Yaeger et al., 1996]. It helps to guide the model into learning invariance properties or have a better estimate of the true data distribution. For a comprehensive overview of data augmentation techniques for computer vision applications, we refer the reader to [Shorten and Khoshgoftaar, 2019], and to [Wang et al., 2020b] for more details in the specific case of few-shot learning. A taxonomy of the different techniques is given in Figure 3.1.

3.1.1 Hand-crafted augmentations

Choosing the best augmentations for a given problem can require guidance from domain experts. Indeed, depending on the data and tasks to solve, some augmentations can be important while others can become inadequate [Cubuk et al., 2019]. Furthermore,

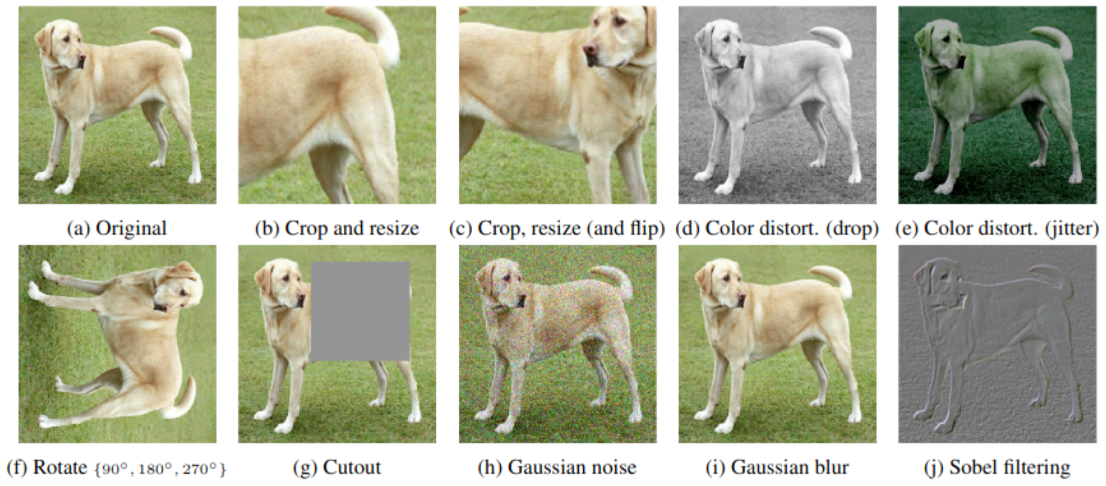

(a) Original
(b) Crop and resize
(c) Crop, resize (and flip)
(d) Color distort. (drop)
(e) Color distort. (jitter)
(f) Rotate {90°, 180°, 270°}
(g) Cutout
(h) Gaussian noise
(i) Gaussian blur
(j) Sobel filtering

Figure 3.2. *Illustration from [Chen et al., 2020b] of common hand-crafted augmentations.*

the magnitude of the transformation must be chosen carefully. Indeed, every image processing function can result in a label changing transformation at some distortion magnitude [Shorten and Khoshgoftaar, 2019]. Common hand-crafted augmentations are based on basic image transformations, illustrated in Figure 3.2, that can be grouped into *geometric transformations* such as flipping, cropping, rotation, translation, and *photometric transformations* with color jittering, gaussian blur or grayscale conversion [Taylor and Nitschke, 2018]. A counterintuitive but effective augmentation technique is *mixing images* together, illustrated in Figure 3.3. The combination can be done linearly [Zhang et al., 2017, Tokozume et al., 2018, Inoue, 2018], with non-linear functions [Summers and Dinneen, 2019, Takahashi et al., 2019], or through random cropping, patching and pasting training images together [Yun et al., 2019, Takahashi et al., 2019, Bochkovskiy et al., 2020, Li et al., 2021].

Another interesting procedure is the *Random erasing*, also called *CutOut*, augmentation [Zhong et al., 2020, DeVries and Taylor, 2017b] and illustrated in Figures 3.2 and 3.4. It works by randomly masking a patch in an image with random pixel values, preventing the model from overfitting on particular visual features in images.

3.1.2 Learning augmentations

In contrast to the above-mentioned transformations that are applied in the input space, augmentations can also be applied in the feature space [Chawla et al., 2002, DeVries and Taylor, 2017a]. To do so, recent methods learn several layers of neural networks for the transformation, using additional unlabeled data [Fu et al., 2015], or online during training [Hariharan and Girshick, 2017, Wang et al., 2018b, Schwartz et al.,

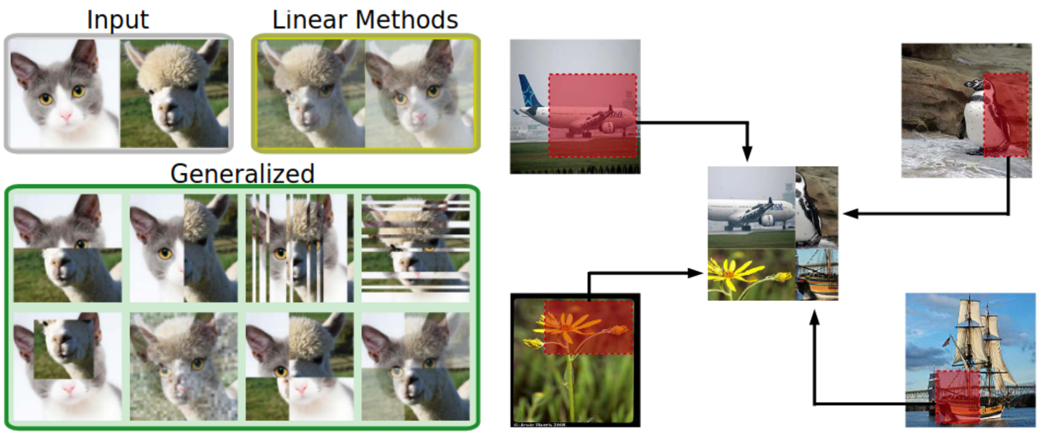


Figure 3.3. *Different ways of mixing images.* ***(left)*** *Non-linear vs linear mixing from [Summers and Dinneen, 2019].* ***(right)*** *Mixing with random patch and cropping [Takahashi et al., 2019].*

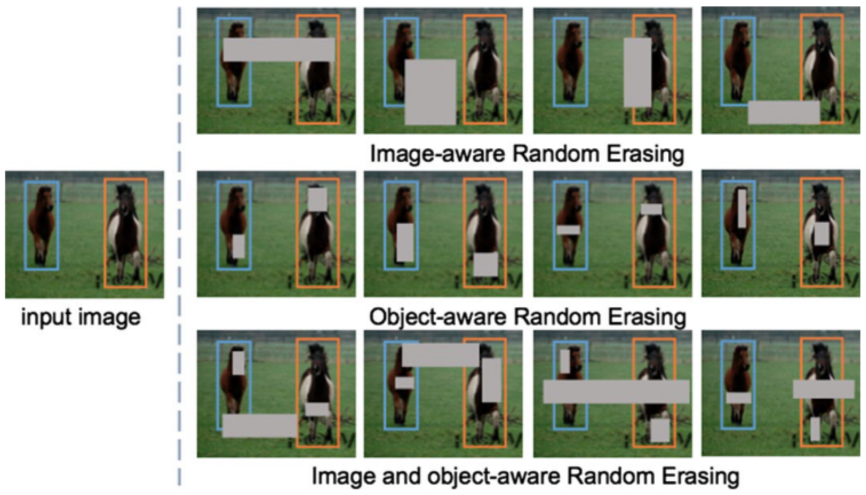


Image-aware Random Erasing
input image
Object-aware Random Erasing
Image and object-aware Random Erasing

Figure 3.4. *Example of random erasing on object detection task [Zhong et al., 2020]*

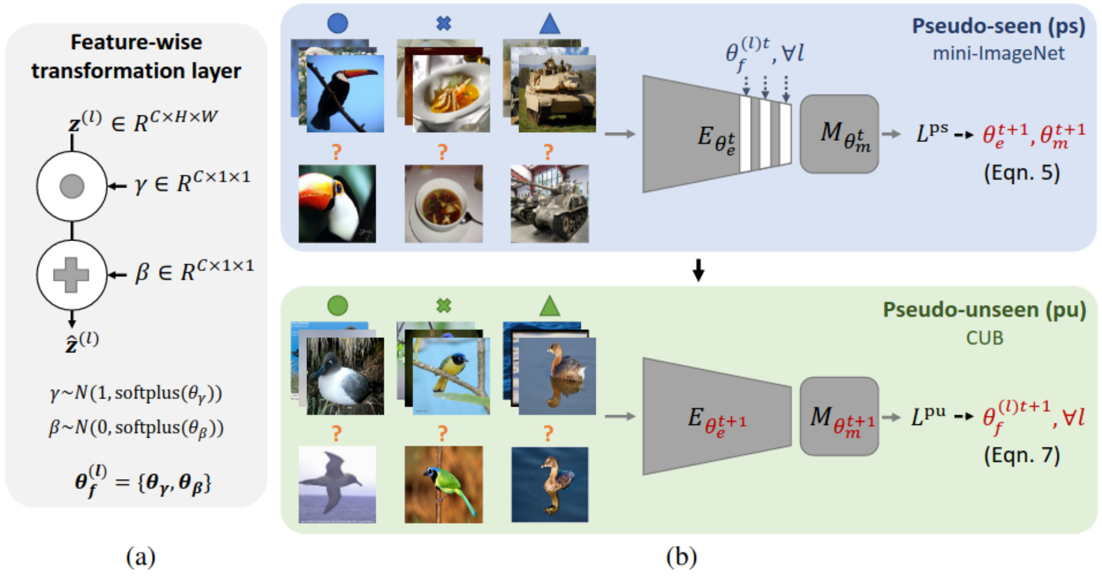

Feature-wise transformation layer
$\mathbf{z}^{(l)} \in R^{C\times H\times W}$
$\gamma \in R^{C\times1\times1}$
$\beta \in R^{C\times1\times1}$
$\hat{\mathbf{z}}^{(l)}$
$\gamma \sim N(1, \mathrm{softplus}(\theta_\gamma))$
$\beta \sim N(0, \mathrm{softplus}(\theta_\beta))$
$\theta_f^{(l)} = \{\theta_\gamma, \theta_\beta\}$
(a)
Pseudo-seen (ps)
mini-ImageNet
$\theta_f^{(l)t}, \forall l$
$E_{\theta_e^t}$
$M_{\theta_m^t}$
L^{ps}
$\theta_e^{t+1}, \theta_m^{t+1}$
(Eqn. 5)
Pseudo-unseen (pu)
CUB
$E_{\theta_e^{t+1}}$
$M_{\theta_m^{t+1}}$
L^{pu}
$\theta_f^{(l)t+1}, \forall l$
(Eqn. 7)
(b)

Figure 3.5. *Illustration of an augmentation learned in the feature space, from [Tseng et al., 2020]. (a) Learned feature-wise transformation layer. (b) The layers are inserted and learned during training, to simulate feature distributions extracted from tasks in various domains.*

2018, Tseng et al., 2020], with one of such method illustrated in Figure 3.5. However, it is very difficult to interpret the augmented data after feature space augmentations. Finally, if transformations are plausible in the image space, they provide greater benefit for improving performance and reducing overfitting than generic augmentations in the feature space [Wong et al., 2016].

3.1.3 Limitations of augmentation procedures

One important consideration when choosing augmentation procedures is the intrinsic bias in the initial dataset, since augmented samples are always based on the original samples. While augmentations are designed to alleviate specific biases and prevent overfitting by modifying limited datasets with characteristics of larger data, they cannot overcome all biases present in small datasets and will always reproduce some of them. An illustration of this problem is given in Figure 3.6. Over-extensive use of augmentations can even lead to more overfitting by emphasizing specific features [Shorten and Khoshgoftaar, 2019]. Therefore, one cannot only rely on data augmentations for few-shot learning.

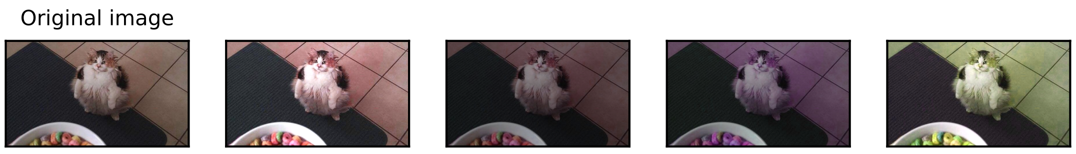


Figure 3.6. *In this example, augmenting the original image of a cat with only color transformations overly biases towards spatial characteristics of the cat. Learning a deep model on limited data with these color transformations will result in more overfitting than learning without augmentations.*

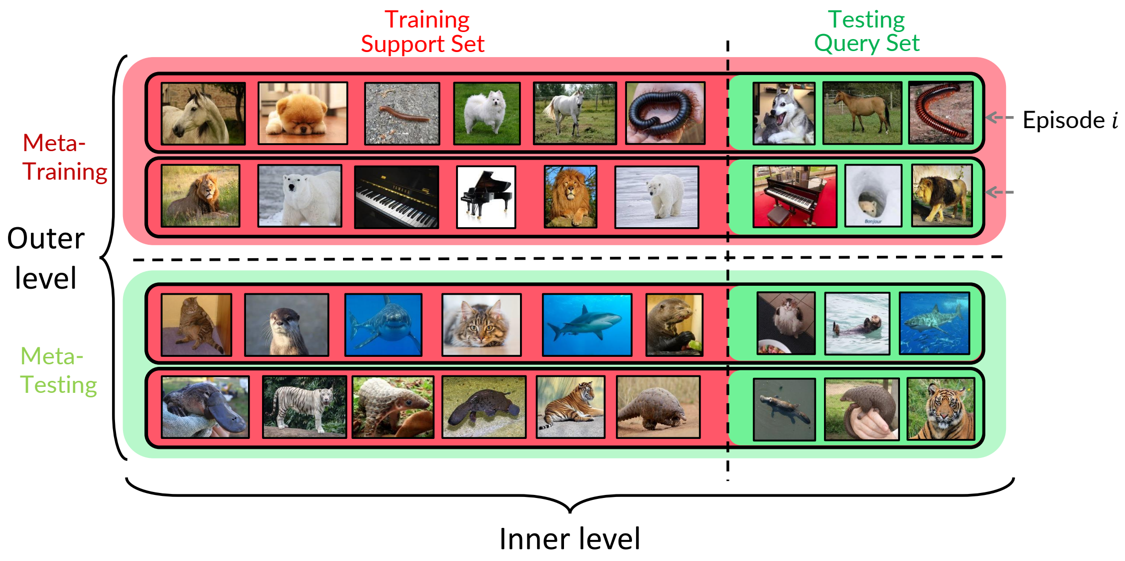


Figure 3.7. *A general illustration of meta-learning for FSC. The training dataset is separated into* episodes, i.e. *small learning tasks. In FSC, an episode is an N-way K-shot classification problem. We illustrate here 3-way 2-shot episodes. The meta-learning algorithm learns each task at the* inner-level *and between tasks at the* outer-level.

3.2 Meta-Learning

Meta-learning, or *learning to learn* [Thrun and Pratt, 1998], refers to the process of improving a learning algorithm over multiple learning *episodes*, *i.e.* a distribution of related tasks, as opposed to multiple data samples in conventional learning frameworks. Meta-learning is separated into an *inner-level* and *outer-level* learning phase. In the inner-level, a learning algorithm solves a *task*, such as image classification, defined by a small dataset and an objective function. In the outer-level, the *inner algorithm* is updated by an *outer algorithm* to improve an *outer objective*, such as generalization performance or learning speed, *i.e.* the number of iterations and samples required for convergence to a good solution.

More formally, we consider that the model learns over a distribution of *tasks* $P(\mathcal{T})$, also called *episodes* in practice, where each task $\mathcal{T}_t$ corresponds to K i.i.d. training observations called *support set* $\mathcal{S}_t = \{(\mathbf{x}_{(t,j)}, \mathbf{y}_{(t,j)})\}_{j=1}^{K} \in \mathbb{X}^K \times \mathbb{Y}^K$ and q i.i.d. testing observations called *query set* $\mathcal{Q}_t = \{(\mathbf{x}'_{(t,j)}, \mathbf{y}'_{(t,j)})\}_{j=1}^{q} \in \mathbb{X}^q \times \mathbb{Y}^q$, all generated by the corresponding distribution μ_t over the joint data space $\mathbb{X} \times \mathbb{Y}$. During *meta-training*, a task $\mathcal{T}_t$ is sampled from $P(\mathcal{T})$ and the model is trained *at the inner level* to learn the new task from the K support samples. The model is then improved *at the outer-level* by considering the test error on the unseen *query samples* from $\mathcal{T}_t$. In effect, the test error on episode $\mathcal{T}_t$ serves as the meta-training error of the meta-learning process. For *meta-testing*, new unseen episodes are sampled from $P(\mathcal{T})$ and the model's performance is measured on the query samples *after* learning from the K support samples. In practice meta-learning is usually done through a *bi-level optimization* procedure. We give an illustration of meta-learning episodes for *Few-Shot Classification (FSC)* in Figure 3.7.

While meta-learning has become a popular paradigm that successfully applied in areas spanning reinforcement learning [Alet et al., 2020], hyperparameter optimization [Franceschi et al., 2018] and neural architecture search [Liu et al., 2019], we will focus on the few-shot image recognition applications [Finn et al., 2017, Snell et al., 2017]. Meta-learning-based approaches are increasingly powerful to train CNNs on small datasets for many vision problems. By learning to solve many tasks with only few training samples, the meta-learned model can more easily adapt to a novel unseen task with few examples as well. We refer the reader to [Hospedales et al., 2021] for a more general presentation of deep meta-learning and its applications, with its respective taxonomy given in Figure 3.8, and to [Wang et al., 2020b] for a detailed review of the FSL applications. Note more details will be provided in Chapter 4 describing our first contribution on FSC. In the following sections, we detail related work applying meta-learning for Few-Shot Image Recognition and Object Detection.

3.2.1 Meta-Learning for Few-Shot Image Recognition

The most common application of meta-learning in computer vision is multi-class FSC, using cross-entropy as the inner and outer loss functions [Snell et al., 2017, Ravi and Larochelle, 2017, Li et al., 2017, Antoniou et al., 2019, Ye and Chao, 2021, Vinyals et al., 2016, Satorras and Estrach, 2018, Rusu et al., 2018, Lee et al., 2019, Yin et al., 2020, Bertinetto et al., 2019, Raghu et al., 2020]. Meta-learning methods are usually separated in *optimization-based* [Finn et al., 2017, Raghu et al., 2020, Oh et al., 2021], *model-based* [Santoro et al., 2016, Qiao et al., 2018a] or *metric-based* [Snell et al., 2017, Allen et al., 2019] categories that are briefly presented below.

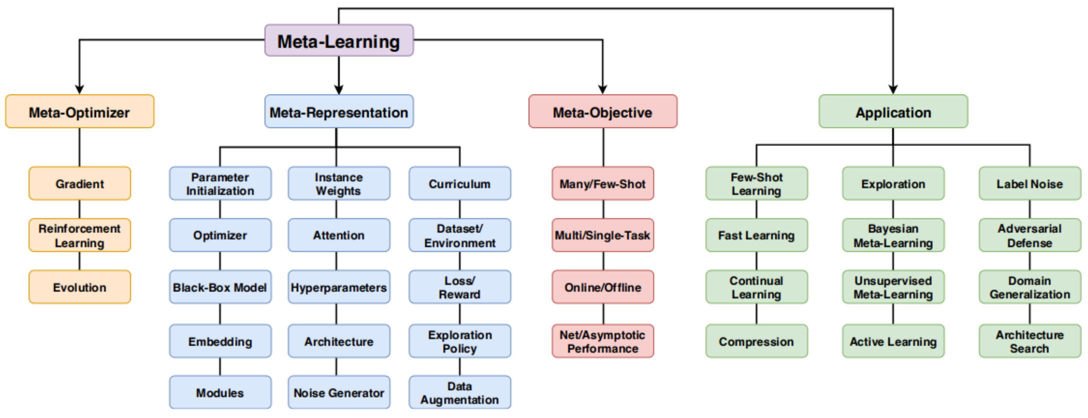
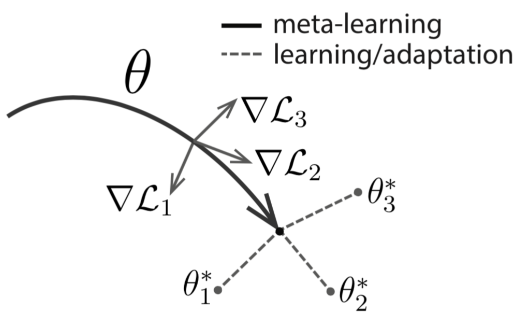
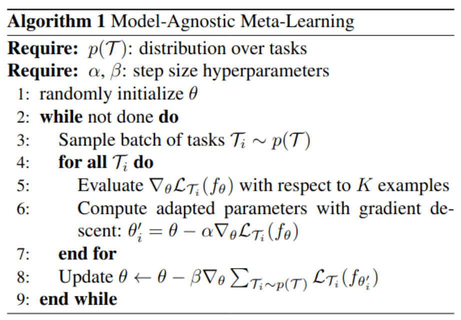


Figure 3.8. *A taxonomy of Meta-Learning and its applications going beyond FSL, from [Hospedales et al., 2021].*



(a)



```
Algorithm 1 Model-Agnostic Meta-Learning
Require: $p(\mathcal{T})$: distribution over tasks
Require: $\alpha, \beta$: step size hyperparameters
1: randomly initialize $\theta$
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:         Evaluate $\nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$ with respect to $K$ examples
6:         Compute adapted parameters with gradient de-
           scent: $\theta'_i = \theta - \alpha\nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$
7:     end for
8:     Update $\theta \leftarrow \theta - \beta\nabla_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta'_i})$
9: end while
```

(b)

Figure 3.9. *Illustrations from [Finn et al., 2017] of* ***(a)*** *the Optimization-based method MAML,* ***(b)*** *the corresponding* pseudo-code.

3.2.1.1 Optimization-based

Optimization-based methods are literally solving the meta-learning objective as an optimization problem. The most representative example is *Model Agnostic Meta-Learning (MAML)* [Finn et al., 2017], illustrated in Figure 3.9, which aims to learn initialization parameters for novel tasks such that a small number of inner steps produces a classifier that performs well. In this case, the optimization is done end-to-end by gradient descent, but other alternatives also considers SVM [Lee et al., 2019], Ridge Regression [Bertinetto et al., 2019] or recurrent networks [Ravi and Larochelle, 2017]. However, optimization through multiple inner-steps leads to several computation and memory challenges which make the models difficult to train [Antoniou et al., 2019] and to analyze theoretically [Finn and Levine, 2018, Arnold et al., 2021].

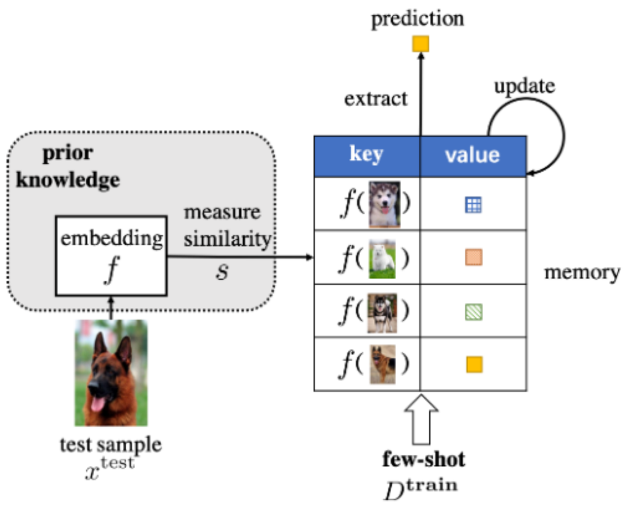


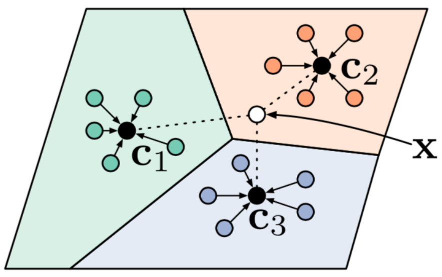
Figure 3.10. *Illustration of memory-augmented neural network from [Wang et al., 2020b]*

3.2.1.2 Model-based

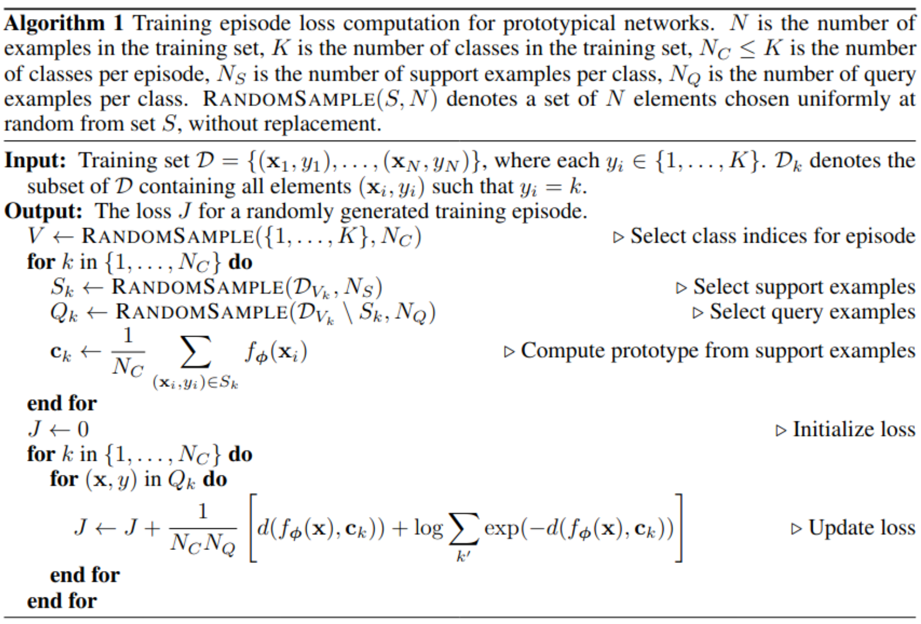
Model-based methods, also called *black-box* methods, wrap the inner learning step in a forward pass of a model designed specifically for learning with few training steps. Typical architectures include convolutional networks [Mishra et al., 2018], hyper-networks [Qiao et al., 2018a, Gidaris and Komodakis, 2018] or memory-augmented neural networks [Graves et al., 2014, Santoro et al., 2016, Kaiser et al., 2017, Munkhdalai and Yu, 2017, Cai et al., 2018] which are illustrated in Figure 3.10. While these methods enjoy simpler optimization without second-order gradients compared to optimization-based methods, they are usually less able to generalize to out-of-distributions tasks and asymptotically weaker as they struggle to embed large training set [Finn and Levine, 2018].

3.2.1.3 Metric-based

Metric-based methods [Snell et al., 2017, Allen et al., 2019, Vinyals et al., 2016, Koch et al., 2015, Sung et al., 2018, Bateni et al., 2020] perform non-parametric learning at the inner-level by comparing query points with support samples through similarity functions, and metric learning at the outer-level to find a good embedding space suited for comparison. The most popular representative is *Prototypical Networks (ProtoNet)* [Snell et al., 2017], illustrated in Figure 3.11, which uses Euclidean distance to compare query samples with class prototypes, represented by the average features of support examples for each class. The non-parametric inner-level computations make these methods simpler and faster to train than other methods, but they are also limited to



(a)



Algorithm 1 Training episode loss computation for prototypical networks. N is the number of examples in the training set, K is the number of classes in the training set, $N_C \leq K$ is the number of classes per episode, N_S is the number of support examples per class, N_Q is the number of query examples per class. RANDOMSAMPLE(S, N) denotes a set of N elements chosen uniformly at random from set S, without replacement.

```
Input: Training set $\mathcal{D} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$, where each $y_i \in \{1, \ldots, K\}$. $\mathcal{D}_k$ denotes the
  subset of $\mathcal{D}$ containing all elements $(\mathbf{x}_i, y_i)$ such that $y_i = k$.
Output: The loss $J$ for a randomly generated training episode.
  $V \leftarrow$ RANDOMSAMPLE$(\{1, \ldots, K\}, N_C)$                ▷ Select class indices for episode
  for $k$ in $\{1, \ldots, N_C\}$ do
    $S_k \leftarrow$ RANDOMSAMPLE$(\mathcal{D}_{V_k}, N_S)$                 ▷ Select support examples
    $Q_k \leftarrow$ RANDOMSAMPLE$(\mathcal{D}_{V_k} \setminus S_k, N_Q)$                ▷ Select query examples
    $\mathbf{c}_k \leftarrow \frac{1}{N_C} \sum_{(\mathbf{x}_i, y_i) \in S_k} f_\phi(\mathbf{x}_i)$      ▷ Compute prototype from support examples
  end for
  $J \leftarrow 0$                                                        ▷ Initialize loss
  for $k$ in $\{1, \ldots, N_C\}$ do
    for $(\mathbf{x}, y)$ in $Q_k$ do
      $J \leftarrow J + \frac{1}{N_C N_Q}\left[d(f_\phi(\mathbf{x}), \mathbf{c}_k)) + \log \sum_{k'} \exp(-d(f_\phi(\mathbf{x}), \mathbf{c}_k))\right]$   ▷ Update loss
    end for
  end for
```

(b)

Figure 3.11. *Illustrations from [Snell et al., 2017] of* ***(a)*** *the Metric-based method ProtoNet,* ***(b)*** *the corresponding* pseudo-code.

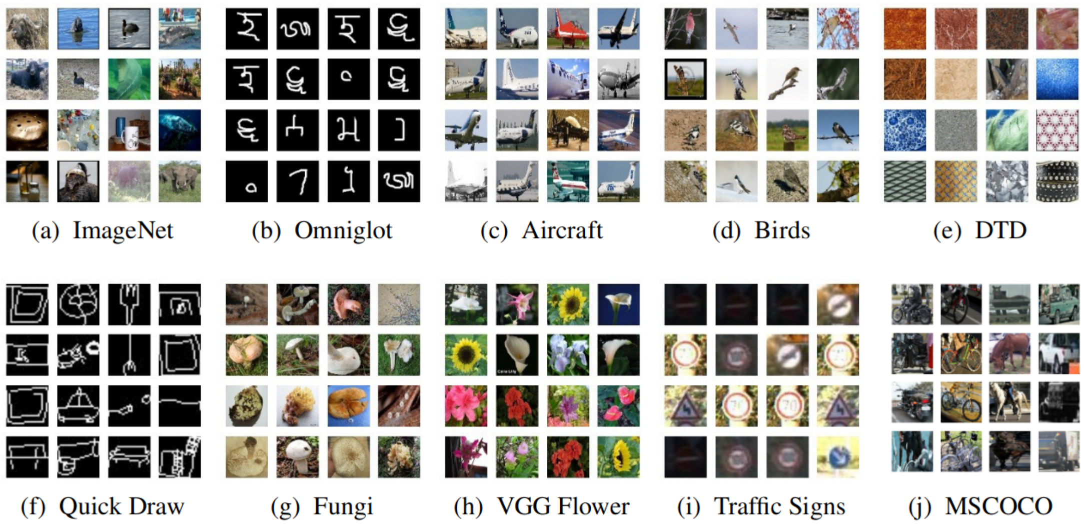
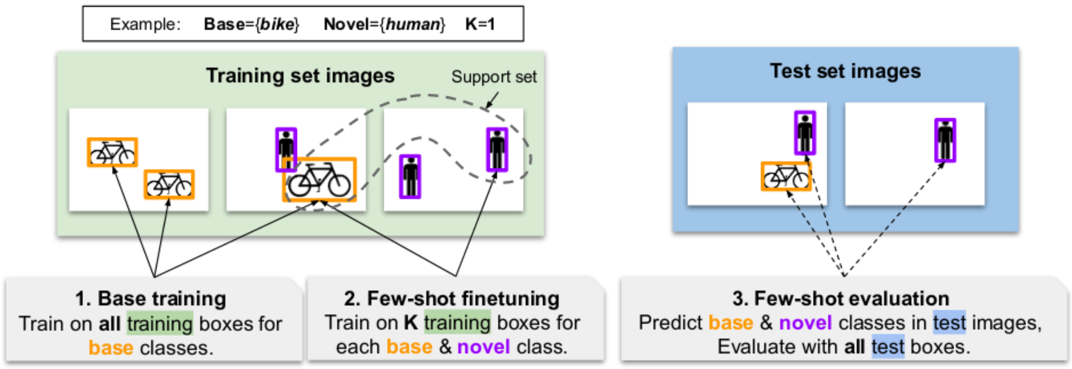


Figure 3.12. *Training examples taken from the various datasets forming Meta-Dataset from [Triantafillou et al., 2020].*

FSL applications compared to other meta-learning methods that could be potentially applied for other settings.

3.2.1.4 Datasets used

We present here common datasets used for FSC problems. The first dataset introduced for this problem is *Omniglot* [Lake et al., 2015], a dataset of 20 instances of 1623 characters from 50 different alphabets. Each image was hand-drawn by different people using Amazon Mechanical Turk. This dataset is still popular in the literature for benchmarks and has the advantage of being lightweight, but it is now considered too simple for modern few-shot learning methods, with state-of-the-art performance of about 99% of accuracy. For more complicated datasets, recent methods are comparing performance on *miniImageNet* [Ravi and Larochelle, 2017], *tiered-ImageNet* [Ren et al., 2018] or *CIFAR-FS* [Bertinetto et al., 2019] datasets. Both miniImageNet and tiered-ImageNet are constructed by taking subsets of the ImageNet dataset [Russakovsky et al., 2015]. The former consists of 100 randomly chosen classes and 600 images for each class, while the latter is chosen such that training classes are semantically unrelated to testing classes to be more complicated, and is composed of 779 165 images divided into 608 classes. *CIFAR-FS*, which stands for *CIFAR100 few-shot*, corresponds to classes from CIFAR-100 [Krizhevsky, 2009] randomly separated and with 600 images for each class, to obtain a dataset harder than Omniglot but lighter than miniImageNet. Recent work introduced *Meta-Dataset* [Triantafillou et al., 2020] for a large-scale benchmark consisting of data from 10 different image datasets. Examples from each



Example: Base={bike} Novel={human} K=1
Training set images
Support set
Test set images
1. Base training
Train on all training boxes for base classes.
2. Few-shot finetuning
Train on K training boxes for each base & novel class.
3. Few-shot evaluation
Predict base & novel classes in test images,
Evaluate with all test boxes.

Figure 3.13. *The Few-shot object detection protocol proposed by [Kang et al., 2019], as illustrated by [Huang et al., 2021]. The model is trained on* base classes, *then finetuned on a combination of* base and novel classes *from a limited* support set. *During testing, the model is evaluated on* base and novel class detection.

dataset are shown in Figure 3.12. This benchmark aims to introduce realistic class imbalance with varying number of classes and training set sizes, to test the robustness across the spectrum of low-shot learning and domain discrepancy.

Meta-learning approaches and the episodic formulation of the problem have steadily improved performance in FSC compared to early methods [Finn et al., 2017, Koch et al., 2015, Vinyals et al., 2016], leading to the recent interest in this line of work. However, the benefit over simpler methods based on transfer learning is also being challenged [Chen et al., 2020a, Tian et al., 2020c, Chen et al., 2021b], as we will see in Section 3.3.

3.2.2 Meta-Learning for Few-Shot Object Detection

Few-Shot Object Detection (FSOD) is the task of learning to detect new categories of objects in an image using only few training examples per class. The categories of objects are separated into two disjoint sets: the *base classes*, for which we have access to many training examples, and *novel classes*, for which we only have few examples, or shots, per class. We give an illustration of the framework introduced by [Kang et al., 2019] in Figure 3.13. With Meta-Learning, methods are trained on *query images*, *i.e.* the image to detect objects from, using annotated *support images*, *i.e.* reference examples of each class to detect. Support images generally have a single bounding-box around the object to detect and are randomly sampled from all base classes during training. Then, a predefined set of base and novel images are used during a *finetuning phase* and in the *evaluation phase*. An example of evaluation in one-shot object detection

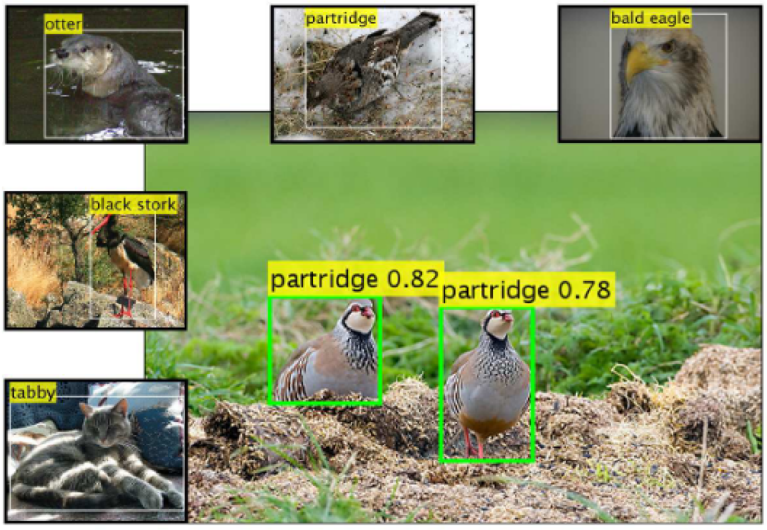


Figure 3.14. *One-shot Object Detection example from [Karlinsky et al., 2019].* Support examples *representing novel classes to detect are represented in the surrounding images. The* query image *is in the center, with detection results for the* partridge *category, one of the novel class.*

with support and query images is given in Figure 3.14. We refer the reader to [Huang et al., 2021] for a more detailed survey on FSOD.

3.2.2.1 Metric-based methods

Similarly to FSC, meta-learning methods for FSOD are based on *metric learning*. They compute prototypes for each object categories using support examples, and then classifies object proposals according to a similarity function to each representative. The prototypes can be obtained through linear operations [Wu et al., 2020b, Fan et al., 2020b, Han et al., 2022, Xiao and Marlet, 2020, Qiao et al., 2021] or learned with a separated module [Karlinsky et al., 2019, Kang et al., 2019, Li and Li, 2021, Yan et al., 2019], and the similarity can be computed through an explicit distance function [Karlinsky et al., 2019, Wu et al., 2020b, Qiao et al., 2021] or from an attention operation [Kang et al., 2019, Li and Li, 2021, Yan et al., 2019, Fan et al., 2020a, Fan et al., 2020b, Han et al., 2022, Xiao and Marlet, 2020]. Even though the large majority of the methods presented could technically be used *without finetuning*, by directly conditioning on the support examples at few-shot evaluation time, in practice, most works find it beneficial to finetune, and in fact many do not even report numbers without finetuning. It shows the necessity found in finetuning, and it is thus difficult to really assess the benefit of the methods without finetuning on target data.

Statistics	Train	Test
No. Class	800	200
No. Image	52350	14152
No. Box	147489	35102
Avg No. Box / Img	2.82	2.48
Min No. Img / Cls	22	30
Max No. Img / Cls	208	199
Avg No. Img / Cls	75.65	74.31
Box Size	[6, 6828]	[13, 4605]
Box Area Ratio	[0.0009, 1]	[0.0009, 1]
Box W/H Ratio	[0.0216, 89]	[0.0199, 51.5]

Table 3.1. FSOD-dataset *summary from [Fan et al., 2020a].*

3.2.2.2 Datasets used

We describe here the dominant FSOD benchmarks used in the literature. PASCAL VOC [Everingham et al., 2010] is one of the most popular smaller benchmarks for traditional object detection, and has naturally been adapted for FSOD [Kang et al., 2019]. The object categories are separated into *15 base classes* and *5 novel classes*, and three different splits between classes are usually considered, denoted *splits 1, 2 and 3*. For *base training*, training and validation images of the *base classes* from (PASCAL) VOC2007 and (PASCAL) VOC2012 are used (*trainval* sets). For *finetuning*, a fixed subset of the VOC2007 and VOC2012 *trainval* sets is taken as the support set. The benchmarks consider between 1 and 10 shots which correspond to between 15 and 150 base bounding boxes and between 5 and 50 novel bounding boxes. For evaluation, about 5k images from VOC2007 test set are used. Another popular dataset for object detection, *Microsoft Common Object in COntext (COCO)* [Lin et al., 2014], has also been adapted to FSOD [Kang et al., 2019]. The object categories are split between *20 novel classes*, shared with PASCAL VOC, and *60 base classes*. A subset of 5k images from COCO validation set are used for evaluation, denoted *val5k*, and the remaining training and validation images of COCO are used for training and finetuning, denoted *trainvalno5k*. A novel dataset, *FSOD* [Fan et al., 2020a], that we will refer to *FSOD-dataset* to avoid confusion, has been introduced specifically for the FSOD problem.

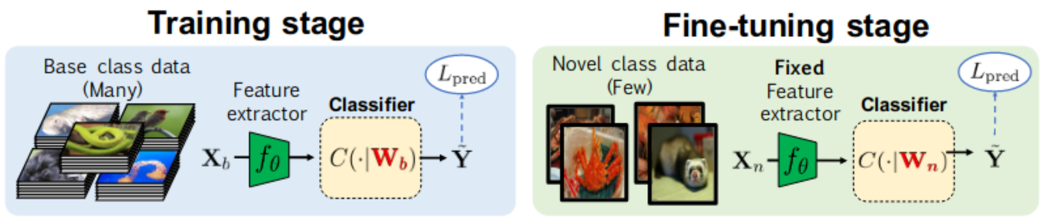


Figure 3.15. *Illustration of the general process of Transfer Learning from [Chen et al., 2019]. The training can be separated in* pretraining *and* fine-tuning *stages, in combination with other techniques.*

The dataset contains 1000 categories with 800 training classes and 200 testing classes, unambiguously separated. The total number of images is around 66k with about 182k bounding boxes. Detailed statistics are given in Table 3.1.

Similarly to FSC, recent work [Wang et al., 2020a, Bar et al., 2022] has shown that simply finetuning a classic detection model through supervised learning on the few-shot dataset after a pretraining phase on a bigger dataset achieves competitive results, compared with more complicated methods tailored for few-shot learning. Furthermore, even with meta-learning, the models learned are mainly doing *feature reuse* between tasks rather than a *rapid learning* of each task [Raghu et al., 2020]. These papers reignited research on *Transfer Learning* with limited labeled data on the *target dataset*, both for FSC and FSOD.

3.3 Transfer Learning by Pretraining

Transfer Learning is one of the oldest practice in deep learning to deal with novel unseen tasks with limited labeled data [Pan and Yang, 2009]. We focus in this section more specifically on pretraining-based methods, a classical learning paradigm which aims to reuse features learned on an *extensive source dataset* to a *potentially limited target dataset*.

3.3.1 Pretraining and Fine-tuning for Few-Shot Learning

Transfer learning can be divided into a first *pretraining stage*, in which we aim to learn a general representation of the data encoded in a *feature extractor*, and then a *fine-tuning stage*, during which the model is fully or partially updated using the few labeled data available for the target task. The general process in the case of FSC is illustrated in Figure 3.15. Even though early works on FSC had shown large improvements against this simple baseline [Vinyals et al., 2016, Ravi and Larochelle, 2017, Finn et al., 2017], later works have found that they achieve competitive performance when compared on common grounds, with same training details and also with deeper networks [Chen et al., 2019, Nakamura and Harada, 2019, Wang et al., 2019], while others have refined the fine-tuning stage specifically for FSL [Gidaris and Komodakis, 2018, Lifchitz et al., 2019, Dhillon et al., 2020, Tian et al., 2020c, Shen et al., 2021]. Several recent works are initializing meta-learning with a pretraining stage, showing that they also benefit from strong pretrained feature extractors [Rusu et al., 2018, Ye et al., 2020, Ye and Chao, 2021]. In [Tian et al., 2020c], authors have also investigated using different type of pretrained models, from classical supervised pretraining, to early *self-supervised pretraining* methods [He et al., 2020, Tian et al., 2020a], and obtained similar results between them, showing the potential of strong pretraining for FSL. More recent works in self-supervised learning are evaluating the fine-tuning performance on limited labeled data [Chen et al., 2020c]. Self-supervised learning have also been used in parallel during the fine-tuning stage to further boost FSL [Gidaris et al., 2019]. Using an *unsupervised pretraining stage* would also alleviate even more the need of labels.

3.3.2 Self-supervised Pretraining

Self-Supervised Learning (Self-SL) is an unsupervised learning procedure in which the data provides its own supervision to learn a good representation. The representation learned can then be transferred and fine-tuned on various downstream tasks such as classification or object detection. Several works have shown that a self-supervised pretrained model can outperform its supervised counterpart [Caron et al., 2020, Grill et al., 2020, Caron et al., 2021]. Self-SL is based on *pretext tasks*, *i.e.* unsupervised tasks that apply *transformations* to the data, or part of it, in order to provide its supervision.

3.3.2.1 Early pretext tasks

Early works proposed different pretext tasks from different set of augmentations to learn good data representations, such as *instance discrimination* [Dosovitskiy et al., 2016], *patch localization* [Doersch et al., 2015], *colorization* [Zhang et al., 2016], *solving jigsaw puzzle* [Noroozi and Favaro, 2016], *counting* [Noroozi et al., 2017] or *angle rotation prediction* [Gidaris et al., 2018]. These tasks are currently outperformed by *Contrastive Learning*, which represents now the majority of recent work in Self-SL.

3.3.2.2 Contrastive Learning

Contrastive Learning is a learning paradigm relying on *instance discrimination* using a pair of *positive views* from the same input contrasted with all other negative instances in the batch of images, called *negatives* [Oord et al., 2018, Wu et al., 2018, He et al., 2020, Chen et al., 2020b, Misra and Maaten, 2020, Grill et al., 2020, Caron et al., 2020, Chen et al., 2020c, Chen et al., 2021a, Denize et al., 2023]. The popular InfoNCE loss function [Oord et al., 2018] is widely used for contrastive learning:

$$\mathcal{L}_{\text{InfoNCE}} = -\frac{1}{N}\sum_{i=1}^{N}\log\left(\frac{\exp(\mathbf{z}_i \cdot \mathbf{z}_i'/\tau)}{\sum_{j=1}^{N}\exp(\mathbf{z}_i \cdot \mathbf{z}_j'/\tau)}\right). \tag{3.1}$$

This loss function computes the pairwise similarity scaled by a temperature parameter τ from a batch of feature vectors $\mathcal{B} = \{(\mathbf{z}_i, \mathbf{z}_i'), i \in N\}$, and maximizes agreement between positive pairs of features $(\mathbf{z}_i, \mathbf{z}_i')$ while pushing negative pairs $\{(\mathbf{z}_i, \mathbf{z}_j'), j \neq i\}$ away. However, InfoNCE [Oord et al., 2018] requires a large amount of negative instances to be effective [Wang and Isola, 2020]. While most methods use a huge batch size [Chen et al., 2020b, Chen et al., 2020c, Chen et al., 2021a, Tian et al., 2020b], the popular MoCo [He et al., 2020, Chen et al., 2020d] maintains a memory buffer of representations to keep a high number of negatives along with a small batch size. *Hard negatives*, *i.e.* negative samples that are difficult for the model to separate from positive ones, are the most important for contrastive learning [Cai et al., 2020], but they are also potentially harmful to the training because of the *class collision* problem [Cai et al., 2020, Wei et al., 2021a, Chuang et al., 2020], where semantically close instances can be used as negatives in the loss computation, which damages the quality of the representation learned. Several samplers of negatives have been proposed to alleviate this problem [Dwibedi et al., 2021, Wang et al., 2021a, Hu et al., 2021]. Other popular methods are based on a siamese architecture, *i.e.* using two copies of the model that are jointly updated, to perform contrastive learning without the use of negatives by ensuring consistency between the outputs of the two models [Grill et al., 2020, Caron et al., 2020, Caron et al., 2021, Zbontar et al., 2021, Bardes et al., 2022]. Recent works [Zheng et al., 2021, Denize et al., 2023] have also tackled the class collision problem by introducing the *relational aspect* between instances.

3.3.3 Pretraining and Fine-tuning for Object Detection

Using pretrained models then finetuning is a popular choice to deal with scarce data settings in Object Detection, since the models can be difficult to train from a random initialization. Figure 3.16 represents a taxonomy of pretraining methods for Few-Shot Object Detection.

3.3.3.1 The use of pretraining in Object Detection

Backbone pretraining Object detectors trained in practice are rarely starting from a random initialization. The backbone of the models are usually pretrained on ImageNet [Russakovsky et al., 2015], before fine-tuning on the target dataset and detection

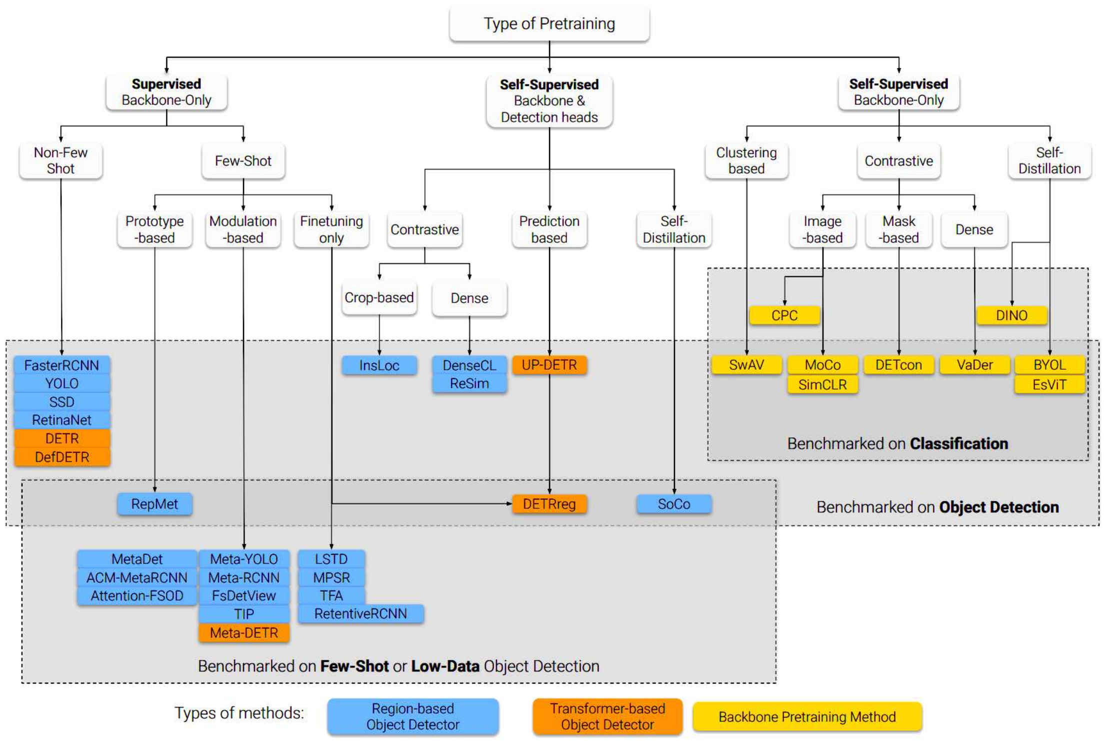


Figure 3.16. *A taxonomy of Few-shot Object Detection methods through the lens of pretraining, with Self-supervised learning, Meta-learning and Transfer learning methods [Huang et al., 2021].*

task [Ren et al., 2015, He et al., 2019]. Until recently, the standard approach was to pretrain the backbone in fully supervised learning. While pretraining from a *large general dataset* allows for faster convergence during fine-tuning on a similar dataset, both in terms of number of samples and iterations, it is less helpful for the localization task [He et al., 2019]. Recent *unsupervised* approaches have started investigating pretraining tasks tailored for object detection by imposing local consistency, either at the *pixel* or *region-level*, with the goal of helping the model to locate. These methods respectively propose to match in the representation space the features corresponding to the same location in the input space [O Pinheiro et al., 2020, Xie et al., 2021, Wang et al., 2021d], or ensure local consistency between features from regions in the image [Roh et al., 2021, Yang et al., 2021a, Xiao et al., 2021].

Pretraining the overall detection model Due to architecture mismatch between detection and classification models, only the backbone feature extractor can be initialized with pretraining while the detection-specific parts of the models must be learned from scratch. Few approaches in the literature have tackled this problem, by pretraining

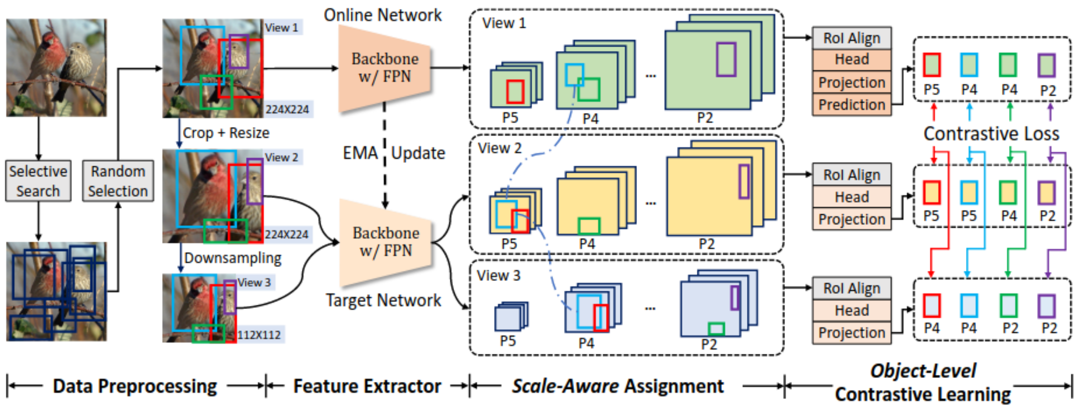


Figure 3.17. *Illustration of SoCo [Wei et al., 2021b]. The pretraining is specialized to two-stage architectures and use a contrastive learning objective.*

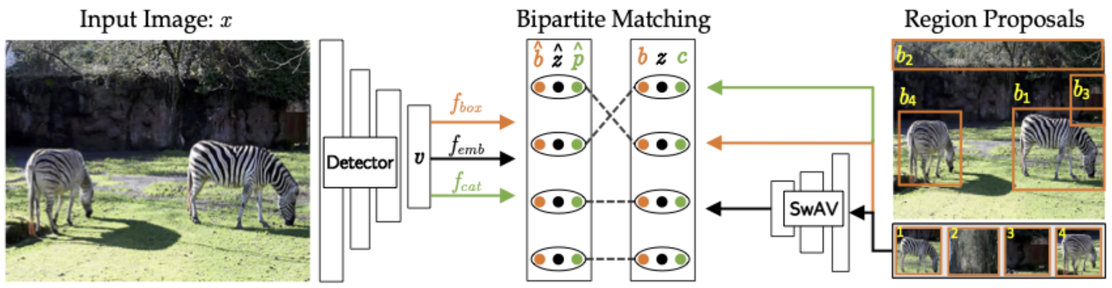


Figure 3.18. *Illustration of DETReg [Bar et al., 2022]. The detector must predict region proposals ($f_{box}(v) = \hat{b}$) given by an unsupervised algorithm [Uijlings et al., 2013], associated object embeddings ($f_{emb}(v) = \hat{z}$) obtained by an existing pretrained self-supervised backbone [Caron et al., 2020], and object scores ($f_{cat}(v) = \hat{p}$), fixed to* 1*.*

detection-specific parts along with the backbone [Wei et al., 2021b], or independently [Dai et al., 2021b, Bar et al., 2022]. SoCo [Wei et al., 2021b], illustrated in Figure 3.17, proposes a contrastive pretraining strategy inspired by BYOL [Grill et al., 2020], for the *overall* detection model but specialized to convolutional two-stage detectors. UP-DETR [Dai et al., 2021b] and DETReg [Bar et al., 2022], the latter more recent method illustrated in Figure 3.18, use a fixed pretrained backbone to extract features from patches and pretrain the detector *independently* by locating and reconstructing the features. Initializing detectors from an overall pretraining leads to even faster convergence and stronger performance with limited labels [Wei et al., 2021b, Bar et al., 2022].

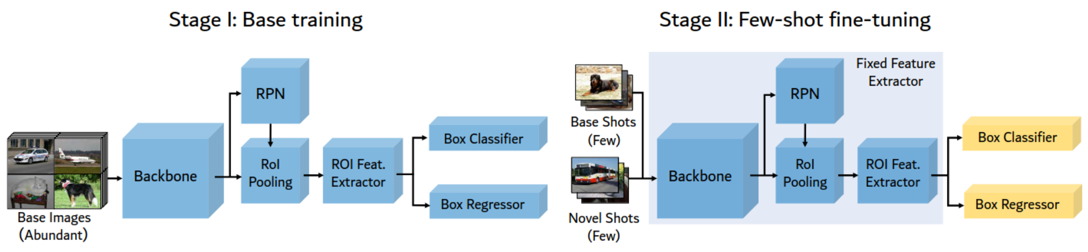


Figure 3.19. *Fine-tuning for FSOD in two-stage, from [Wang et al., 2020a]. Pretraining and fine-tuning detection models represents a strong baseline for FSOD, compared to more complex approaches presented in Section 3.2.2.*

3.3.3.2 Fine-Tuning Detectors

Fine-tuning for Few-Shot Object Detection Similarly to FSC, recent work have shown that simply fine-tuning traditional object detectors with only minor architecture modifications achieves strong results on FSOD [Wang et al., 2020a]. This fine-tuning approach is illustrated in Figure 3.19. Following works have then further refined the fine-tuning stage for FSOD [Wu et al., 2020a, Fan et al., 2021].

Fine-tuning for Low-shot Object Detection Instead of having a distinction between *base classes* with many examples, and *novel classes* with few shots, *Low-shot Object Detection* (LSOD) assumes that the number of examples for *all* classes is limited. *Data scarcity* is generally simulated by considering a fraction of the labels of traditional object detection datasets. The most prominent benchmark is *mini*COCO, which uses random subsets of 1%, 5% or 10% of the full COCO dataset. Several pretraining methods for object detection report numbers in this setting to compare performance, since comparison of transfer learning on the full COCO dataset may be of limited significance due to the large-scale supervision available [Yang et al., 2021a, Wang et al., 2021d, Wei et al., 2021b, Bar et al., 2022].

Unsupervised pretraining methods presented in this section have the advantage of leveraging large unlabeled data available by providing an initialization of the models more adapted to the target task [Reed et al., 2022]. One can further leverage these unlabeled data after fine-tuning, through *semi-supervised learning*.

3.4 Semi-supervised learning with few annotations

Semi-supervised learning is a learning paradigm aiming to train models by using both labeled and unlabeled data. It can improve performance by using additional unlabeled

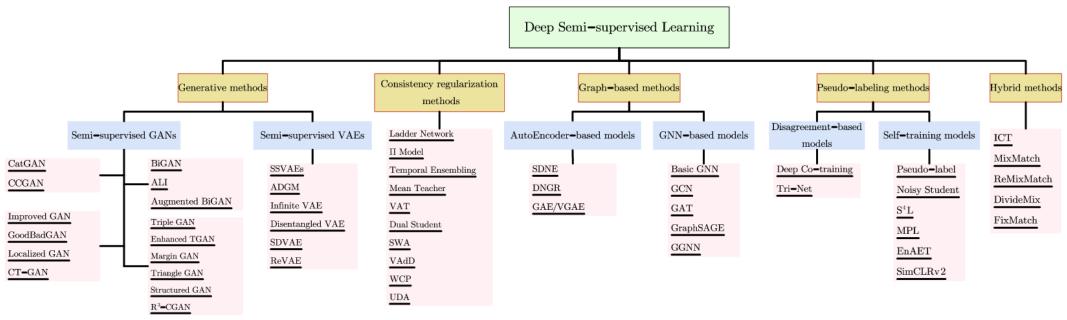


Figure 3.20. *A general taxonomy of semi-supervised methods with deep learning, from [Yang et al., 2021c].*

samples compared to supervised learning algorithms which can only use labeled data, alleviating the need for numerous labels [Oliver et al., 2018]. One can easily extend fully supervised or unsupervised algorithms to obtain semi-supervised ones. In this section, we will focus on presenting *few-annotation learning* applications for *image classification* and *object detection*, and limit the discussion to *consistency regularization* and *pseudo-labeling* as they represent the most successful and promising approaches in semi-supervised learning. We refer the reader to [van Engelen and Hoos, 2020, Yang et al., 2021c] for a general overview of the semi-supervised learning paradigm, with a taxonomy of deep learning-based methods given in Figure 3.20.

3.4.1 Few-Annotation Classification

In order to present and explain the key concepts in semi-supervised learning, we begin by the image classification problem. We are interested specifically in Few-Annotation Classification (FAC), in which the model has only access to few labeled examples along with a large number of unlabeled data.

3.4.1.1 Consistency Regularization

Consistency regularization methods are based on a prior assumption that realistic perturbations of data should not change the output of the model [Oliver et al., 2018]. These methods follow the general structure of *Teacher-Student*: the model learns as the *student*, and generates targets as the *teacher* to compute a *consistency regularization term* in the final loss function. The consistency constraint can be formally written as:

$$\mathbb{E}_{\mathbf{x}\in\mathbf{X}}\left[\mathcal{R}\left(f_\theta(\mathbf{x}), f'_{\theta'}(t(\mathbf{x}))\right)\right], \tag{3.2}$$

where $f_\theta(\mathbf{x})$ are the predictions of the student model for input $\mathbf{x}$, $f'_{\theta'}(t(\mathbf{x}))$ are the consistency target of the teacher model f' with parameters θ' for the same input transformed through t, and $\mathcal{R}$ measures the distance between the two vectors which is usually MSE or KL-divergence.

Consistency constraints can be considered at the input-level, by adding perturbations to input examples [Rasmus et al., 2015, Pezeshki et al., 2016, Sajjadi et al., 2016, Miyato et al., 2018b, Xie et al., 2020a], or at the network-level, by averaging the weights [Tarvainen and Valpola, 2017, Izmailov et al., 2018, Ke et al., 2019], the predictions [Laine and Aila, 2017] or even by dropping connections [Park et al., 2018, Zhang and Qi, 2020].

3.4.1.2 Self-Training

Self-Training differs from consistency regularization in that the former relies on *high confident predictions* and the latter on consistency between *rich transformations*. Self-Training methods can be based on *disagreement* [Zhou and Li, 2010] of predictions on unlabeled instances, from different views of the data [Qiao et al., 2018b] or between different models [Chen et al., 2018], or based on *Pseudo-labeling* by leveraging the model's predictions on unlabeled data [Grandvalet and Bengio, 2004, Lee et al., 2013, Xie et al., 2020b, Beyer et al., 2019, Chen et al., 2020c, Wang et al., 2021c, Pham et al., 2021]. Self-training has recently been at the heart of a lot of improvements in unsupervised training with the advent of large-scale training and architectures [Chen et al., 2020c, Caron et al., 2021].

3.4.1.3 Hybrid Methods

Hybrid methods [Verma et al., 2019, Berthelot et al., 2019, Li et al., 2020, Berthelot et al., 2020, Sohn et al., 2020a] combine ideas from both self-training and consistency regularization above-mentioned methods. They use input and network transformations along with entropy minimization [Berthelot et al., 2019], distribution alignment [Li et al., 2020, Berthelot et al., 2020, Verma et al., 2019] or pseudo-labeling [Sohn et al., 2020a]. These methods achieve currently state-of-the-art performance on various benchmarks in FAC, using as low as *a single* labeled example for each class with FixMatch [Sohn et al., 2020a] for which an illustration is given in Figure 3.21.

3.4.2 Semi-Supervised Object Detection for Few-Annotation Learning

First *Semi-Supervised Object Detection* (SSOD) methods were mostly based on *self-training* [Tang et al., 2016, Wang et al., 2018a, Gao et al., 2019], or *consistency regularization* [Jeong et al., 2019], before combining both. More recent methods

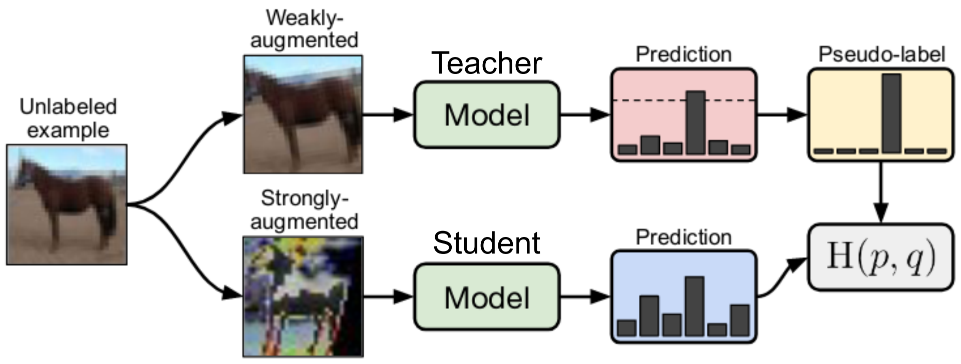


Figure 3.21. *Illustration from FixMatch [Sohn et al., 2020a], a popular hybrid method taking inspiration from both* pseudo-labeling *and* consistency regularization*. Two differently augmented views are created for each unlabeled example, with one view going to the model as the teacher to create a pseudo-label that will be used to supervise the model as the student with the other view.*

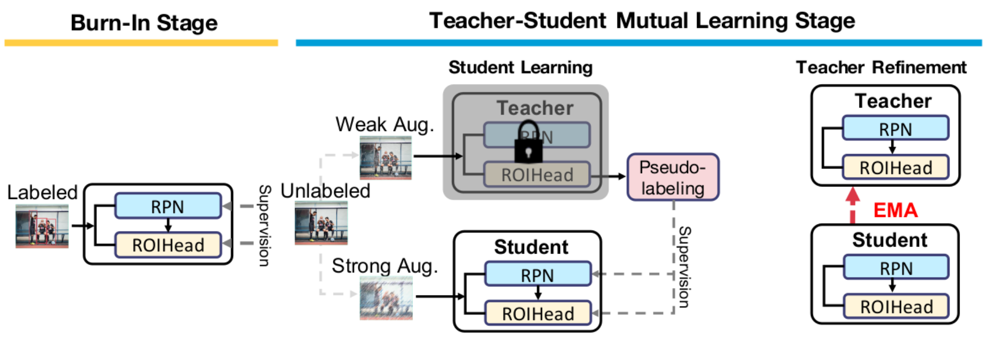


Figure 3.22. *Illustration of* Unbiased Teacher *[Liu et al., 2021], a popular approach in SSOD inspired by hybrid methods. Semi-supervised learning is initialized after a fully supervised training (*Burn-In stage*). The model is based on a* Teacher-Student *architecture, using two* differently *augmented views of the data, and* pseudo-labeling*. The teacher model is an independent copy of the student model, updated throughout training by the* Exponential Moving Average (EMA) *[Tarvainen and Valpola, 2017] of the student's weights.*

[Sohn et al., 2020b, Liu et al., 2021, Tang et al., 2021, Xu et al., 2021] are mainly taking inspiration from *hybrid methods* presented above, and focusing on *two-stage convolutional detectors*, presented in Section 2.3.2. They achieve strong performance in FAL for object detection by leveraging additional unlabeled examples along with LSOD training data. A popular recent approach is illustrated in Figure 3.22.

3.5 Towards the contributions of this thesis

As can be seen throughout this chapter, the problem of learning with limited labels in Deep Learning spans various topics, and can be tackled from different ways. The most obvious option is to work on the data itself to artificially increase its diversity, but it has its limits. The main research direction is to refine training schemes to deal with limited data or labels. *Meta-learning* with *episodic training*, is a popular framework for FSL and originally designed for this setting. Besides, it has found applications for other problems such as Reinforcement Learning [Finn et al., 2017] or Continual Learning [Gupta et al., 2020]. However, a strong theoretical justification of meta-learning algorithms matching with empirical results is still lacking. Inspired by recent theoretical results in *multitask representation learning* [Du et al., 2020, Tripuraneni et al., 2020, Wang et al., 2021b] published at the beginning of this thesis, our first research direction investigate the link between meta-learning and multitask representation learning algorithms to leverage strong learning bounds in this setting. Then as novel works challenging meta-learning in favor of fine-tuning approaches was published [Chen et al., 2019, Raghu et al., 2020, Wang et al., 2020a], we focus on leveraging unlabeled data along with few annotated data in object detection through pretraining and semi-supervised learning to develop the other contributions of this thesis.

4 Improving Few-Shot Classification with Meta-Learning through Multi-Task Learning

Contents

In this chapter, we introduce our first contribution which focuses on Meta-Learning algorithms for Few-Shot Learning (FSL), and more specifically their

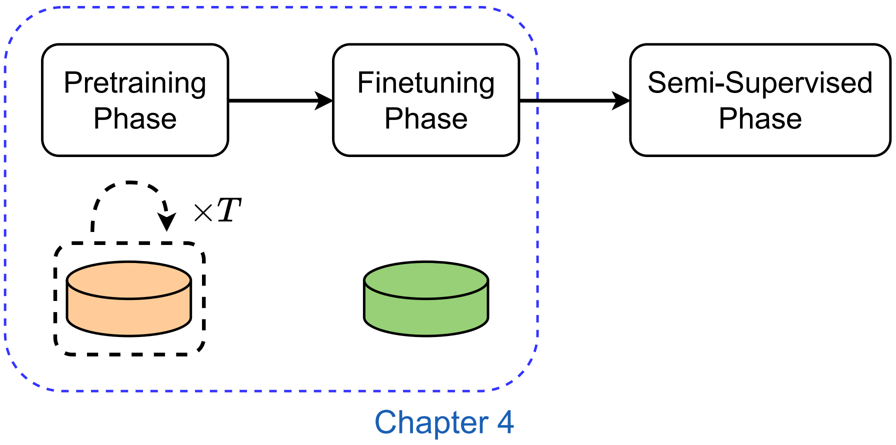


Figure 4.1. *Illustration of the position of our contributions from this chapter in the full training pipeline presented in Chapter 1. We are interested on Meta-Learning for Pretraining then Finetuning.*

link with Multi-Task Representation Learning (MTR) algorithms. We recall the positioning in the general training pipeline in Figure 4.1. As recent theoretical works in MTR theory have achieved efficient learning bounds for FSL, our goal is to leverage their theoretical settings to improve and better understand popular Meta-learning algorithms in practice. The organization of the chapter is as follows. We present the existing theoretical results for the MTR problem with their corresponding assumptions, and introduce considered meta-learning algorithms in Section 4.2. In Section 4.3, we investigate how metric-based and gradient-based algorithms behave in practice with respect to the identified assumptions and provide theoretical explanation to the observed behavior. We further show that one can force meta-learning algorithms to satisfy such assumptions through adding an appropriate spectral regularization term to their objective function. In Section 4.4, we provide an experimental evaluation of several state-of-the-art meta-learning methods and highlight the different advantages brought by the proposed regularization technique in practice for FSC tasks. Finally, we conclude the chapter in Section 4.5.

4.1 Introduction

As discussed in the previous chapter, meta-learning is an appealing solution for learning efficient models in the scarce data regime. Indeed, meta-learning aims at producing a model on data coming from a set of (meta-train) source tasks to use it as a starting point for learning successfully a new previously unseen (meta-test) target task. Each of these tasks can be composed of small sets of data samples.

The success of many meta-learning approaches is directly related to their capacity of learning a good representation [Raghu et al., 2020] from a set of tasks making it closely related to Multi-Task Representation learning (MTR). For this latter, several theoretical studies [Baxter, 2000, Pentina and Lampert, 2014, Maurer et al., 2016, Amit and Meir, 2018, Yin et al., 2020][1] provided probabilistic learning bounds that require the amount of data in the meta-train source task *and* the number of meta-train tasks to tend to infinity for it to be efficient. While capturing the underlying general intuition, these bounds do not suggest that all the source data is useful in such learning setup due to the additive relationship between the two terms mentioned above and thus, for instance, cannot explain the empirical success of MTR in few-shot classification (FSC) task. To tackle this drawback, two very recent studies [Du et al., 2020, Tripuraneni et al., 2020] aimed at finding deterministic assumptions that lead to faster learning rates allowing MTR algorithms to benefit from all the source data. Contrary to probabilistic bounds that have been used to derive novel learning strategies for meta-learning algorithms [Amit and Meir, 2018, Yin et al., 2020], there has been no attempt to verify the validity of the assumptions leading to the fastest known learning rates in practice or to enforce them through an appropriate optimization procedure.

In this chapter, we aim to use the recent advances in MTR theory [Tripuraneni et al., 2020, Du et al., 2020] to explore the inner workings of these popular meta-learning methods. Our rationale for such an approach stems from a recent work [Wang et al., 2021b] proving that the optimization problem behind the majority of meta-learning algorithms can be written as an MTR problem. Thus, we believe that looking at meta-learning algorithms through the recent MTR theory lens, could lead to better understanding the capacity to work well in few-shot regime. In particular, we take a closer look at two families of meta-learning algorithms, notably: gradient-based algorithms [Ravi and Larochelle, 2017, Nichol et al., 2018, Li et al., 2017, Oreshkin et al., 2018, Lee et al., 2019, Bertinetto et al., 2019, Park and Oliva, 2019, Chen et al., 2020e, Raghu et al., 2020] including MAML [Finn et al., 2017] and metric-based algorithms [Koch et al., 2015, Vinyals et al., 2016, Snell et al., 2017, Sung et al., 2018, Allen et al., 2019, Li et al., 2019, Simon et al., 2020] with its most prominent example given by ProtoNet [Snell et al., 2017], which we presented in the previous chapter.

Our main contributions in this chapter are then two-fold:

1. We empirically show that tracking the validity of assumptions on optimal predictors used in [Tripuraneni et al., 2020, Du et al., 2020] reveals a striking difference between the behavior of gradient-based and metric-based methods in how they

[1]We omit the results on meta-learning in the context of online convex optimization [Finn et al., 2019, Balcan et al., 2019, Khodak et al., 2019, Denevi et al., 2019] as they concern a different learning setup.

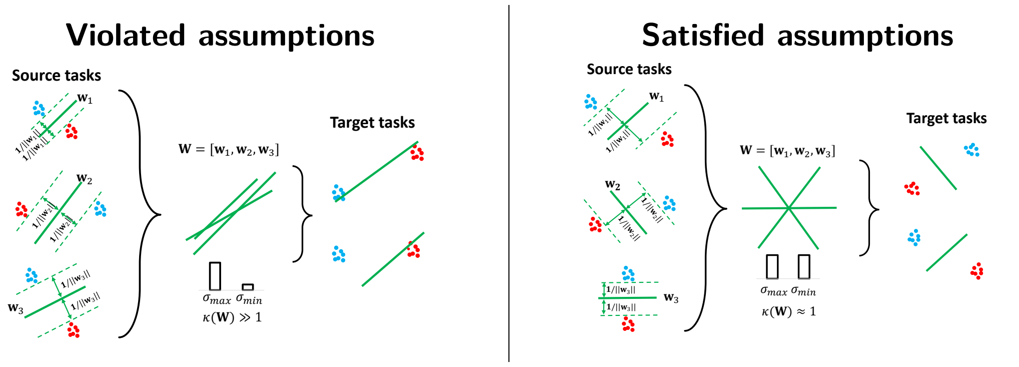


Figure 4.2. *Illustration of the intuition behind the assumptions derived from the MTR learning theory.* ***(left)*** *Lack of diversity and increasing norm of the linear predictors restrict them from being useful on the target task.* ***(right)*** *When the assumptions are satisfied, the linear predictors cover the embedding space evenly and their norm remains roughly constant on source tasks making them useful for a previously unseen task.*

learn their optimal feature representations. We provide elements of theoretical analysis that explain this behavior and explain the implications of it in practice. Our work is thus complementary to Wang et al. [Wang et al., 2021b] and connects MTR, FSC and Meta-Learning from both theoretical and empirical points of view.

2. We show that theoretical assumptions mentioned above can be forced during the training of meta-learning algorithms for both families of considered methods, and that enforcing them leads to better generalization of the considered algorithms for FSC baselines. We further show that our analysis can be extended to a Multi-Task Learning algorithm (MTL [Wang et al., 2021b]), and that enforcing the same theoretical assumptions leads to improved results as well.

Before presenting our contributions, we give in the next section some necessary knowledge to introduce the concepts that will be presented in this chapter.

4.2 Preliminary Knowledge

We start by recalling the framework of Multi-Task Representation Learning (MTR). Then we provide review of the existing learning bounds and assumptions allowing one to obtain guarantees in this setting. Finally, we describe the meta-learning frameworks considered in the context of this contribution.

4.2.1 Multi-Task Representation Learning Setup

Given a set of T source tasks observed through finite size samples of size n_1 grouped into matrices $\mathbf{X}_t = (\mathbf{x}_{t,1}, \dots, \mathbf{x}_{t,n_1}) \in \mathbb{R}^{n_1 \times d}$ and vectors of outputs $\mathbf{y}_t = (y_{t,1}, \dots, y_{t,n_1}) \in \mathbb{R}^{n_1}$, $\forall t \in [[T]] := \{1, \dots, T\}$ generated by their respective distributions μ_t, the goal of Multi-Task Representation Learning (MTR) is to learn a shared representation ϕ belonging to a certain class of functions $\Phi := \{\phi \mid \phi : \mathbb{X} \to \mathbb{V},\ \mathbb{X} \subseteq \mathbb{R}^d,\ \mathbb{V} \subseteq \mathbb{R}^k\}$, generally represented as (deep) neural networks, and linear predictors $\mathbf{w}_t \in \mathbb{R}^k$, $\forall t \in [[T]]$ grouped in a matrix $\mathbf{W} \in \mathbb{R}^{T \times k}$. More formally, this is done by solving the following joint optimization problem:

$$\hat{\phi}, \widehat{\mathbf{W}} = \operatorname*{arg\,min}_{\phi \in \Phi, \mathbf{W} \in \mathbb{R}^{T \times k}} \frac{1}{Tn_1} \sum_{t=1}^{T} \sum_{i=1}^{n_1} \ell(y_{t,i}, \langle \mathbf{w}_t, \phi(\mathbf{x}_{t,i}) \rangle), \tag{4.1}$$

where $\ell : \mathbb{Y} \times \mathbb{Y} \to \mathbb{R}_+$, with $\mathbb{Y} \subseteq \mathbb{R}$, is a loss function. Once such a representation $\hat{\phi}$ is learned, we want to apply it to a new previously unseen target task observed through a pair $(\mathbf{X}_{T+1} \in \mathbb{R}^{n_2 \times d}, \mathbf{y}_{T+1} \in \mathbb{R}^{n_2})$ containing n_2 samples generated by the distribution μ_{T+1}. We expect that a linear classifier $\mathbf{w}$ learned on top of the obtained representation leads to a low true risk over the whole distribution μ_{T+1}. For this, we first use $\hat{\phi}$ to solve the following problem:

$$\hat{\mathbf{w}}_{T+1} = \operatorname*{arg\,min}_{\mathbf{w} \in \mathbb{R}^k} \frac{1}{n_2} \sum_{i=1}^{n_2} \ell(y_{T+1,i}, \langle \mathbf{w}, \hat{\phi}(\mathbf{x}_{T+1,i}) \rangle). \tag{4.2}$$

Then, we define the true target risk of the learned linear classifier $\hat{\mathbf{w}}_{T+1}$ as: $\mathcal{L}(\hat{\phi}, \hat{\mathbf{w}}_{T+1}) = \mathbb{E}_{(\mathbf{x},y) \sim \mu_{T+1}}[\ell(y, \langle \hat{\mathbf{w}}_{T+1}, \hat{\phi}(\mathbf{x}) \rangle)]$ and want it to be as close as possible to the ideal true risk $\mathcal{L}(\phi^*, \mathbf{w}^*_{T+1})$ where $\mathbf{w}^*_{T+1}$ and ϕ^* satisfy:

$$\forall t \in [[T+1]] \text{ and } (\mathbf{x}, y) \sim \mu_t, \quad y = \langle \mathbf{w}^*_t, \phi^*(\mathbf{x}) \rangle + \varepsilon, \quad \varepsilon \sim \mathcal{N}(0, \sigma^2). \tag{4.3}$$

Equivalently, most of the works found in the literature seek to upper-bound the *excess risk* defined as $\mathrm{ER}(\hat{\phi}, \hat{\mathbf{w}}_{T+1}) := \mathcal{L}(\hat{\phi}, \hat{\mathbf{w}}_{T+1}) - \mathcal{L}(\phi^*, \mathbf{w}^*_{T+1})$.

4.2.2 Learning Bounds and Assumptions

First studies in the context of MTR relied on the probabilistic assumption [Baxter, 2000, Pentina and Lampert, 2014, Maurer et al., 2016, Amit and Meir, 2018, Yin et al., 2020] stating that meta-train and meta-test tasks distributions are all sampled i.i.d. from the same random distribution. This assumption, however, is considered unrealistic as in many learning settings, such as FSC, source and target tasks' data are often given by different draws (without replacement) from the same dataset. In this setup, the above-mentioned works obtained the bounds having the following form:

$$\text{ER}(\hat{\phi}, \hat{\mathbf{w}}_{T+1}) \leq O\left(\frac{1}{\sqrt{n_1}} + \frac{1}{\sqrt{T}}\right).$$

Such a guarantee implies that even with the increasing number of source data, one would still have to increase the number of tasks as well, in order to draw the second term to 0. A natural improvement to this bound was then proposed by [Du et al., 2020] and [Tripuraneni et al., 2020] that obtained the bounds on the excess risk behaving as

$$\text{ER}(\hat{\phi}, \hat{\mathbf{w}}_{T+1}) \leq O\left(\frac{C(\Phi)}{n_1 T} + \frac{k}{n_2}\right), \tag{4.4}$$

where $C(\Phi)$ is a measure of the complexity of Φ. Both these results show that all the source and target samples are useful in minimizing the excess risk. Thus, in the FSC regime where target data is scarce, all source data helps to learn well. From a set of assumptions made by the authors in both of these works, we note the following two:

Assumption 1: Diversity of the source tasks The matrix of optimal predictors $\mathbf{W}^*$ should cover all the directions in $\mathbb{R}^k$ evenly. More formally, this can be stated as

$$\kappa(\mathbf{W}^*) := \frac{\sigma_1(\mathbf{W}^*)}{\sigma_k(\mathbf{W}^*)} = O(1), \tag{4.5}$$

where $\sigma_i(\cdot)$ denotes the i^{th} singular value of $\mathbf{W}^*$. As pointed out by the authors, such an assumption can be seen as a measure of diversity between the source tasks that are expected to be complementary to each other to provide a useful representation for a previously unseen target task. In the following, we will refer to $\kappa(\mathbf{W})$ as the *condition number* for matrix $\mathbf{W}$.

Assumption 2: Consistency of the classification margin The norm of the optimal predictors $\mathbf{w}^*$ should not increase with the number of tasks seen during meta-training[2]. This assumption says that the classification margin of linear predictors should remain constant thus avoiding over- or under-specialization to the seen tasks.

While being highly insightful, the authors did not provide any experimental evidence suggesting that verifying these assumptions in practice helps to learn more efficiently in the considered learning setting. An intuition behind these assumptions can be seen in Figure 4.2. When the assumptions do not hold, the linear predictors can be biased towards a single part of the space and over-specialized to the tasks. The representation learned will not generalize well to unseen tasks. If the assumptions are respected, the linear predictors are complementary and will not under- or over-specialize to the tasks seen. The representation learned can more easily adapt to the target tasks and achieve better generalization.

[2]While not stated separately, this assumption is used in [Du et al., 2020] to derive the final result on p.5 after the discussion of Assumption 4.3.

4.2.3 Meta-Learning Algorithms

We precise now the meta-learning frameworks that are considered in this chapter. Meta-learning algorithms considered below learn an optimal representation sequentially via the so-called episodic training strategy introduced by [Vinyals et al., 2016], instead of jointly minimizing the training error on a set of source tasks as done in MTR. As introduced in Chapter 3, episodic training mimics the training process at the task scale with each task data being decomposed into a training set *(support set $\mathcal{S}$)* and a testing set *(query set $\mathcal{Q}$)*. Recently, [Chen et al., 2020a] showed that the episodic training setup used in meta-learning leads to a generalization bounds of $O(\frac{1}{\sqrt{T}})$. This bound is independent of the task sample size n_1, which could explain the success of this training strategy for FSC in the asymptotic limit. However, unlike the results obtained by [Du et al., 2020] studied in this chapter, the lack of dependence on n_1 makes such a result uninsightful in practice as we are in a finite-sample size setting. This bound does not give information on other parameters to leverage when the task number cannot increase. We now recall two major families of meta-learning approaches below.

Metric-based methods These methods learn an embedding space in which feature vectors can be compared using a similarity function (usually a L_2 distance or cosine similarity) [Koch et al., 2015, Vinyals et al., 2016, Snell et al., 2017, Sung et al., 2018, Allen et al., 2019, Li et al., 2019, Simon et al., 2020]. They typically use a form of contrastive loss as their objective function, similarly to Neighborhood Component Analysis (NCA) [Goldberger et al., 2005] or Triplet Loss [Hoffer and Ailon, 2015]. In this chapter, we focus our analysis on the popular Prototypical Networks [Snell et al., 2017] (ProtoNet) that computes prototypes as the mean vector of support points belonging to the same class: $\mathbf{c}_i = \frac{1}{|\mathcal{S}_i|}\sum_{\mathbf{s}\in\mathcal{S}_i}\phi(\mathbf{s})$, with $\mathcal{S}_i$ the subset of support points belonging to class i.

ProtoNet minimizes the negative log-probability of the true class i computed as the softmax over distances to prototypes $\mathbf{c}_i$:

$$\mathcal{L}_{proto}(\mathcal{S},\mathcal{Q},\phi) := \mathbb{E}_{\mathbf{q}\sim\mathcal{Q}}\left[-\log\frac{\exp(-d(\phi(\mathbf{q}),\mathbf{c}_i))}{\sum_j \exp\left(-d(\phi(\mathbf{q}),\mathbf{c}_j)\right)}\right] \tag{4.6}$$

with d being a distance function used to measure similarity between points in the embedding space.

Gradient-based methods These methods learn through end-to-end or two-step optimization [Ravi and Larochelle, 2017, Nichol et al., 2018, Li et al., 2017, Oreshkin et al., 2018, Lee et al., 2019, Bertinetto et al., 2019, Park and Oliva, 2019, Chen et al., 2020e, Raghu et al., 2020] where given a new task, the goal is to learn a model from the task's training data specifically adapted for this task. MAML [Finn et al., 2017] updates its parameters θ using an end-to-end optimization process to find the best initialization such that a new task can be learned quickly, *i.e.* with few examples.

More formally, given the loss ℓ_t for each task $t \in [[T]]$, MAML minimizes the expected task loss after an *inner loop* or *adaptation* phase, computed by a few steps of gradient descent initialized at the model's current parameters:

$$\mathcal{L}_{MAML}(\theta) := \mathbb{E}_{t\sim\eta}[\ell_t(\theta - \alpha\nabla\ell_t(\theta))], \tag{4.7}$$

with η the distribution of the meta-training tasks and α the learning rate for the adaptation phase. For simplicity, we take a single step of gradient update in this equation.

In what follows, we establish our theoretical analysis for the popular methods ProtoNet and MAML. We add their improved variations respectively called Infinite Mixture Prototypes [Allen et al., 2019] (IMP) and Meta-Curvature [Park and Oliva, 2019] (MC) in the experiments to validate our findings. The interested reader can find an introduction to these two more advanced algorithms in Appendix A.2.

4.3 Understanding Meta-learning Algorithms through MTR Theory

In this section, we study the behavior of gradient- and metric-based meta-learning algorithms with respect to the theoretical insights from MTR theory.

4.3.1 Link between MTR and Meta-learning

Recently, [Wang et al., 2021b] has shown that meta-learning algorithms that only optimize the last layer in the inner-loop, solve the same underlying optimization procedure as multi-task learning. In particular, their contributions have the following implications:

1. For *Metric-based algorithms*, the majority of methods can be seen as MTR problems. This is true, in particular, for ProtoNet and IMP algorithms considered in this work.

2. In the case of *Gradient-based algorithms*, such methods as ANIL [Raghu et al., 2020] and MetaOptNet [Lee et al., 2019] that do not update the embeddings during the inner-loop, can be also seen as multi-task learning. However, MAML and MC in principal do update the embeddings even though there exists strong evidence suggesting that the changes in the weights during their inner-loop are mainly affecting the last layer [Raghu et al., 2020]. Consequently, we follow [Wang et al., 2021b] and use this assumption to analyze MAML and MC in MTR framework as well.

3. In practice, [Wang et al., 2021b] showed that the mismatch between the multi-task and the actual episodic training setup leads to a negligible difference.

In the following section, we start by empirically verifying that the behavior of meta-learning methods reveals very distinct features when looked at through the prism of the considered MTR theoretical assumptions. We then set on a quest of explaining the differences in their behavior, leading to novel insights into meta-learning algorithms and interesting open problems for future research.

4.3.2 What happens in practice?

To verify whether theoretical results from MTR setting are also insightful for episodic training used by popular meta-learning algorithms, we first investigate the natural behavior of MAML and ProtoNet when solving FSC tasks on the popular *miniImageNet* [Ravi and Larochelle, 2017], *tieredImageNet* [Ren et al., 2018] and *Omniglot* [Lake et al., 2015] datasets. The full experimental setup is detailed in Section 4.4.1.

To verify Assumption 1 from MTR theory, we want to compute singular values of $\mathbf{W}$ during the meta-training stage and to follow their evolution. In practice, as T is typically quite large, we propose a more computationally efficient solution that is to calculate the condition number only for the last batch of N predictors (with $N \ll T$) grouped in the matrix $\mathbf{W}_N \in \mathbb{R}^{N \times k}$ that capture the latest dynamics in the learning process. We further note that $\sigma_i(\mathbf{W}_N \mathbf{W}_N^\top) = \sigma_i^2(\mathbf{W}_N), \ \forall i \in [[N]]$ implying that we can calculate the SVD of $\mathbf{W}_N \mathbf{W}_N^\top$ (or $\mathbf{W}_N^\top \mathbf{W}_N$ for $k \leq N$) and retrieve the singular values from it afterwards. We now want to verify whether $\mathbf{w}_t$ cover all directions in the embedding space and track the evolution of the ratio of singular values $\kappa(\mathbf{W}_N)$ during training.

For the first assumption to be satisfied, we expect $\kappa(\mathbf{W}_N)$ to decrease gradually during the training thus improving the generalization capacity of the learned predictors and preparing them for the target task. To verify the second assumption, the norm of the linear predictors should not increase with the number of tasks seen during training, *i.e.*, $\|\mathbf{w}\|_2 = O(1)$ or, equivalently, $\|\mathbf{W}\|_F^2 = O(T)$ and $\|\mathbf{W}_N\|_F = O(1)$.

For gradient-based methods, linear predictors are directly the weights of the last layer of the model. Indeed, for each task, the model learns a batch of linear predictors and we can directly take the weights for $\mathbf{W}_N$. Meanwhile, metric-based methods do not use linear predictors but compute a similarity between features. In the case of ProtoNet, the similarity is computed with respect to class prototypes that are mean features of the instances of each class. For the Euclidean distance, *this is equivalent to a linear model* with the prototype of a class acting as the linear predictor of this class [Snell et al., 2017]. This means that we can apply our analysis directly to the

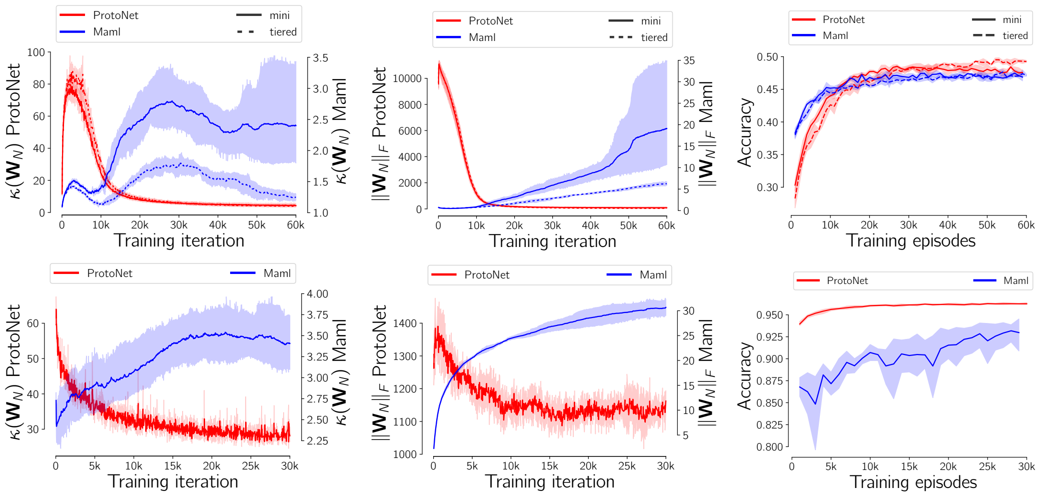


Figure 4.3. *Evolution of* $\kappa(\mathbf{W}_N)$ *(left),* $\|\mathbf{W}_N\|_F$ *(middle) and accuracy (right) during the training of ProtoNet (*red, left axes*) and MAML (*blue, right axes*) on miniImageNet (*mini, solid lines*) and tieredImageNet (*tiered, dashed lines*) with 5-way 1-shot episodes (*top*), and on the Omniglot dataset with 20-way 1-shot episodes (*bottom*).*

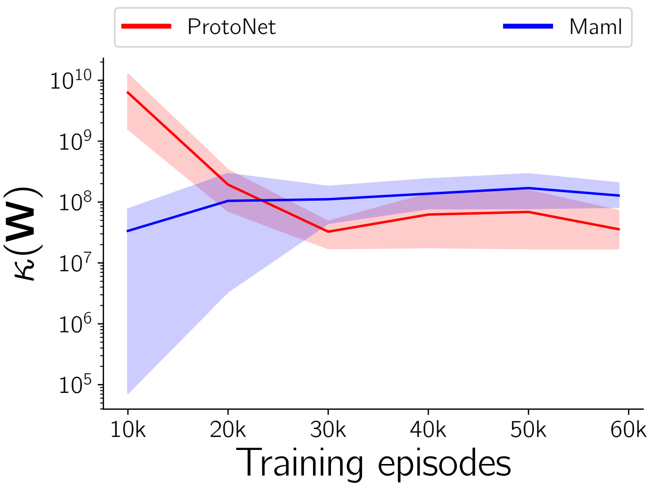


Figure 4.4. *Evolution of* $\kappa(\mathbf{W})$ *(in log scale) during the training of ProtoNet (in red) and MAML (in blue) on miniImageNet 5-way 1-shot episodes.*

prototypes computed by ProtoNet. In this case, the matrix $\mathbf{W}^*$ will be the matrix of the optimal prototypes and we can then take the *prototypes* computed for each task as our matrix $\mathbf{W}_N$.

From Figure 4.3, we can see that for MAML (*blue*), both $\|\mathbf{W}_N\|_F$ (*left*) and $\kappa(\mathbf{W}_N)$ (*middle*) increase with the number of tasks seen during training, whereas ProtoNet (*red*) naturally learns prototypes with a good coverage of the embedding space, and minimizes their norm. Since we compute the singular values of the last N predictors in $\kappa(\mathbf{W}_N)$, we can only compare the *overall behavior* throughout training between methods. For the sake of completeness, we also compute $\kappa(\mathbf{W})$ at different stages in the training, provided in Figure 4.4. To do so, we fix the encoder ϕ_T learned after T episodes and recalculate the linear predictors of the T past training episodes with this fixed encoder. We can see that $\kappa(\mathbf{W})$ of ProtoNet also decreases during training and reach a lower final value than $\kappa(\mathbf{W})$ of MAML. This confirms that the dynamics of $\kappa(\mathbf{W}_N)$ and $\kappa(\mathbf{W})$ are similar whereas the values $\kappa(\mathbf{W}_N)$ between methods should not be directly compared. The behavior of $\kappa(\mathbf{W})$ also validate our finding that ProtoNet learns to cover the embedding space with prototypes. This behavior is rather peculiar as neither of the two methods explicitly controls the theoretical quantities of interest, and still, ProtoNet manages to do it implicitly.

Before confirming this claim through extensive empirical evaluations involving more baseline methods and benchmark datasets, we first prove several results that provide an explanation to the difference of behavior of these two families of methods.

4.3.3 The case of Meta-Learning algorithms

The differences observed above for the two methods call for a deeper analysis of their behavior. To this end, we provide an explanation of why ProtoNet naturally leads to small condition number of the obtained predictors and a consistent behavior of their norm, while for MAML we consider a common simplified learning model leaving the general result as an open problem for future works.

ProtoNet We start by first explaining why ProtoNet learns prototypes that cover the embedding space efficiently. This result is given by the following theorem (cf. Appendix A.3 for the full proof).

Theorem 4.3.1. *(Normalized ProtoNet)*
If $\forall i \ \|\mathbf{c}_i\| = 1$, then $\forall \hat{\phi} \in \arg\min_\phi \mathcal{L}_{proto}(S, Q, \phi)$, the matrix of the optimal prototypes $\mathbf{W}^*$ is well-conditioned, *i.e.* $\kappa(\mathbf{W}^*) = O(1)$.

Proof sketch. We prove this result by recalling that, similarly to other metric-based methods, $\mathcal{L}_{proto}$ is a form of contrastive loss studied in [Wang and Isola, 2020]. By rewriting $\mathcal{L}_{proto}$ and using a part of their Theorem 1 proof on the convergence of the loss, we get the desired result. ■

This theorem explains the empirical behavior of ProtoNet in FSC task: the minimization of its objective function naturally minimizes the condition number when the norm of the prototypes is low. In particular, it implies that norm minimization seems to initiate the minimization of the condition number seen afterwards due to the contrastive nature of the loss function minimized by ProtoNet. We confirm this latter implication through experiments in Section 4.4 showing that norm minimization is enough for considered metric-based methods to obtain the well-behaved condition number and that minimizing both seems redundant.

MAML Unfortunately, the analysis of MAML in the most general case is notoriously harder, as even expressing its loss function and gradients in the case of an overparametrized linear regression model with only 2 parameters requires using a symbolic toolbox for derivations [Arnold et al., 2021].

To this end, we resort to the linear regression model considered in this latter paper and defined as follows. We assume for all $t \in [[T]]$ that the task parameters $\boldsymbol{\theta}_t$ are normally distributed with $\boldsymbol{\theta}_t \sim \mathcal{N}(\mathbf{0}_d, \boldsymbol{I}_d)$, the inputs $\mathbf{x}_t \sim \mathcal{N}(\mathbf{0}_d, \boldsymbol{I}_d)$ and the output $y_t \sim \mathcal{N}(\langle \boldsymbol{\theta}_t, \mathbf{x}_t \rangle, 1)$. For each t, we consider the following learning model and its associated square loss:

$$\hat{y}_t = \langle \mathbf{w}_t, \mathbf{x}_t \rangle, \quad \ell_t = \mathbb{E}_{p(\mathbf{x}_t, y_t | \boldsymbol{\theta}_t)} (y_t - \langle \mathbf{w}_t, \mathbf{x}_t \rangle)^2. \tag{4.8}$$

We can now state the following result (the full proof can be found in Appendix A.3).

Proposition 4.3.1. Let $\forall t \in [[T]]$, $\boldsymbol{\theta}_t \sim \mathcal{N}(\mathbf{0}_d, \boldsymbol{I}_d)$, $\mathbf{x}_t \sim \mathcal{N}(\mathbf{0}_d, \boldsymbol{I}_d)$ and $y_t \sim \mathcal{N}(\langle \boldsymbol{\theta}_t, \mathbf{x}_t \rangle, 1)$. Consider the learning model from Equation (4.8), let $\boldsymbol{\Theta}_i := [\boldsymbol{\theta}_i, \boldsymbol{\theta}_{i+1}]^T$, and denote by $\widehat{\mathbf{W}}_2^i$ the matrix of last two predictors learned by MAML at iteration i starting from $\widehat{\mathbf{w}}_0 = \mathbf{0}_d$. Then, we have that:

$$\forall i, \quad \kappa(\widehat{\mathbf{W}}_2^{i+1}) \geq \kappa(\widehat{\mathbf{W}}_2^i), \quad \text{if } \sigma_{\text{min}}(\boldsymbol{\Theta}_i) = 0. \tag{4.9}$$

Proof sketch. We use the analytical expression for the t^{th} task gradient given in [Arnold et al., 2021] for this case. We then show that when starting the optimization process from $\widehat{\mathbf{w}}_0 = \mathbf{0}_d$, the sequence of the obtained linear predictors are interdependent such that $\widehat{\mathbf{W}}_2^{i+1} \propto \widehat{\mathbf{W}}_2^i + \boldsymbol{\Theta}_i$. We then use upper and lower bounds for singular values of a sum of matrices to derive the desired result. ■

This proposition provides an explanation of why MAML may tend to increase the ratio of singular values during the iterations. Indeed, the condition when this happens indicates that the optimal predictors forming matrix $\boldsymbol{\Theta}_i$ are linearly dependent implying that its smallest singular values becomes equal to 0. While this is not expected to be the case for all iterations, we note, however, that in FSC task the draws from the dataset are in general not i.i.d. and thus may correspond to co-linear optimal predictors. In every such case, the condition number is expected to remain non-decreasing, as illustrated in Figure 4.3 (*left*) where for MAML, contrary to ProtoNet, $\kappa(\mathbf{W}_N)$ exhibits plateaus but also intervals where it is increasing.

This highlights a major difference between the two approaches: MAML does not specifically seek to diversify the learned predictors, while ProtoNet does. Even though the condition number and the classifier norm both increase for MAML, it still achieves good performance. **This means that both assumptions are not necessary to meta-learning. However, verifying them leads to a more favorable framework that allows for better performance as confirmed by our experimental evaluation below.** Proposition 4.3.1 gives a hint to the essential difference between the methods studied. On the one hand, ProtoNet constructs its classifiers directly from the data of the current task and they are independent of the other tasks. On the other hand, for MAML, the weights of the classifiers are reused between tasks and only slightly adapted to be specific to each task. This limits the generalization capabilities of the linear predictors learned by MAML since they are based on predictors from previous tasks.

In the following section, we advance this insight further by showing that for some data generating distributions one may learn two data representations that achieve a perfect fidelity to the data generating process but differ drastically in terms of the coverage of the embedding space provided by their linear predictors.

4.3.4 Enforcing the assumptions

So far, we have gathered evidence for the fact that MTR theory seems to be insightful for meta-learning algorithms as well. As satisfying the assumptions from MTR theory is expected to come in hand with better generalization performance, we now study what impact forcing these assumptions may have on the learning process when the optimal predictors involved in the data generating process do not naturally satisfy them.

Why should we force the assumptions ? From the results obtain by [Du et al., 2020], and with the same assumptions, we can easily make appear $\kappa(\mathbf{W}^*)$ to obtain a more explicit bound:

Proposition 4.3.2. If $\forall t \in [[T]], \|\mathbf{w}_t^*\| = O(1)$ and $\kappa(\mathbf{W}^*) = O(1)$, and $\mathbf{w}_{T+1}$ follows a distribution ν such that $\|\mathbb{E}_{\mathbf{w}\sim\nu}[\mathbf{w}\mathbf{w}^\top]\| \leq O\left(\frac{1}{k}\right)$, then

$$\mathrm{ER}(\hat{\phi}, \hat{\mathbf{w}}_{T+1}) \leq O\left(\frac{C(\Phi)}{n_1 T} \cdot \kappa(\mathbf{W}^*) + \frac{k}{n_2}\right). \tag{4.10}$$

Proposition 4.3.2 suggests that the terms $\|\mathbf{w}_t^*\|$ and $\kappa(\mathbf{W}^*)$ underlying the assumptions directly impact the tightness of the established bound on the excess risk. The full proof can be found in Appendix A.3.

Can we force the assumptions ? According to the previous result, satisfying the assumptions from MTR theory is expected to come in hand with better performance. However all the terms involved refer to optimal predictors, that we cannot act upon. Thus, we aim to answer the following question:

Given $\mathbf{W}^*$ such that $\kappa(\mathbf{W}^*) \gg 1$, can we learn $\widehat{\mathbf{W}}$ with $\kappa(\widehat{\mathbf{W}}) \approx 1$ while solving the underlying classification problems equally well?

While obtaining such a result for any distribution seems to be very hard in the considered learning setup, we provide a constructive proof for the existence of a distribution for which the answer to the above-mentioned question is positive in the case of two tasks. The latter restriction comes out of the necessity to analytically calculate the singular values of $\mathbf{W}$ but we expect our example to generalize to more general setups and a larger number of tasks as well.

Proposition 4.3.3. Let $T = 2$, $\mathbb{X} \subseteq \mathbb{R}^d$ be the input space and $\mathbb{Y} = \{-1, 1\}$ be the output space. Then, there exist distributions μ_1 and μ_2 over $\mathbb{X} \times \mathbb{Y}$, representations $\widehat{\phi} \neq \phi^*$ and matrices of predictors $\widehat{\mathbf{W}} \neq \mathbf{W}^*$ that satisfy the data generating model (Equation (4.3)) with $\kappa(\widehat{\mathbf{W}}) \approx 1$ and $\kappa(\mathbf{W}^*) \gg 1$.

Proof sketch. The construction used in this proof is illustrated in Figure 4.5. We define two uniform distributions μ_1 and μ_2 parametrized by a scalar $\varepsilon > 0$ and find a tuple $(\mathbf{W}^*, \phi^*)$ that satisfies the data generating process from Equation (4.3). To this end, ϕ^* is picked as a linear operator that projects the data generated by μ_i to a two-dimensional space by discarding its last $d-2$ dimensions and ensuring that $\kappa(\mathbf{W}^*) = \frac{1}{\varepsilon}$. The dependence of the ratio on ε allows us to make it arbitrary large. Then, we construct another operator $\widehat{\phi}$ that projects the data generated by μ_i to a two-dimensional space by discarding the first and last $d-1$ dimensions. We then show that in this space the matrix of optimal predictors $\widehat{\mathbf{W}}$ has a ratio $\kappa(\widehat{\mathbf{W}}) \xrightarrow{\varepsilon \to 0} 1$. ■

The established results show that in some cases even when $\mathbf{W}^*$ does not satisfy Assumptions 1-2 in the ϕ^* space, it may still be possible to learn a new representation $\widehat{\phi}$ such that the optimal predictors in this space do satisfy them. The full proof of this result can be found in Appendix A.3.

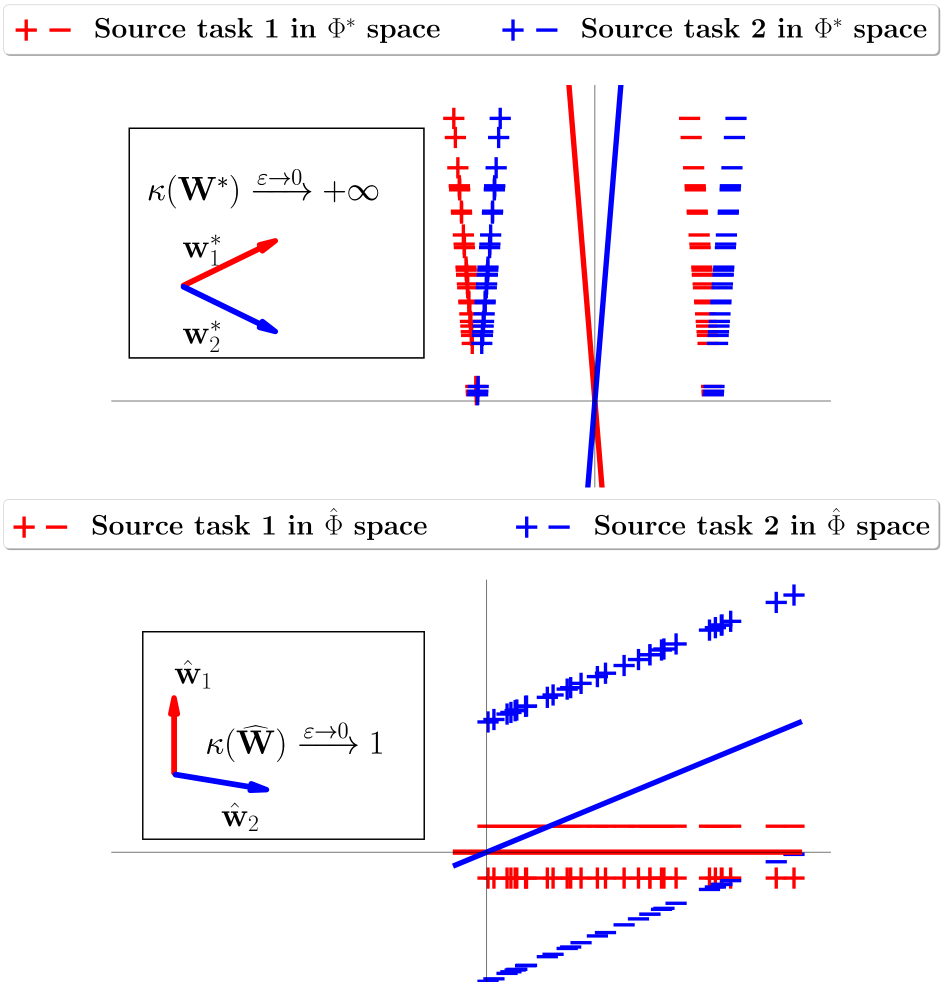


Figure 4.5. *Visualization of the distributions used in the constructive example for the proof of Proposition 4.3.3, with $\epsilon = 0.02$. In this example, $\kappa(\widehat{\mathbf{W}})$ is closer to 1 than $\kappa(\mathbf{W}^*)$. It shows that we can search for a representation $\widehat{\phi}$ such that optimal predictors in this space are fulfilling the assumptions, while solving the underlying problem equally well.*

How to force the assumptions ? This can be done either by considering the constrained problem:

$$\widehat{\phi}, \widehat{\mathbf{W}} = \underset{\phi \in \Phi, \mathbf{W} \in \mathbb{R}^{T \times k}}{\arg\min} \frac{1}{Tn_1} \sum_{t=1}^{T} \sum_{i=1}^{n_1} \ell(y_{t,i}, \langle \mathbf{w}_t, \phi(\mathbf{x}_{t,i}) \rangle)$$
$$\text{s.t. } \kappa(\mathbf{W}) = O(1), \|\mathbf{w}_t\| = 1, \tag{4.11}$$

or by using a more common strategy that consists in adding $\kappa(\mathbf{W})$ and $\|\mathbf{W}\|_F^2$ directly as regularization terms:

$$\widehat{\phi}, \widehat{\mathbf{W}} = \underset{\phi \in \Phi, \mathbf{W} \in \mathbb{R}^{T \times k}}{\arg\min} \frac{1}{Tn_1} \sum_{t=1}^{T} \sum_{i=1}^{n_1} \ell(y_{t,i}, \langle \mathbf{w}_t, \phi(\mathbf{x}_{t,i}) \rangle) + \lambda_1 \kappa(\mathbf{W}) + \lambda_2 \|\mathbf{W}\|_F^2. \quad (4.12)$$

By adding $\|\mathbf{W}\|_F^2$ in the loss, we force the model to have a low norm on the weights. Since it cannot be put to 0 or below, the model will keep the norm relatively constant instead of increasing it. The second regularizer term is a softer way to apply the constraint on the norm rather than normalizing the weights as in Equation (4.11).

According to Theorem 7.1 from [Lewis and Sendov, 2005], subgradients of singular values function are well-defined for absolutely symmetric functions. In our case, we are computing the squared singular values $\sigma^2(\mathbf{W})$ and we retrieve the singular values by taking the square root. This means that effectively, we are computing $\kappa(\mathbf{W}) = \max(|\sigma(\mathbf{W})|)/\min(|\sigma(\mathbf{W})|)$, which is an absolutely symmetric function. Consequently, subgradients of the spectral regularization term $\kappa(\mathbf{W})$ are well-defined and can be optimized efficiently when used in the objective function.

To the best of our knowledge, such regularization terms based on insights from the advances in MTR theory have never been used in the literature before. We also further use the basic quantities involved in the proposed regularization terms as indicators of whether a given meta-learning algorithm naturally satisfies the assumptions ensuring an efficient meta-learning in practice or not.

4.3.5 Positioning with respect to Previous Work

Below, we discuss several related studies aiming at improving the general understanding of meta-learning, and mention other regularization terms specifically designed for meta-learning.

Understanding meta-learning While a complete theory for meta-learning is still lacking, several recent works aim to shed light on phenomena commonly observed in meta-learning by evaluating different intuitive heuristics. For instance, [Raghu et al., 2020] investigate whether MAML works well due to rapid learning with significant changes in the representations when deployed on target task, or due to feature reuse where the learned representation remains almost intact. They establish that the latter factor is dominant and propose a new variation of MAML that freezes all but task-specific layers of the neural network when learning new tasks. In [Goldblum et al., 2020], the authors explain the success of meta-learning approaches by their capability to either cluster classes more tightly in feature space (task-specific adaptation approach), or to search for meta-parameters that lie close in weight space to many task-specific minima (full fine-tuning approach). Finally, the effect of the number of shots on the

FSC accuracy was studied in [Cao et al., 2020] for ProtoNet. More recently, [Ye and Chao, 2021] studied the impact of the permutation of labels when training MAML. They show that this key part of meta-training, severely impacts MAML as it does not enjoy permutation invariance. Our work investigates a new aspect of meta-learning that has never been studied before and, unlike [Raghu et al., 2020], [Cao et al., 2020], [Goldblum et al., 2020] and [Ye and Chao, 2021], provides a more complete experimental evaluation with the two different approaches of meta-learning.

Normalization Multiple methods in the literature introduce a normalization of their features either to measure cosine similarity instead of Euclidean distance [Gidaris and Komodakis, 2018, Qi et al., 2018, Chen et al., 2019] or because of the noticed improvement in their performance [Wang et al., 2019, Tian et al., 2020c, Wang et al., 2021b]. In this chapter, we proved in Section 4.3.3 above that for ProtoNet prototypes normalization is enough to achieve a good coverage of the embedding space, and we empirically show in Section 4.4 below that it leads to better performance. Since we only normalize the prototypes and not all the features, we do not measure cosine similarity. Moreover, with our Theorem 4.3.1, we give explanations through MTR theory regarding the link between the coverage of the representation space and performance.

Common regularization strategies In general, we note that regularization in meta-learning (i) is applied to either the weights of the whole neural network [Balaji et al., 2018, Yin et al., 2020], or (ii) the predictions [Jamal and Qi, 2019, Goldblum et al., 2020] or (iii) is introduced via a prior hypothesis biased regularized empirical risk minimization [Pentina and Lampert, 2014, Kuzborskij and Orabona, 2017, Denevi et al., 2018a, Denevi et al., 2018b, Denevi et al., 2019]. Contrary to the first group of methods and to the weight decay approach [Krogh and Hertz, 1992], we do not regularize the whole weight matrix learned by the neural network but the linear predictors of its last layer. While weight decay is used to avoid overfitting by penalizing large magnitudes of weights, our goal is to keep the classification margin unchanged during the training to avoid over-/under-specialization to source tasks. Similarly, spectral normalization proposed by [Miyato et al., 2018a] does not affect the condition number κ. Second, we regularize the singular values of the matrix of linear predictors obtained in the last batch of tasks instead of the predictions used by the methods of the second group (*e.g.*, using the theoretic-information quantities in [Jamal and Qi, 2019]). Finally, the works of the last group are related to the online setting with convex loss functions only and do not specifically target the spectral properties of the learned predictors.

4.4 Impact of enforcing theoretical assumptions

In this section, we investigate the impact of enforcing the aforementioned theoretical assumptions for meta-learning algorithms in practice.

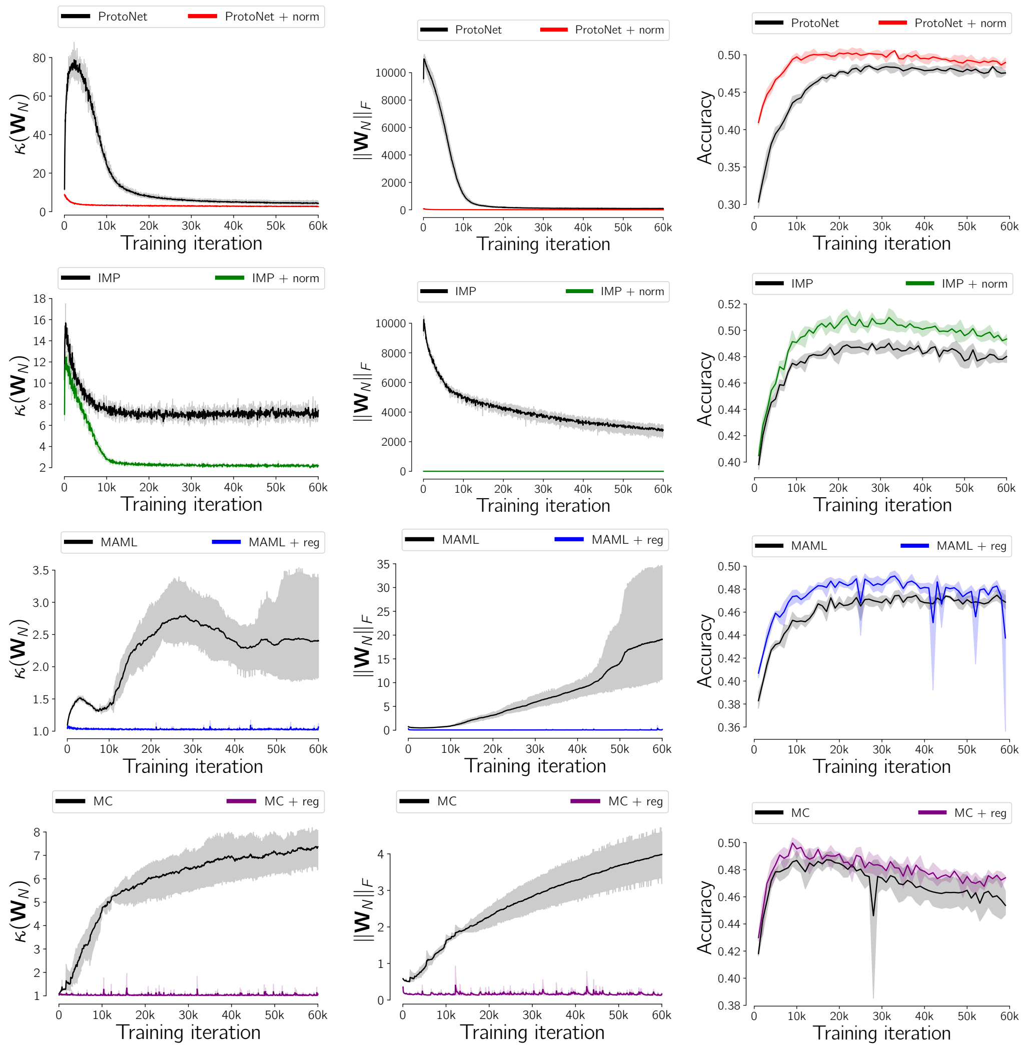


Figure 4.6. *Evolution of* $\kappa(\mathbf{W}_N)$ *(*left*),* $\|\mathbf{W}_N\|_F$ *(*middle*) and the accuracy (*right*) on 5-way 1-shot episodes from* miniImageNet*, for the metric-based methods ProtoNet (*first row*) and IMP (*second row*), as well as the gradient-based methods MAML (*third row*) and MC (*last row*) and their regularized or normalized counterparts.*

4.4.1 Experimental Setup

We consider the FSC problem on the three benchmarks introduced in Chapter 3 that we recall briefly here:

1. **Omniglot** [Lake et al., 2015] is a dataset of 20 instances of 1623 characters from 50 different alphabets. Each image was hand-drawn by different people. The images are resized to 28×28 pixels and the classes are augmented with

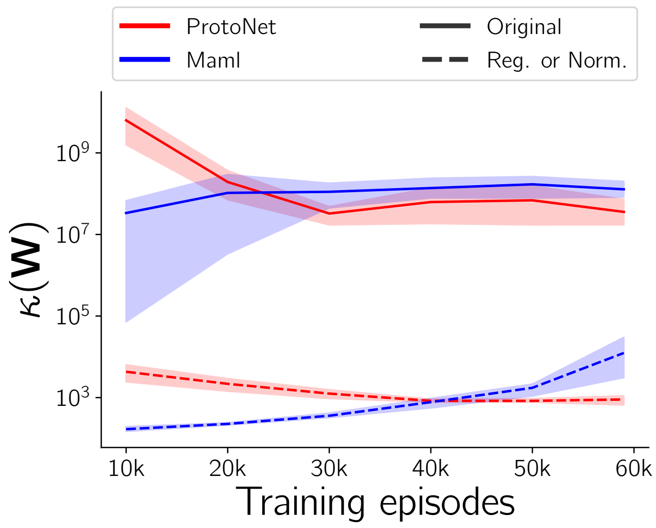


Figure 4.7. *Evolution of $\kappa(\mathbf{W})$ on 5-way 1-shot episodes from* miniImageNet*, for ProtoNet (*red, straight lines*), MAML (*blue, straight lines*) along with their regularized and normalized counterparts (*respectively the same colors, in dashed lines*).*

rotations by multiples of 90 degrees.

2. **miniImageNet** [Ravi and Larochelle, 2017] is a dataset made from randomly chosen classes and images taken from the ILSVRC-12 dataset [Russakovsky et al., 2015]. The dataset consists of 100 classes and 600 images for each class. The images are resized to 84×84 pixels and normalized.

3. **tieredImageNet** [Ren et al., 2018] is also a subset of ILSVRC-12 dataset. However, unlike miniImageNet, training classes are semantically unrelated to testing classes. The dataset consists of $779,165$ images divided into 608 classes. Here again, the images are resized to 84×84 pixels and normalized.

For each dataset, we follow a common experimental protocol used in [Finn et al., 2017, Chen et al., 2019]. More specifically, we use a four-layer convolution backbone with 64 filters (C64) as done by [Chen et al., 2019] optimized with Adam [Kingma and Ba, 2015] and a learning rate of 0.001. On *miniImageNet* and *tieredImageNet*, models are trained on 60000 5-way 1-shot or 5-shot episodes and on 30000 20-way 1-shot or 5-shot episodes for *Omniglot*. We use a batch size of 4 and evaluate on the validation set every 1000 episodes. We keep the best performing model on the validation set to evaluate on the test set. The seeds used for all experiments are 1, 10, 100 and 1000. For MAML and MC, we use an inner learning rate of 0.01 for *miniImageNet* and *tieredImageNet*, and 0.1 for *Omniglot*. During training, we perform 5 inner gradient

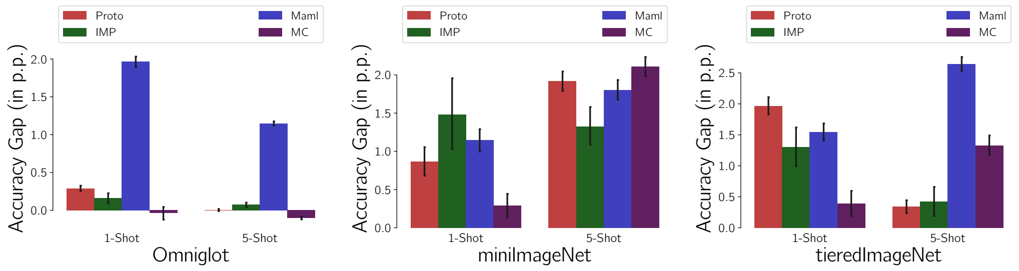


Figure 4.8. *Accuracy gap (in p.p.) when adding the normalization of prototypes for ProtoNet (*red*) and IMP (*green*), and both spectral and norm regularization for MAML (*blue*) and MC (*purple*) enforcing the theoretical assumptions on Omniglot (*left*), miniImagenet (*middle*) and tieredImagenet (*right*) datasets. Exact performance values are reported in Appendix A.4.*

step and 10 step during testing. For all FSC experiments, unless explicitly stated, we use the regularization parameters $\lambda_1 = \lambda_2 = 1$ in Equation (4.12). We measure the performance using the top-1 accuracy with 95% confidence intervals, reproduce the experiments with 4 different random seeds using a single NVIDIA V100 GPU, and average the results over 2400 test tasks.

We also provide experiments with the ResNet-12 architecture backbone [Lee et al., 2019]. In this case, we follow the recent practice and initialize the models with the weights pretrained on the entire meta-training set [Ye et al., 2020, Rusu et al., 2018, Qiao et al., 2018a]. Like in their protocol, this initialization is updated by meta-training with ProtoNet or MAML on at most 20000 episodes, grouping every 100 episodes into an *epoch*. Then, the best performing model on the validation set, evaluated at every epoch, is kept and the performance on 10000 test tasks is measured. For all experiments with the ResNet-12 architecture, the SGD optimizer with a weight decay of 0.0005 and momentum of 0.9 and a batch of episodes of size 1 are used. For ProtoNet, following the protocol of Ye et al. [Ye et al., 2020], an initial learning rate of 0.0002, decayed by a factor 0.5 every 40 epochs, is used. For MAML, following Ye et Chao [Ye and Chao, 2021], the initial learning rate is set to 0.001, decayed by a factor 0.1 every 20 epochs. The number of inner loop updates are respectively set to 15 and 20 with a step size of 0.05 and 0.1 for 1-shot and 5-shot episodes on the miniImageNet dataset, and respectively 20 and 15 with a step size of 0.001 and 0.05 on the tieredImageNet dataset.

4.4.2 Metric-based Methods

4.4.2.1 Using normalized class prototypes

Theorem 4.3.1 tells us that with normalized class prototypes that act as linear predictors, ProtoNet naturally decreases the condition number of their matrix. Furthermore, since the prototypes are directly the image features, adding a regularization term on the norm of the prototypes makes the model collapse to the trivial solution which maps all images to 0. To this end, we choose to ensure the theoretical assumptions for metric-based methods (ProtoNet and IMP) only with the prototype normalization:

$$\widehat{\phi}, \widehat{\mathbf{W}} = \underset{\phi \in \Phi, \mathbf{W} \in \mathbb{R}^{T \times k}}{\arg\min} \frac{1}{Tn_1} \sum_{t=1}^{T} \sum_{i=1}^{n_1} \ell(y_{t,i}, \langle \tilde{\mathbf{w}}_t, \phi(\mathbf{x}_{t,i}) \rangle), \tag{4.13}$$

where $\tilde{\mathbf{w}} = \frac{\mathbf{w}}{\|\mathbf{w}\|}$ are the normalized prototypes. According to Theorem 4.3.1, the normalization of the prototypes makes the problem similar to the constrained problem given in Equation (4.11).

As can be seen in Figures 4.6 and 4.7, the normalization of the prototypes has the intended effect on the condition number of the matrix of predictors. Indeed, $\kappa(\mathbf{W}_N)$ (*left*) stay constant and low during training, and we achieve a much lower $\kappa(\mathbf{W})$ (*right*) than without normalization.

From Figure 4.8, we note that normalizing the prototypes from the very beginning of the training process has an overall positive effect on the obtained performance, and this gain is statistically significant in most of the cases according to the *Wilcoxon signed-rank* test ($p < 0.05$) [Wilcoxon, 1945, David and Siegel, 1956]. In Table 4.1, we compare the performance obtained against state-of-the-art algorithms behaving similarly to Instance Embedding algorithms [Ye et al., 2020] such as ProtoNet, depending on the architecture used. Even with a ResNet-12 architecture, the proposed normalization still improves the performance to reach competitive results with the state-of-the-art. On the miniImageNet 5-way 5-shot benchmark, our normalized ProtoNet achieves 80.95%, better than DSN (78.83%), CTM (78.63%) and SimpleShot (80.02%).

4.4.2.2 Further enforcing a low condition number on Metric-based methods

To guide the model into learning an encoder with the lowest condition number, we consider adding $\kappa(\mathbf{W}_N)$ as a regularization term in Equation (4.13). In addition to the normalization of the prototypes, this should further enforce the assumption on the condition number. Unfortunately, this latter strategy hinders the convergence of the network and leads to numerical instabilities. It is most likely explained by prototypes being computed from image features which suffer from rapid changes across batches, making the smallest singular value $\sigma_N(\mathbf{W}_N)$ close to 0. Consequently, we propose to replace the condition number as a regularization term by the *negative entropy of the vector of singular values* as follows:

$$H_\sigma(\mathbf{W}_N) := \sum_{i=1}^{N} \mathrm{softmax}(\sigma(\mathbf{W}_N))_i \cdot \log \mathrm{softmax}(\sigma(\mathbf{W}_N))_i, \tag{4.14}$$

where $\mathrm{softmax}(\cdot)_i$ is the i^{th} output of the *softmax* function. Since uniform distribution has the highest entropy, regularizing with $\kappa(\mathbf{W}_N)$ or $H_\sigma(\mathbf{W}_N)$ leads to a better coverage of $\mathbb{R}^k$ by ensuring a nearly identical importance regardless of the direction.

We obtain the following regularized optimization problem:

$$\widehat{\phi}, \widehat{\mathbf{W}} = \operatorname*{arg\,min}_{\phi \in \Phi, \mathbf{W} \in \mathbb{R}^{T \times k}} \frac{1}{Tn_1} \sum_{t=1}^{T} \sum_{i=1}^{n_1} \ell(y_{t,i}, \langle \tilde{\mathbf{w}}_t, \phi(\mathbf{x}_{t,i}) \rangle) + \lambda_1 H_\sigma(\mathbf{W}), \tag{4.15}$$

where $\tilde{\mathbf{w}} = \frac{\mathbf{w}}{\|\mathbf{w}\|}$ are the normalized prototypes.

In Table 4.2, we report the performance of ProtoNet without normalization, with normalization and with both normalization and regularization on the entropy. Finally, we can see that further enforcing a regularization on the singular values through the entropy does not help the training since ProtoNet naturally learns to minimize the singular values of the prototypes. In Table 4.3, we show that reducing the *strength* of the regularization with the entropy can help to improve the performance.

4.4.3 Gradient-based Methods

Gradient-based methods learn a batch of linear predictors for each task, and we can directly take them as $\mathbf{W}_N$ to compute its SVD. In the following experiments, we consider the regularized problem of Equation (4.12) for MAML as well as Meta-Curvature (MC). As expected, the dynamics of $\|\mathbf{W}_N\|_F$ and $\kappa(\mathbf{W}_N)$ during the training of the regularized methods remain bounded and the effect of the regularization is confirmed with the lower value of $\kappa(\mathbf{W})$ achieved (cf. Figure 4.6).

The impact of our regularization on the results is quantified in Figure 4.8 where a statistically significant accuracy gain is achieved in most cases, according to the *Wilcoxon signed-rank* test ($p < 0.05$) [Wilcoxon, 1945, David and Siegel, 1956]. In Table 4.1, we compare the performance obtained to state-of-the-art gradient-based algorithms. We can see that our proposed regularization is globally improving the results, even with a bigger architecture such as ResNet-12 and with an additional pretraining. On the miniImageNet 5-way 5-shot benchmark, with our regularization

MAML achieves 82.45%, better than TADAM (76.70%), MetaOptNet (78.63%) and MATE with MetaOptNet (78.64%). In Table 4.4, we compare the performance of MAML without regularization ($\lambda_1 = \lambda_2 = 0$), with a regularization on the condition number $\kappa(\mathbf{W}_N)$ ($\lambda_1 = 1$ and $\lambda_2 = 0$), on the norm of the linear predictors ($\lambda_1 = 0$ and $\lambda_2 = 1$), and with both regularization terms ($\lambda_1 = \lambda_2 = 1$) on Omniglot and miniImageNet. We can see that both regularization terms are important in the training and that using only a single term can be detrimental to the performance.

4.4.4 Multi-Task learning Methods

We implement our regularization on a recent Multi-Task Learning (MTL) method [Wang et al., 2021b], following the same experimental protocol. The objective is to empirically validate our analysis on a method using the MTR framework. As mentioned in Section 4.3.5, the authors introduce feature normalization in their method, speculating that it improves coverage of the representation space [Wang and Isola, 2020]. Using their code, we reproduce their experiments on three different settings compared in Table 4.5: the vanilla MTL, the MTL with feature normalization, and MTL with our proposed regularization on the condition number and the norm of the linear predictors. We use $\lambda_1 = 1$ in all the settings, and $\lambda_2 = 1$ in the 1-shot setting and $\lambda_2 = 0.01$ in the 5-shot settings. Our regularization, as well as the normalization, globally improve the performance over the non-normalized models. Notably, our regularization is the most effective when there is the less data which is well-aligned with the MTR theory in few-shot setting. We can also note that in most of the cases, the normalized models and the regularized ones achieve similar results, hinting that they may have a similar effect.

Table 4.6 presents the effect of varying independently either parameter λ_1 or λ_2 in the regularization, the other being fixed to 1. From these results, we can see that performance is much more impacted by the condition number regularization (parameter λ_1) than by the normalization (parameter λ_2). Indeed, varying the regularization weight can lead from the lowest accuracy (74.64%, for $\lambda_1 = 0$) to one of the highest accuracies (76.15% for $\lambda_1 = 0.2$).

All of these results show that our analysis and our proposed regularization are also valid in the MTL framework.

4.4.5 Out-of-domain Analysis

In the theoretical MTR framework, one additional critical assumption made is that the task data marginal distributions are similar[3], which does not hold in a *cross-domain* setting, where we evaluate a model on a dataset different from the training

[3]We refer the reader to Appendix A.1 for more information on this assumption

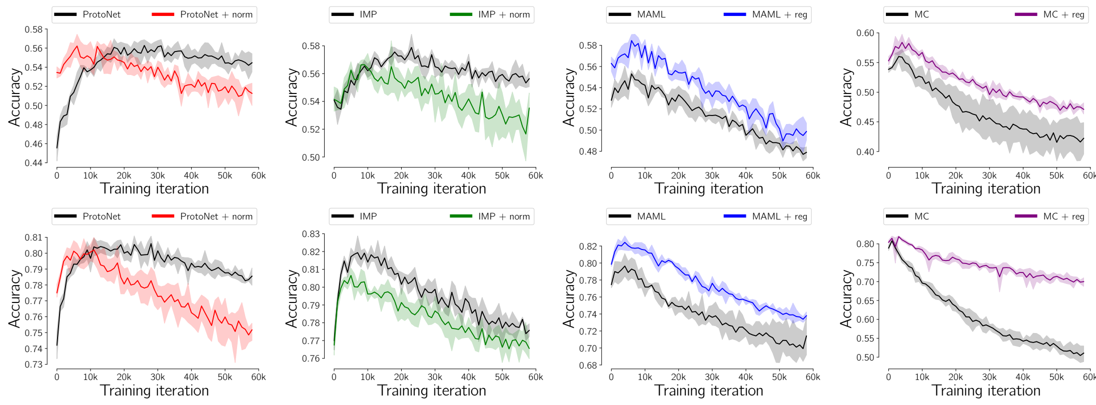


Figure 4.9. *Evolution of accuracy of 5-way 1-shot (*top*, resp. 5-way 5-shot,* bottom*) meta-test episodes from* CropDisease *during meta-training on 5-way 1-shot (*top*, resp. 5-way 5-shot,* bottom*) episodes from* miniImageNet*, for ProtoNet, IMP, MAML, MC (*from left to right*) and their regularized or normalized counterparts (*in red, green, blue and purple, respectively*). All results are averaged over 4 different random seeds. The shaded areas show 95% confidence intervals.*

dataset. In this setting, we do not have the same guarantees that our regularization or normalization schemes will be as effective as in *same-domain*. To verify this, we measure out-of-domain performance on the **CropDiseases** dataset [Mohanty et al., 2016] adopted by [Guo et al., 2020]. Following their protocol, this dataset is used only for testing purposes. In this specific experiment, evaluated models are trained on *miniImageNet*.

On the one hand, for metric-based methods, the improvement in the *same-domain* setting does not translate to the *cross-domain* setting. From Figure 4.9, we can see that even though the low condition number in the beginning of training leads to improved early generalization capabilities of ProtoNet, this is not the case for IMP. We attribute this discrepancy between ProtoNet and IMP to a difference in cluster radius parameters of IMP and normalized IMP, making the encoder less adapted to *out-of-domain* features. On the other hand, we found that gradient-based models keep their accuracy gains when evaluated in cross-domain setting with improved generalization capabilities due to our regularization. This can be seen on Figure 4.9, where we achieve an improvement of about 2 percentage points *(p.p.)* for both MAML and MC models on both 1-shot and 5-shot settings.

These results confirm that minimizing the norm and condition number of the linear predictors learned improves the generalization capabilities of meta-learning models. As opposed to metric-based methods which are already implicitly doing so, the addition of the regularization terms for gradient-based methods leads to a more significant improvement of performance in *cross-domain*.

4.5 Conclusion

In this chapter, we studied the validity of the theoretical assumptions made in recent papers of Multi-Task Representation Learning theory when applied to popular metric- and gradient-based meta-learning algorithms. We found a striking difference in their behavior and provided both theoretical and experimental arguments explaining that metric-based methods satisfy the considered assumptions, while gradient-based don't. We further used this as a starting point to implement a regularization strategy ensuring these assumptions and observed that it leads to faster learning and better generalization.

While this work proposes an initial approach to bridging the gap between theory and practice for Meta-Learning, some questions remain open on the inner workings of these algorithms. Considering settings different from the episodic one represents another important line of research for the understanding of few-shot learning [Tian et al., 2020c, Yang et al., 2021b]. In the remaining chapters, we are interested in the more challenging Object Detection task that requires to take into account the *location* of objects along with their classification. For object detectors in FSL, recent work have found that pretraining and fine-tuning are more effective than episodic approaches [Wang et al., 2020a, Bar et al., 2022], which motivated the development presented in the subsequent chapters.

		miniImageNet 5-way		tieredImageNet 5-way	
Model	Arch.	1-shot	5-shot	1-shot	5-shot
Gradient-based algorithms					
MAML [Finn et al., 2017]	C64	48.70 ± 1.84	63.11 ± 0.92	-	-
ANIL [Raghu et al., 2020]	C64	48.0 ± 0.7	62.2 ± 0.5	-	-
Meta-SGD [Li et al., 2017]	C64	**50.47** ± **1.87**	64.03 ± 0.94	-	-
TADAM [Oreshkin et al., 2018]	R12	58.50 ± 0.30	76.70 ± 0.30	-	-
MC [Park and Oliva, 2019]	R12	61.22 ± 0.10	75.92 ± 0.17	**66.20** ± **0.10**	**82.21** ± **0.08**
MetaOptNet [Lee et al., 2019]	R12	62.64 ± 0.61	78.63 ± 0.46	65.99 ± 0.72	81.56 ± 0.53
MATE [Chen et al., 2020e]	R12	62.08 ± 0.64	78.64 ± 0.46	-	-
MAML (Ours)	C64	47.93 ± 0.83	64.47 ± 0.69	50.08 ± 0.91	67.5 ± 0.79
MAML + reg. (Ours)	C64	49.15 ± 0.85	**66.43** ± **0.69**	51.5 ± 0.9	70.16 ± 0.76
MC (Ours)	C64	49.28 ± 0.83	63.74 ± 0.69	55.16 ± 0.94	71.95 ± 0.77
MC + reg. (Ours)	C64	49.64 ± 0.83	65.67 ± 0.70	**55.85** ± **0.94**	**73.34** ± **0.76**
MAML (Ours)	R12	63.52 ± 0.20	81.24 ± 0.14	63.96 ± 0.23	81.79 ± 0.16
MAML + reg. (Ours)	R12	**64.04** ± **0.22**	**82.45** ± **0.14**	64.32 ± 0.23	81.28 ± 0.11
Metric-based algorithms					
ProtoNet [Snell et al., 2017]	C64	46.61 ± 0.78	65.77 ± 0.70	–	–
IMP [Allen et al., 2019]	C64	49.6 ± 0.8	**68.1** ± **0.8**	–	–
SimpleShot [Wang et al., 2019]	C64	49.69 ± 0.19	66.92 ± 0.17	51.02 ± 0.20	68.98 ± 0.18
Relation Nets [Sung et al., 2018]	C64	50.44 ± 0.82	65.32 ± 0.70	–	–
SimpleShot [Wang et al., 2019]	R18	**62.85** ± **0.20**	**80.02** ± **0.14**	**69.09** ± **0.22**	**84.58** ± **0.16**
CTM [Li et al., 2019]	R18	62.05 ± 0.55	78.63 ± 0.06	64.78 ± 0.11	81.05 ± 0.52
DSN [Simon et al., 2020]	R12	62.64 ± 0.66	78.83 ± 0.45	66.22 ± 0.75	82.79 ± 0.48
ProtoNet (Ours)	C64	49.53 ± 0.41	65.1 ± 0.35	51.95 ± 0.45	71.61 ± 0.38
ProtoNet + norm. (Ours)	C64	50.29 ± 0.41	67.13 ± 0.34	**54.05** ± **0.45**	71.84 ± 0.38
IMP (Ours)	C64	48.85 ± 0.81	66.43 ± 0.71	52.16 ± 0.89	71.79 ± 0.75
IMP + norm. (Ours)	C64	**50.69** ± **0.8**	67.29 ± 0.68	53.46 ± 0.89	**72.38** ± **0.75**
ProtoNet (Ours)	R12	59.25 ± 0.20	77.92 ± 0.14	41.39 ± 0.21	83.06 ± 0.16
ProtoNet + norm. (Ours)	R12	**62.69** ± **0.20**	**80.95** ± **0.14**	**68.44** ± **0.23**	**84.20** ± **0.16**

Table 4.1. *Performance comparison of FSC models on FSC benchmarks.* For a given architecture, ***bold*** *values are the highest accuracy and* *underlined values* *are near-highest accuracies (at less than 1-point lower than the highest one).*

Dataset	Episodes	without Norm., $\lambda_1 = 0$	with Norm., $\lambda_1 = 0$	with Norm., $\lambda_1 = 1$
Omniglot	20-way 1-shot	$95.56 \pm 0.10\%$	$\mathbf{95.89 \pm 0.10}\%$	$91.90 \pm 0.14\%$
	20-way 5-shot	$\mathbf{98.80 \pm 0.04}\%$	$\mathbf{98.80 \pm 0.04}\%$	$96.40 \pm 0.07\%$
miniImageNet	5-way 1-shot	$49.53 \pm 0.41\%$	$\mathbf{50.29 \pm 0.41}\%$	$49.43 \pm 0.40\%$
	5-way 5-shot	$65.10 \pm 0.35\%$	$\mathbf{67.13 \pm 0.34}\%$	$65.71 \pm 0.35\%$
tieredImageNet	5-way 1-shot	$51.95 \pm 0.45\%$	$\mathbf{54.05 \pm 0.45}\%$	$53.54 \pm 0.44\%$
	5-way 5-shot	$\mathbf{71.61 \pm 0.38}\%$	$\mathbf{71.84 \pm 0.38}\%$	$70.30 \pm 0.40\%$

Table 4.2. *Performance of ProtoNet with and without our regularization on the entropy and/or normalization. All accuracy results (in %) are averaged over 2400 test episodes and 4 different random seeds and are reported with* 95% *confidence interval. Further enforcing regularization on the singular values can be detrimental to performance.*

Dataset	Episodes	Original	$\lambda_1 = 0$	$\lambda_1 = 1$	$\lambda_1 = 0.1$	$\lambda_1 = 0.01$	$\lambda_1 = 0.001$	$\lambda_1 = 0.0001$
miniImageNet	5-way 1-shot	$49.53 \pm 0.41\%$	$\mathbf{50.29 \pm 0.41}\%$	$49.43 \pm 0.40\%$	$\mathbf{50.19 \pm 0.41}\%$	$\mathbf{50.44 \pm 0.42}\%$	$\mathbf{50.46 \pm 0.42}\%$	$\mathbf{50.45 \pm 0.42}\%$
	5-way 5-shot	$65.10 \pm 0.35\%$	$\mathbf{67.13 \pm 0.34}\%$	$65.71 \pm 0.35\%$	$66.69 \pm 0.36\%$	$66.69 \pm 0.34\%$	$\mathbf{67.2 \pm 0.35}\%$	$\mathbf{67.12 \pm 0.35}\%$
Omniglot	20-way 1-shot	$95.56 \pm 0.10\%$	$\mathbf{95.89 \pm 0.10}\%$	$91.90 \pm 0.14\%$	$94.38 \pm 0.12\%$	$95.60 \pm 0.10\%$	$95.7 \pm 0.10\%$	$95.77 \pm 0.10\%$
	20-way 5-shot	$98.80 \pm 0.04\%$	$98.80 \pm 0.04\%$	$96.40 \pm 0.07\%$	$97.93 \pm 0.05\%$	$98.62 \pm 0.04\%$	$98.76 \pm 0.04\%$	$\mathbf{98.91 \pm 0.03}\%$

Table 4.3. *Ablative study on the strength of the regularization with normalized ProtoNet. All accuracy results (in %) are averaged over 2400 test episodes and 4 random seeds and are reported with* 95% *confidence interval.*

	$\lambda_1 = 0$	$\lambda_1 = 1$
$\lambda_2 = 0$	$91.72 \pm 0.29\%$	$89.86 \pm 0.31\%$
	$97.07 \pm 0.14\%$	$72.47 \pm 0.17\%$
$\lambda_2 = 1$	$92.80 \pm 0.26\%$	$\mathbf{95.67 \pm 0.20\%}$
	$96.99 \pm 0.14\%$	$\mathbf{98.24 \pm 0.10\%}$

(a)

	$\lambda_1 = 0$	$\lambda_1 = 1$
$\lambda_2 = 0$	$47.93 \pm 0.83\%$	$47.76 \pm 0.84\%$
	$64.47 \pm 0.69\%$	$64.44 \pm 0.68\%$
$\lambda_2 = 1$	$48.27 \pm 0.81\%$	$\mathbf{49.16 \pm 0.85\%}$
	$64.16 \pm 0.72\%$	$\mathbf{66.43 \pm 0.69\%}$

(b)

Table 4.4. *Ablative study of the regularization parameter for MAML, on Omniglot* ***(a)*** *with 20-way 1-shot (*top values*) and 20-way 5-shot (*bottom values*) episodes, and miniImageNet* ***(b)*** *with 5-way 1-shot (*top values*) and 5-way 5-shot (*bottom values*) episodes. All accuracy results (in %) are averaged over 2400 test episodes and 4 different random seeds and are reported with 95% confidence interval. We can see that in all cases, using both regularization terms is important.*

		miniImageNet 5-way		tieredImageNet 5-way	
Model	Arch.	1-shot	5-shot	1-shot	5-shot
MTL	R12	55.73 ± 0.18	76.27 ± 0.13	62.49 ± 0.21	81.31 ± 0.15
MTL + norm.	R12	59.49 ± 0.18	**77.3 ± 0.13**	**66.66 ± 0.21**	**83.59 ± 0.14**
MTL + reg. (Ours)	R12	**61.12 ± 0.19**	76.62 ± 0.13	66.28 ± 0.22	81.68 ± 0.15

Table 4.5. *Performance comparison of MTL models [Wang et al., 2021b] on FSC benchmarks with the ResNet-12 backbone architecture.* ***Bold*** *values are the highest accuracy and underlined values are near-highest accuracies (at less than 1-point lower than the highest one).*

λ_1	1	0.8	0.6	0.4	0.2	0.1	0.05	0.01	0
Accuracy ($\lambda_2 = 1$)	75.84	75.85	76.02	76.11	76.15	75.99	75.65	75.08	74.64
λ_2	1	0.8	0.6	0.4	0.2	0.1	0.05	0.01	0
Accuracy ($\lambda_1 = 1$)	75.84	76.09	75.81	76.28	76.23	76.1	76.25	**76.42**	76.06

Table 4.6. *Performance of MTL [Wang et al., 2021b] when varying either λ_1 or λ_2, the other being fixed to 1, on the miniImageNet 5-way 5-shot benchmark. All accuracy results (in %) are averaged over 2000 test episodes on a single random seed.*

5 Unsupervised Pretraining for Object Detection with Fewer Annotation

Contents

The present chapter investigates unsupervised pretraining specifically designed for Object Detection with Transformers. We recall the positioning in the general training pipeline in Figure 5.1. The objective is to propose a strong pretraining strategy for a more efficient fine-tuning of these detectors with few labeled data. The chapter is organized as follows. In Section 5.2, we review previous work on unsupervised pretraining with a focus on application for object detection, and discuss the positioning of our method. Then, in Section 5.3, we introduce our proposed ProSeCo, *and detail the localization-aware contrastive loss used in Section 5.3.3. In Section 5.4, we present results on different benchmarks with a focus on learning with fewer data with several ablation studies. We conclude this chapter in Section 5.5.*

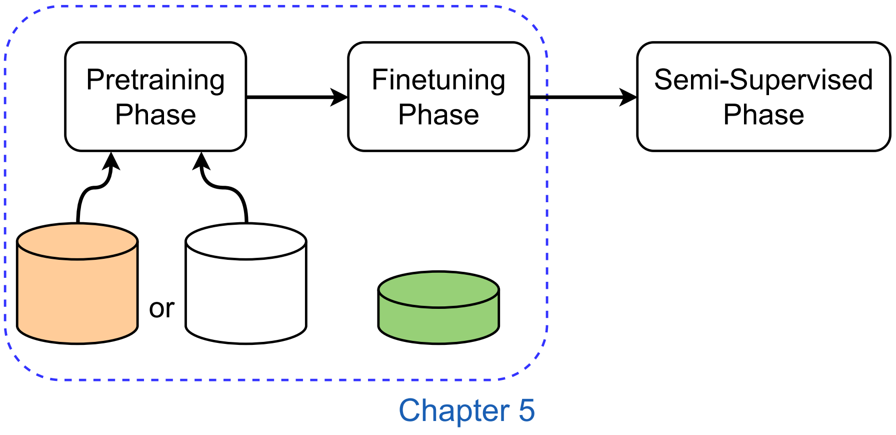


Figure 5.1. *Illustration of the position of our contributions from this chapter in the full training pipeline presented in Chapter 1. We are interested on Unsupervised Pretraining for Object Detection.*

5.1 Introduction

In this chapter, we address the object detection task in the scarce data regime. As shown in the recent literature, unsupervised pretraining which consists in initializing a model with a pretrained backbone, helps train big architectures more efficiently [Chen et al., 2020c, Caron et al., 2020, He et al., 2020]. While gathering data is not difficult in most cases, its labeling is always time-consuming and costly. Pretraining leverages huge amounts of unlabeled data and helps achieve better performance with fewer data and fewer iterations, when finetuning the pretrained models afterwards.

The design of pretraining tasks for dense problems such as Object Detection has to take into account the fine-grained information in the image. Furthermore, complex object detectors contain different specific parts that can be either pretrained *independently* [Xiao et al., 2021, Xie et al., 2021, Wang et al., 2021d, Hénaff et al., 2021, Dai et al., 2021b, Bar et al., 2022] or *jointly* [Wei et al., 2021b]. The different pretraining possibilities for Object Detection in the literature are illustrated in Figure 5.2. A pretraining of the backbone tailored to dense tasks has been the subject of many recent efforts [Xiao et al., 2021, Xie et al., 2021, Wang et al., 2021d, Hénaff et al., 2021] (*Backbone Pretraining*), but few have been interested in incorporating the detection-specific components of the architectures during pretraining [Dai et al., 2021b, Bar et al., 2022, Wei et al., 2021b] (*Overall Pretraining*). Among them, SoCo [Wei et al., 2021b] focuses on convolutional architectures and pretrains the whole detector, *i.e.* the backbone along with the detection heads (approach *e.* in Figure 5.2), whereas UP-DETR [Dai et al., 2021b] and DETReg [Bar et al., 2022] pretrain only the transformers [Vaswani et al., 2017] in transformer-based object detectors [Carion et al., 2020, Zhu et al., 2021] and keep the backbone fixed (approach *c.* in Figure 5.2).

Due to the numerous parameters that must be learned and the huge number of iterations needed because of random initialization, pretraining the entire detection model is expensive (Figure 5.2, *e.*). On the other hand, pretraining only the detection-specific parts with a fixed backbone leads to fewer parameters and allows leveraging strong pretrained backbones already available. However, fully relying on aligning embeddings given by the fixed backbone during pretraining and those given by the detection head, as done in DETReg or UP-DETR, introduces a discrepancy in the information contained in the features (Figure 5.2, *c.*). Indeed, while the pretrained backbone has been trained to learn image-level features, the object detector must understand object-level information in the image. Aligning inconsistent features hinders the pretraining quality.

In this chapter, we propose our second contribution, *Proposal Selection Contrast* (ProSeCo), an unsupervised pretraining method using transformer-based detectors with a fixed pretrained backbone. ProSeCo makes use of two models. The first one aims to alleviate the discrepancy in the features by maintaining a copy of the whole detection model. This model is referred to as a *teacher* in charge of the *object proposals* embeddings, and is updated through an Exponential Moving Average (EMA) of another *student* network making the object predictions and using a similar architecture. This latter network is trained by a contrastive learning approach that leverages the high number of object proposals that can be obtained from the detectors. This methodology, in addition to the absence of batch normalization in the architectures, reduces the need for a large batch size. We further adapt the contrastive loss commonly used in pretraining to take into account the locations of the object proposals in the image, which is crucial in object detection. In addition, the localization task is independently learned through region proposals generated by Selective Search [Uijlings et al., 2013]. Our contributions are summarized as:

- We propose *Proposal Selection Contrast (ProSeCo)*, a contrastive learning method tailored for pretraining transformer-based object detectors.
- We introduce the information of the localization of object proposals for the selection of positive examples in the contrastive loss to improve its efficiency for pretraining.
- We show that our proposed ProSeCo outperforms previous pretraining methods for transformer-based object detectors on standard benchmarks as well as novel benchmarks.

Before detailing these contributions, we first provide a review of specific related work.

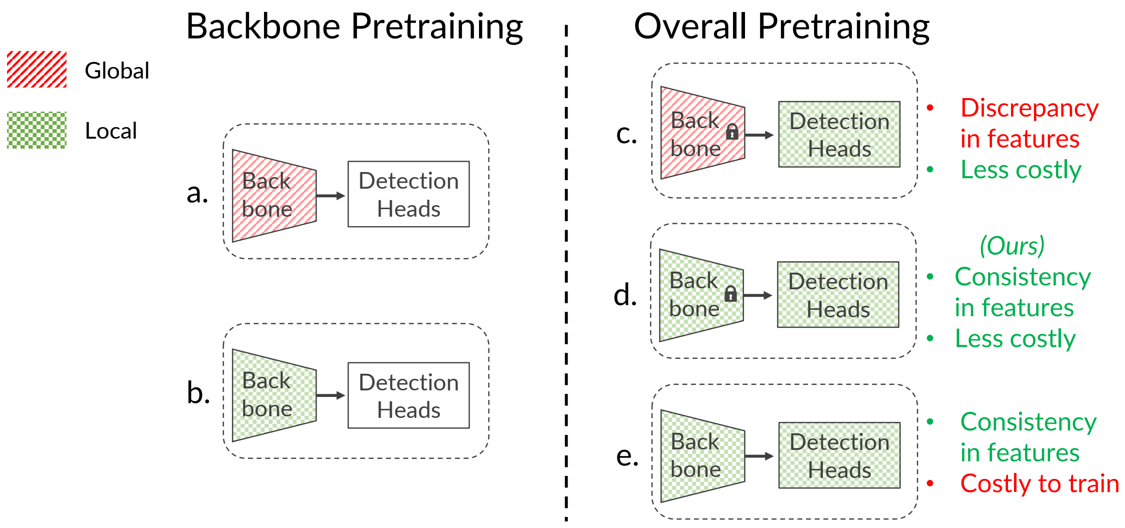


Figure 5.2. *Illustration of the different pretraining possibilities for Object Detection. The pretraining can be either limited to the backbone (*left*), or overall including the detection heads (*right*). The few previous overall approaches either suffer from a discrepancy in the features between the backbone, that is trained at the image-level (*global*), and the detection heads, trained to encode object-level (*local*) information (*c.*), or from the cost of training lots of parameters with a large batch size (*e.*).*

5.2 Related Work

Supervised Object Detection with transformer-based architectures As presented in Chapters 2 and 3, Object Detection is an important and extensively researched problem in computer vision [Girshick et al., 2014, Girshick, 2015, Ren et al., 2015, Redmon et al., 2016, Liu et al., 2016, Lin et al., 2017a, Tian et al., 2019]. It essentially combines object localization and classification tasks. We are particularly interested in this chapter and the following in the recent detector family based on an encoder-decoder architecture using Transformers [Vaswani et al., 2017], the DETR [Carion et al., 2020] family of transformer-based architectures introduced in Chapter 2. The training complexity of this architecture was subsequently improved in Deformable DETR (Def. DETR) [Zhu et al., 2021], by changing the attention operations into deformable attention, resulting in a faster convergence speed. Several other follow-up works [Dai et al., 2021a, Meng et al., 2021, Wang et al., 2021e, Liu et al., 2022a, Yang et al., 2022, Li et al., 2022] have also focused on increasing the training efficiency of DETR. Transformer-based architectures now represent a strong alternative to traditional convolutional object detectors, reaching better performance for a similar training cost. Furthermore, recent work have shown strong results of transformer-based detectors in a data-scarce setting [Bar et al., 2022], compared to convolutional architectures [Liu et al., 2021], which we also observe and discuss in Chapter 6.

Unsupervised Pretraining for detection models We introduced in Chapter 3 recent pretraining methods tailored for OD, but few of them have tackled the problem of pretraining detection-specific parts of the architectures. We discuss their limitations here. SoCo [Wei et al., 2021b] proposes a contrastive pretraining strategy for convolutional detectors inspired by BYOL [Grill et al., 2020]. Object locations are generated using Selective Search [Uijlings et al., 2013], and then object-level features are extracted and contrasted with each other using a *teacher-student* strategy. The small amount of object proposals and object features generated requires using a large batch size for the contrastive loss to be effective. Pretraining the backbone along with the detection modules this way makes the method difficult and costly to train due to the high amount of parameters to learn. For transformer-based architectures, UP-DETR [Dai et al., 2021b] and DETReg [Bar et al., 2022] use a fixed pretrained backbone to extract features respectively from random patch, or from object locations given by Selective Search [Uijlings et al., 2013], then pretrain the detector by locating and reconstructing the features of the patch extracted from the input images. However, since the features to reconstruct are obtained by a backbone which was trained to encode image-level information, there is a discrepancy in the information between the features to match.

Our proposed ProSeCo is designed specifically for transformer-based detectors, and use a fixed backbone pretrained for *local information*. In this work, we leverage the high amount of object proposals generated by transformer-based detectors as instances for contrastive learning. Target object-level features and localizations are provided by a *teacher* detection model updated through EMA, inspired by recent advances in self-supervised and semi-supervised learning [Liu et al., 2021, Grill et al., 2020, Denize et al., 2023, Wei et al., 2021b]. The *student* detection model is pretrained by computing the contrastive loss between *object proposals* given by the student and teacher detectors. The large number of proposals generated by transformer-based detectors alleviates the need for a large batch size for the contrastive loss. The contrastive loss function used is further improved by introducing the location of objects for selecting positive proposals.

5.3 Overview of the approach

We present in this section our proposed unsupervised pretraining approach, illustrated in Figure 5.3, beginning with the data processing pipeline. Then, we detail the contrastive loss used with the localization-aware positive object proposal selection. The transformer-based detectors are built on a general architecture that consists of a back-

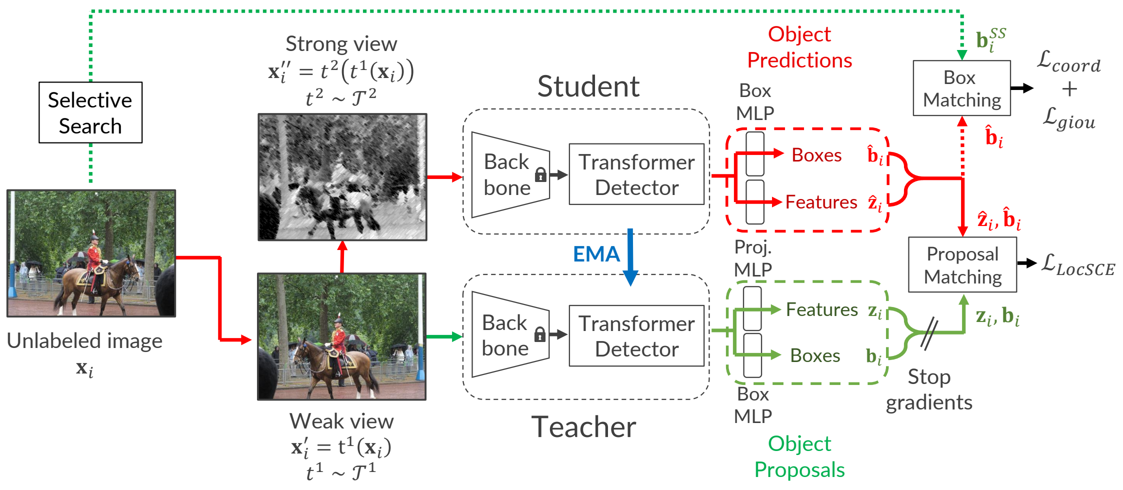


Figure 5.3. *Overview of our proposed* ProSeCo *for unsupervised pretraining. The method follows a* student-teacher architecture, *with the teacher updated through an Exponential Moving Average (EMA) of the student. For each input image, K random boxes are computed using the* Selective Search *algorithm, and two different views are generated through an asymmetric set of* weak augmentations $\mathcal{T}^1$ *and* strong augmentations $\mathcal{T}^2$*. Then,* object predictions *are obtained from the student model for the strongly augmented view, and* object proposals *from the teacher model with the weakly augmented view. Finally, the boxes predicted by the student are matched with the boxes sampled from Selective Search to compute the localization losses $\mathcal{L}_{coord}$ and $\mathcal{L}_{giou}$, and the full predictions are matched with the object proposals to compute* our novel contrastive loss $\mathcal{L}_{LocSCE}$ *between them.*

bone encoder (*e.g.* a ResNet-50), followed by several transformers encoder and decoder layers, and finally two prediction heads for the bounding boxes coordinates and class logits [Carion et al., 2020, Zhu et al., 2021].

5.3.1 Data processing pipeline

Throughout the section, we assume to have sampled a batch of unlabeled images $\mathbf{x} = \{\mathbf{x}_i\}_{i=1}^{N_b}$, with $\mathbf{x}_i$ the i^{th} image and N_b the batch size.

For each input image $\mathbf{x}_i$, we compute two different views with two asymmetric distributions of augmentations[1] $\mathcal{T}^1$ and $\mathcal{T}^2$: a *weakly augmented view* $\mathbf{x}_i{}' = t^1(\mathbf{x}_i)$, with $t^1 \sim \mathcal{T}^1$, and a *strongly augmented view* obtained from the weakly augmented one $\mathbf{x}_i{}'' = t^2(t^1(\mathbf{x}_i))$, with $t^2 \sim \mathcal{T}^2$.

To guide the model into discovering locations of objects in unlabeled images, we use

[1]The exact sets of augmentations chosen are given in Table 5.1.

bounding boxes obtained from the *Selective Search* algorithm [Uijlings et al., 2013], similarly to previous work [Wei et al., 2021b, Bar et al., 2022]. Since Selective Search is deterministic and the generated boxes are not ordered, we compute the boxes for all images offline and, at training time, randomly sample K boxes $\mathbf{b}_i^{SS} = \{\mathbf{b}_{(i,j)}^{SS} \in \mathbb{R}^4\}_{j=1}^{K}$ for each image in the batch. Then, the two views and the corresponding boxes sampled are used as input for our method.

5.3.2 Pretraining method

As shown in Figure 5.3, our approach is composed of a *student-teacher architecture* [Grill et al., 2020, Denize et al., 2023, Wei et al., 2021b]. With ProSeCo, we extend the student-teacher pretraining for transformer-based detectors to tackle the discrepancy in information-level when aligning features, and introduce a *dual unsupervised bipartite matching* presented below.

First, the backbone and the detection heads are respectively initialized from pretrained weights and randomly for both the student and teacher models. The teacher model is updated through an *Exponential Moving Average (EMA)* of the student's weights at every training iteration:

$$\theta_t \leftarrow \alpha\theta_t + (1-\alpha)\theta_s. \tag{5.1}$$

where θ_t and θ_s are respectively the teacher's and student's weights, and $\alpha \in [0,1]$ is the *keep rate parameter*. This way, the teacher model represents a slightly delayed but more stable version of the student. When $\alpha = 1$, the teacher is not updated and remains constant. When $\alpha = 0$, all weights are updated to be equal to those of the student. Naturally, there is a trade-off between a too high and too low keep rate parameter.

For both models, the classification heads in the detectors are replaced by an MLP, called *projector*, to obtain latent representations of the objects.

From the weakly augmented view $\mathbf{x}_i{}'$, the teacher model provides *object proposals* $\mathbf{y}_i = \{\mathbf{y}_{(i,j)}\}_{j=1}^{N} = \{(\mathbf{z}_{(i,j)}, \mathbf{b}_{(i,j)}\}_{j=1}^{N}$, with $\mathbf{z}_{(i,j)}$ the latent embedding and $\mathbf{b}_{(i,j)}$ the coordinates of the j^{th} object found. The student model infers predictions $\hat{\mathbf{y}}_i = \{\hat{\mathbf{y}}_{(i,j)}\}_{j=1}^{N} = \{(\hat{\mathbf{z}}_{(i,j)}, \hat{\mathbf{b}}_{(i,j)}\}_{j=1}^{N}$ from the corresponding strongly augmented view $\mathbf{x}_i{}''$.

Then, we apply an *unsupervised* Hungarian algorithm [Munkres, 1957] for *proposal matching* to find from all the permutations of N elements $\mathfrak{S}_N$, the optimal bipartite matching $\hat{\sigma}_i^{\text{prop}}$ between the predictions $\hat{\mathbf{y}}_i$ of the student and the object proposals $\mathbf{y}_i$

of the teacher:

$$\hat{\sigma}_i^{\mathsf{prop}} = \arg\min_{\sigma \in \mathfrak{S}_N} \sum_{j=1}^{N} \mathcal{L}_{\mathsf{prop\_match}}(\mathbf{y}_{(i,j)}, \hat{\mathbf{y}}_{(i,\sigma(j))}). \tag{5.2}$$

Therefore, for each image $\mathbf{x}_i$, the j^{th} proposal $\mathbf{y}_{(i,j)}$ found by the teacher is associated to the $\hat{\sigma}_i^{\mathsf{prop}}(j)^{\mathsf{th}}$ prediction of the student $\hat{\mathbf{y}}_{(i,\hat{\sigma}_i^{\mathsf{prop}}(j))}$. Our matching cost $\mathcal{L}_{\mathsf{prop\_match}}$ for the Hungarian algorithm takes into account both features and bounding box predictions through a linear combination of features similarity $\mathcal{L}_{\mathsf{sim}}$, the ℓ_1 loss of the box coordinates $\mathcal{L}_{\mathsf{coord}}$, and the generalized IoU loss $\mathcal{L}_{\mathsf{giou}}$ from [Rezatofighi et al., 2019]:

$$\mathcal{L}_{\mathsf{sim}}(\mathbf{z}_{(i,j)}, \hat{\mathbf{z}}_{(i,\sigma(j))}) = \frac{\langle \mathbf{z}_{(i,j)}, \hat{\mathbf{z}}_{(i,\sigma(j))} \rangle}{\|\mathbf{z}_{(i,j)}\|_2 \cdot \|\hat{\mathbf{z}}_{(i,\sigma(j))}\|_2} \tag{5.3}$$

$$\mathcal{L}_{\mathsf{coord}}(\mathbf{b}_{(i,j)}, \hat{\mathbf{b}}_{(i,\hat{\sigma}_i(j))}) = \|\mathbf{b}_{(i,j)} - \hat{\mathbf{b}}_{(i,\hat{\sigma}_i(j))}\|_1 \tag{5.4}$$

$$\begin{aligned}\mathcal{L}_{\mathsf{prop\_match}}(\mathbf{y}_{(i,j)}, \hat{\mathbf{y}}_{(i,\sigma(j))}) = \Big[&\lambda_{\mathsf{sim}} \mathcal{L}_{\mathsf{sim}} \left(\mathbf{z}_{(i,j)}, \hat{\mathbf{z}}_{(i,\sigma(j))}\right) \\ &+ \lambda_{\mathsf{coord}} \mathcal{L}_{\mathsf{coord}} \left(\mathbf{b}_{(i,j)}, \hat{\mathbf{b}}_{(i,\sigma(j))}\right) + \lambda_{\mathsf{giou}} \mathcal{L}_{\mathsf{giou}} \left(\mathbf{b}_{(i,j)}, \hat{\mathbf{b}}_{(i,\sigma(j))}\right)\Big].\end{aligned} \tag{5.5}$$

Similarly, we also use an *unsupervised* Hungarian algorithm for *box matching*, to find the optimal bipartite matching $\hat{\sigma}_i^{\mathsf{box}} \in \mathfrak{S}_N$ between the predicted boxes $\hat{\mathbf{b}}_i$ of the student and the sampled boxes $\mathbf{b}_i^{SS}$ from Selective Search, using the matching cost $\mathcal{L}_{\mathsf{box\_match}}$:

$$\hat{\sigma}_i^{\mathsf{box}} = \arg\min_{\sigma \in \mathfrak{S}_N} \sum_{j=1}^{N} \mathcal{L}_{\mathsf{box\_match}}(\mathbf{y}_{(i,j)}, \hat{\mathbf{y}}_{(i,\sigma(j))}), \tag{5.6}$$

$$\mathcal{L}_{\mathsf{box\_match}}(\mathbf{b}_{(i,j)}^{SS}, \hat{\mathbf{b}}_{(i,\sigma(j))}) = \lambda_{\mathsf{coord}} \mathcal{L}_{\mathsf{coord}} \left(\mathbf{b}_{(i,j)}^{SS}, \hat{\mathbf{b}}_{(i,\sigma(j))}\right) + \lambda_{\mathsf{giou}} \mathcal{L}_{\mathsf{giou}} \left(\mathbf{b}_{(i,j)}^{SS}, \hat{\mathbf{b}}_{(i,\sigma(j))}\right). \tag{5.7}$$

Finally, the global unsupervised loss $\mathcal{L}_u$ used for training is a combination of a loss function between the object latent embeddings of the teacher and student models, and between the object localization of the student predictions and Selective Search boxes.

More formally, it is computed as:

$$\mathcal{L}_u(\mathbf{x}) = \lambda_{\text{contrast}}\mathcal{L}_{\text{LocSCE}}\left(\mathbf{y}, \hat{\mathbf{y}}, \hat{\sigma}^{\text{prop}}\right) + \\ \frac{1}{N_b K}\sum_{i=1}^{N_b}\Bigg[\sum_{j=1}^{K}\lambda_{\text{coord}}\mathcal{L}_{\text{coord}}\left(\mathbf{b}^{SS}_{(i,j)}, \hat{\mathbf{b}}_{(i,\hat{\sigma}^{\text{box}}_i(j))}\right) + \sum_{j=1}^{K}\lambda_{\text{giou}}\mathcal{L}_{\text{giou}}\left(\mathbf{b}^{SS}_{(i,j)}, \hat{\mathbf{b}}_{(i,\hat{\sigma}^{\text{box}}_i(j))}\right)\Bigg]. \tag{5.8}$$

In the above equations, we define $\lambda_{\text{sim}}, \lambda_{\text{coord}}, \lambda_{\text{giou}}, \lambda_{\text{contrast}} \in \mathbb{R}^+$ as the coefficients for the different losses. For the consistency in the latent embeddings of the objects, we introduce the object locations information in our contrastive loss $\mathcal{L}_{\text{LocSCE}}$. This loss is used to contrast the predictions $\hat{\mathbf{y}} = \{\hat{\mathbf{y}}_i\}_{i=1}^{N_b}$ of the student with object proposals $\mathbf{y} = \{\mathbf{y}_i\}_{i=1}^{N_b}$ found by the teacher, matched according to the proposal matching $\hat{\sigma}^{\text{prop}} = \{\hat{\sigma}^{\text{prop}}_i\}_{i=1}^{N_b}$ over the batch. We detail the computations behind this loss in the next section.

5.3.3 Localization-aware contrastive loss

Inspired by advances in self-supervised learning, we propose a contrastive objective function [Oord et al., 2018, Chen et al., 2020b, He et al., 2020] between object proposals that also maintains relations [Zheng et al., 2021, Denize et al., 2023] among these proposals. This objective function extends the latest SCE [Denize et al., 2023], as well as the original InfoNCE [Oord et al., 2018], for *instance discrimination between object proposals* and is illustrated in Figure 5.4. We compute the contrastive loss between all the object latent embeddings from a batch of image. The *positive pair* of objects in an image $\mathbf{x}_i$ is given by the proposal matching σ^{prop}_i. First, we define the distributions of similarity between objects embeddings :

$$p'_{(in,jm)} = \frac{\mathbb{1}_{i\neq n}\mathbb{1}_{j\neq m}\exp(\mathbf{z}_{(i,j)}\cdot\mathbf{z}_{(n,m)}/\tau_t)}{\sum_{k=1}^{N_b}\sum_{l=1}^{N}\mathbb{1}_{i\neq k}\mathbb{1}_{j\neq l}\exp(\mathbf{z}_{(i,j)}\cdot\mathbf{z}_{(k,l)}/\tau_t)}, \tag{5.9}$$

$$p''_{(in,jm)} = \frac{\exp(\mathbf{z}_{(i,j)}\cdot\hat{\mathbf{z}}_{(n,m)}/\tau)}{\sum_{k=1}^{N_b}\sum_{l=1}^{N}\exp(\mathbf{z}_{(i,j)}\cdot\hat{\mathbf{z}}_{(k,l)}/\tau)}. \tag{5.10}$$

The distribution $p'_{(in,jm)}$ represents the *relations* between weakly augmented object embeddings scaled by the temperature τ_t, and $p''_{(in,jm)}$ the similarity between the strongly augmented embeddings and the weakly augmented ones, scaled by τ. Then, the *target similarity distribution* for the objective function is a weighted combination of one-hot label and the teacher's embeddings relations:

$$w_{(in,jm)} = \lambda_{\text{SCE}}\cdot\mathbb{1}_{i=n}\mathbb{1}_{j=m} + (1-\lambda_{\text{SCE}})\cdot p'_{(in,jm)}. \tag{5.11}$$

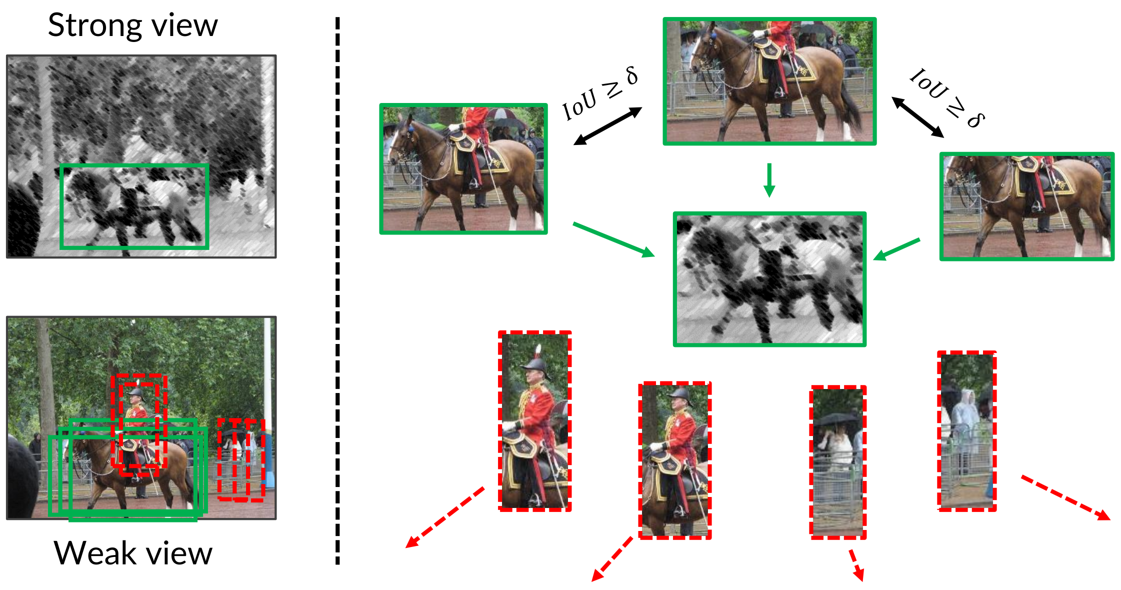


Figure 5.4. *Illustration of the effect of the* localized contrastive loss *used. Predictions of the student and teacher models are contrasted with each other to leverage the large number of object proposals obtained from transformer-based detectors. To introduce the* object locations *information, overlapping proposals* (in green) *in each weak view, according to an* IoU threshold δ*, are also considered as positive along with the matched proposal. Proposals that neither match nor overlap the matched proposal, are considered as negative* (in red) *in the contrastive loss.*

To introduce the location information in our objective, we compute the pairwise *Intersection over Union* (IoU) between object proposals of the same image and consider overlapping objects as other positives when computing the target similarity distribution:

$$w^{\text{Loc}}_{(in,jm)} = \lambda_{\text{SCE}} \cdot \mathbb{1}_{i=n}\mathbb{1}_{IoU_i(j,m)\geq\delta} + (1-\lambda_{\text{SCE}}) \cdot p'_{(in,jm)}, \tag{5.12}$$

where $IoU_i(j,m)$ corresponds to the IoU between teacher proposals $\mathbf{y}_{(i,j)}$ and $\mathbf{y}_{(i,m)}$ found in the same image $\mathbf{x}_i$, and δ is the *IoU threshold* to consider the proposal as a positive example. Finally, we use this tailored target similarity distribution in our *Localized SCE* (LocSCE) loss:

$$\mathcal{L}_{\text{LocSCE}}(\mathbf{y}, \hat{\mathbf{y}}, \hat{\sigma}^{\text{prop}}) = -\frac{1}{N_b N}\sum_{i=1}^{N_b}\sum_{n=1}^{N_b}\sum_{j=1}^{N}\sum_{m=1}^{N} w^{\text{Loc}}_{(in,jm)} \log(p''_{(in,j\hat{\sigma}^{\text{prop}}_n(m))}). \tag{5.13}$$

Note that we do not match the proposals according to $\hat{\sigma}^{\text{prop}}$ in the target similarity, as we compare proposals obtained by the teacher model. We also require the full proposal as input of the loss to compute the pairwise IoU using the box coordinates.

We can also extend InfoNCE the same way:

$$\mathcal{L}_{\text{InfoNCE}}(\mathbf{z}, \hat{\mathbf{z}}, \hat{\sigma}^{\text{prop}}) = -\frac{1}{N_b N}\sum_{i=1}^{N_b}\sum_{j=1}^{N}\log\left(\frac{\exp(\mathbf{z}_{(i,j)}\cdot\hat{\mathbf{z}}_{(i,\hat{\sigma}_i^{\text{prop}}(j))}/\tau)}{\sum_{k=1}^{N_b}\sum_{l=1}^{N}\exp(\mathbf{z}_{(i,j)}\cdot\hat{\mathbf{z}}_{(k,l)}/\tau)}\right). \quad (5.14)$$

Similarly to our *LocSCE*, the localization of the objects can be introduced in the InfoNCE loss function to obtain the *LocNCE* objective as follows:

$$\mathcal{L}_{\text{LocNCE}}(\mathbf{y}, \hat{\mathbf{z}}, \hat{\sigma}^{\text{prop}}) = -\frac{1}{N_b N}\sum_{i=1}^{N_b}\sum_{j=1}^{N}\sum_{m=1}^{N}\log\left(\frac{\mathbb{1}_{IoU_i(j,m)\geq\delta}\exp(\mathbf{z}_{(i,j)}\cdot\hat{\mathbf{z}}_{(i,\hat{\sigma}_i^{\text{prop}}(m))}/\tau)}{\sum_{k=1}^{N_b}\sum_{l=1}^{N}\exp(\mathbf{z}_{(i,j)}\cdot\hat{\mathbf{z}}_{(k,l)}/\tau)}\right). \quad (5.15)$$

We recover the original formulations of both SCE and InfoNCE when $\delta = 1$. These formulations lead to an effective batch size of $N_b{\cdot}N$. The localization-aware contrastive loss functions aim to pull together the objects embeddings that overlap subsequently with each others, as they should correspond to the same object in the image.

5.4 Experiments

In this section, we present a comparative study of the results of our proposed method on standard and novel benchmarks for learning with fewer data, as well as an ablative study on the most relevant parts. First, we recall the datasets introduced in Chapter 3 with the corresponding evaluation and training settings.

5.4.1 Implementation details

Datasets and evaluation We use different datasets throughout this work that we present here.

- For pretraining purposes, we use the standard *ImageNet ILSVRC 2012 (IN)* [Russakovsky et al., 2015] dataset, which contains 1.2M training images separated in 1000 class categories.

- We also use the *MS-COCO (COCO)* [Lin et al., 2014] dataset for pretraining in ablation studies, but mainly for finetuning and evaluation purposes. This dataset contains 80 classes of objects and about 118k training images *(train2017 subset)*. Performance is evaluated on the *val2017* subset.

- For finetuning, we also use the *PASCAL VOC 2007 and 2012 (VOC 07-12)* [Everingham et al., 2010] datasets. This dataset contains 20 classes of objects, and we use the combination of the *trainval* subsets from both VOC2007 and VOC2012 for training, corresponding to about 16k training images in total. Performance is evaluated on the *test* subset from VOC2007.

- For a more complicated dataset, we use the *Few-Shot Object Detection* dataset [Fan et al., 2020a]. Since the dataset is designed as *open-set*, *i.e.* with different classes between training and testing, we separately use the *train* and *test* sets for benchmarking. We separate the test set into a training and testing subset, by randomly taking 80% of images for training and the 20% remaining for testing, and do the same for the train set. We make sure that all classes appears at least once in both training and testing subsets. The images selected for training and testing will be made available for reproducibility. This separation leads to about 11k training images and 3k testing images for the *test* set and about 42k training images and 10k testing images for the *train* set.

To evaluate the performance in learning with fewer data, following previous work [Wei et al., 2021b, Bar et al., 2022], we consider the *Mini-COCO* benchmarks, where we randomly sample 1%, 5% or 10% of the training data. Similarly, we also introduce the novel *Mini-VOC* benchmark, in which we randomly sample 5% or 10% of the training data. We also use the Few-Shot Object Detection (FSOD) dataset [Fan et al., 2020a] in novel *FSOD-test* and *FSOD-train* benchmarks. In all benchmarks, the image ids selected for training and testing will be made available for reproducibility.

Pretraining We initialize the backbone with the publicly available pretrained SCRL [Roh et al., 2021] checkpoint and pretrain ProSeCo for 10 epochs on IN. The hyperparameters are set as follows: the EMA keep rate parameter to 0.999, the IoU threshold $\delta = 0.5$, a batch size of $N_b = 64$ images over 8 A100 GPUs, and the coefficients in the different losses $\lambda_{\text{sim}} = \lambda_{\text{contrast}} = 2$ which is the same value used for the coefficient governing the class cross-entropy in the supervised loss. The projector is defined as a 2-layer MLP with a hidden layer of 4096 and a last layer of 256, without batch normalization. Following SCE [Denize et al., 2023], we set the temperatures $\tau = 0.1, \tau_t = 0.07$ and the coefficient $\lambda_{\text{SCE}} = 0.5$. Similarly to DETReg [Bar et al., 2022], we sample $K = 30$ boxes from the outputs of Selective Search for each image at every iteration. Other training and architecture hyperparameters are defined as in Def. DETR [Zhu et al., 2021] with, specifically, the coefficients $\lambda_{\text{coord}} = 5$ and $\lambda_{\text{giou}} = 2$, the number of object proposals (queries) $N = 300$, and the learning rate is set to $lr = 2 \cdot 10^{-4}$. For weak augmentations $\mathcal{T}^1$, we use a random combination of flip, resize and crop, and for strong augmentations $\mathcal{T}^2$, we use a random combination of color jittering, grayscale and Gaussian blur. In $\mathcal{T}^1$, we resize images with the same range of scales as the supervised training protocol on COCO (*Large-scale*).

Augmentations used We detail in Table 5.1 the distributions of augmentations used to create the weak view and the strong view. The *Weak Augmentations* follow standard supervised training for transformer-based detectors [Carion et al., 2020, Zhu et al., 2021]. The *Strong Augmentations* follow typical contrastive learning augmentations [Chen et al., 2020b, Bar et al., 2022].

Finetuning protocols For finetuning the pretrained models, we follow the standard supervised learning hyperparameters of Def. DETR [Zhu et al., 2021]. In all experiments, we train the models with a batch size of 32 images over 8 A100 GPUs until the validation performance stops increasing, *i.e.* for Mini-COCO, up to 2000 epochs for 1%, 500 epochs for 5%, 400 epochs for 10%, for Mini-VOC, up to 2000 epochs for both 5% and 10%, up to 100 epochs for both FSOD-test and PASCAL VOC, and up to 50 epochs for FSOD-train. We always decay the learning rates by a factor of 0.1 after about 80% of the number of training epochs. To compare our method to DETReg [Bar et al., 2022] on our novel benchmarks, we use their publicly available checkpoints from github.

5.4.2 Finetuning and transfer learning

Different datasets We evaluate the transfer learning ability of our pretrained model on several datasets. Tables 5.2a and 5.2b present the results obtained compared to previous methods in the literature when learning from fewer labeled data. We can see that our method outperforms state-of-the-art results in unsupervised pretraining on all benchmarks and datasets, and obtain even more strong results when training data is scarce. The improvement is even more significant as the overall performance with few training data is low. When using 5% of the COCO training data (*i.e.* Mini-COCO 5% in Table 5.2a), corresponding to about 5.9k images, ProSeCo achieves 28.8 mAP, which represents an improvement of +5.2 *percentage point* (p.p.) over the supervised pretraining baseline and +2 p.p. over both state-of-the-art overall pretraining methods.

Different models Then, we evaluate pretraining with other transformer-based detectors in different settings. In Table 5.3, we fine-tune with DAB-Def. DETR [Liu et al., 2022a] after pretraining with Def. DETR. We observe that ProSeCo still performs strongly when transferring the pretrained weights to other architectures. We can see that using a more consistent pretrained backbone is important for pretraining the overall detection model. In Table 5.4, we pretrain *and* fine-tune with Conditional DETR [Meng et al., 2021]. We can see that in the most scarce setting, pretraining

Weak Augmentations ($\mathcal{T}^1$)		
Augmentations	Probability	Parameters
Horizontal Flip	0.5	–
Resize	0.5	*Mid-scale:* short edge = range(320,481,16) *Large-scale:* short edge = range(480,801,32)
Resize	0.5	short edge $\in \{400, 500, 600\}$
Random Size Crop		min size = 384 ; max size = 600
Resize		*Mid-scale:* short edge = range(320,481,16) *Large-scale:* short edge = range(480,801,32)
Strong Augmentations ($\mathcal{T}^2$)		
Color Jitter	0.8	(brightness, contrast, saturation, hue) = (0.4, 0.4, 0.4, 0.1)
GrayScale	0.2	–
Gaussian Blur	0.5	(sigma x, sigma y) = (0.1, 2.0)

Table 5.1. *The different sets of augmentations used for each branch (*weak *or* strong*).* Probability *indicates the probability of applying the corresponding augmentation.*

the detection-specific parts with our ProSeCo is less effective than only pretraining the backbone with SCRL [Roh et al., 2021]. However, with more labels pretraining the overall detection models improves the performance. We can see that Conditional DETR has globally lower performance in FSL than Def. DETR, which might hinders overall pretraining in the scarcest setting.

5.4.3 Ablation Studies

In the following, we provide several ablation studies for our proposed approach. All experiments and results for the different ablations are compared on the Mini-COCO 5% benchmark with the pretrained SwAV backbone unless explicitly stated.

Method	Detector	Pretrain.	Mini-COCO		
		Dataset	1% (1.2k)	5% (5.9k)	10% (11.8k)
Supervised	Def. DETR	IN	13.0	23.6	28.6
SwAV [Caron et al., 2020]	Def. DETR	IN	13.3	24.5	29.5
SCRL [Roh et al., 2021]	Def. DETR	IN	16.4	26.2	30.6
DETReg [Bar et al., 2022]	Def. DETR	COCO	15.8	26.7	30.7
DETReg [Bar et al., 2022]	Def. DETR	IN	15.9	26.1	30.9
Supervised [Wei et al., 2021b]	Mask R-CNN	IN	–	19.4	24.7
SoCo* [Wei et al., 2021b]	Mask R-CNN	IN	–	26.8	31.1
ProSeCo (Ours)	Def. DETR	IN	**18.0**	**28.8**	**32.8**

(a)

Method	FSOD-test	FSOD-train	PASCAL VOC	Mini-VOC	
	100% (11k)	100% (42k)	100% (16k)	5% (0.8k)	10% (1.6k)
Supervised	39.3	42.6	59.5	33.9	40.8
DETReg [Bar et al., 2022]	43.2	43.3	63.5	43.1	48.2
ProSeCo (Ours)	**46.6**	**47.2**	**65.1**	**46.1**	**51.3**

(b)

Table 5.2. *Performance (mAP in %) of our proposed pretraining approach after fine-tuning using different percentage of training data (with the corresponding number of images reported). We show that our ProSeCo outperforms previous pretraining methods in all benchmarks.*

Pretraining dataset and backbone In Table 5.5a, we show the effect of changing the pretraining dataset or the fixed backbone used in our pretraining method. We can see that using a backbone more adapted to dense tasks that learned local information (*e.g.* SCRL) helps the model by having consistent features (+1 p.p.), compared to a backbone pretrained for global features (*e.g.* SwAV). Furthermore, even with a less adapted backbone, our ProSeCo initialized with the SwAV backbone outperforms DETReg (+1.7 p.p.). To compare with IN, we also pretrain ProSeCo on COCO for 120

Pretraining	Mini-COCO		
	1% (1.2k)	5% (5.9k)	10% (11.8k)
Supervised	16.7	27.0	31.0
DETReg	17.7	28.6	33.0
ProSeCo w/ SwAV	16.2	28.7	33.9
ProSeCo w/ SCRL	**19.4**	**29.9**	**34.1**

Table 5.3. *Performance comparison when transferring pretrained weights to DAB-Deformable DETR [Liu et al., 2022a], after fine-tuning on different fractions of labeled data.*

Pretraining	Mini-COCO		
	1% (1.2k)	5% (5.9k)	10% (11.8k)
Supervised	8.7	19.3	24.6
SCRL	**11.7**	20.9	26.2
ProSeCo	9.6	**22.4**	**27.5**

Table 5.4. *Performance comparison when pretraining and fine-tuning with Conditional DETR [Meng et al., 2021] on different fractions of labeled data.*

epochs. We obtain better results when pretraining the model on IN than using COCO thanks to the large number of different images in IN (about 10 times the number of images of COCO, leading to +0.4 p.p.), which is consistent with previous findings [Wei et al., 2021b]. The difference in the number of images might be slightly compensated in COCO by the closeness to the downstream task and the diversity of objects.

Location information in contrastive loss In Table 5.5b, we show the effect of the location information in the contrastive loss SCE. We can see that when introducing multiple positive examples for each image based on the IoU threshold δ (*i.e.* $\forall\delta < 1$), we achieve better results than with the original SCE loss (*i.e.* $\delta = 1$). Notably, the best results are achieved with $\delta = 0.5$ (+1.7 p.p.). We also compare the two different contrastive objectives considered for pretraining in Table 5.5b. We can see that using the InfoNCE loss leads to slightly better results (+0.3 p.p.). However, when introducing the location information, SCE benefits much more than InfoNCE ($+1.7$ p.p. compared to $+0.6$ p.p.). This might be that the selection of positives from the location helps to introduce easy positive examples, and thanks to this, the relational aspect of SCE can focus on the more difficult positives. In the end, LocSCE achieves stronger results than LocNCE ($+0.8$ p.p.).

Hyperparameters The Table 5.6 presents an ablation study on different important hyperparameters of our approach. We experimented first with the same batch size applied in [Bar et al., 2022] (*Abl. Batch*), but found that using a smaller batch size (*Base*) leads to improved results ($+0.2$ p.p.). We evaluated different image scales as a parameter of the weak data augmentations distribution in *Abl. Scales*. *Mid-scale* corresponds to a resizing of the images such that the shortest edge is between 320 and 480 pixels, as used in previous work [Dai et al., 2021b, Bar et al., 2022], and *Large-scale* to a resize between 480 and 800 pixels, used for supervised learning on COCO. Exact values for these parameters can be found in Table 5.1. We found that increasing the size of the images during pretraining is important to have more meaningful information in the boxes, and a more precise localization of the boxes ($+0.7$ p.p.). Following the best results from [Denize et al., 2023], we evaluated the performance for $\tau_t \in \{0.05, 0.07\}$. We found that $\tau_t = 0.07$ leads to the best performance ($+0.3$ p.p.). We considered several EMA keep rate parameter values following previous work [He et al., 2020, Wei et al., 2021b, Denize et al., 2023], and found that 0.999 achieves the best results ($+0.1$ p.p.).

Increasing the number of queries In Table 5.7, we provide an ablation on the number of object proposals (queries) N in Def. DETR, when pretraining with ProSeCo and finetuning afterwards. A higher N leads to more parameters in the model and longer computing time, but we can see that the results of Def. DETR are relatively stable w.r.t. to the number of queries. On the other hand, ProSeCo benefits from increasing the number of queries since it means a higher effective batch size during contrastive learning. However, the default value of $N = 300$ leads to the best results both with and without pretraining.

Finetuning with a lot of data In Table 5.8, we present results when finetuning the models on the full COCO dataset (118k training images) under the $1\times$ training schedule [Wei et al., 2021b, Li et al., 2022], *i.e.* 12 training epochs and decaying the learning rate in the last epoch. The improvements in the large-scale annotated data regime are limited, which can be observed also in previous work [Dai et al., 2021b, Bar et al., 2022]. As we can see, after pretraining on IN, our ProSeCo reaches similar results than DETReg [Bar et al., 2022]. We believe that this limitation comes from the pretrained backbone that stays fixed during pretraining, and from the extensive

Pretraining	Dataset	mAP
ProSeCo w/ SwAV	COCO	27.4
ProSeCo w/ SwAV	IN	27.8
ProSeCo w/ SCRL	IN	**28.8**

(a)

Loss	δ	mAP
InfoNCE	1.0	26.4
LocNCE	0.5	**27.0**
SCE	1.0	26.1
LocSCE (Ours)	0.2	27.0
LocSCE (Ours)	0.7	27.1
LocSCE (Ours)	0.5	**27.8**

(b)

Table 5.5. ***(a)*** *Comparison after finetuning when using different pretrained backbone and/or pretraining datasets for ProSeCo.* ***(b)*** *Comparison of the effect of the location information using different IoU threshold δ and for the different constrastive loss considered. All performance (mAP in %) are measured on Mini-COCO 5%.*

supervision during fine-tuning. However, as we can see in Table 5.2b, we outperform DETReg on our *FSOD-train* benchmark, which represents a setting with mid-scale annotated data (42k training images).

5.4.4 Pretraining cost

We compare the cost of pretraining *in terms of memory and hardware used* to SoCo [Wei et al., 2021b] in Table 5.9 since it is the closest in terms of pretraining pipeline. The information are derived from their paper and official github repository.

Even though A100 GPUs are faster than V100 GPUs, we are *training much faster* which is partly explained by the fact that they learn the backbone along with the detection heads during pretraining, leading to more parameters to learn and more computations. Furthermore, our ProSeCo requires a smaller batch size leading to less memory and thus less GPUs needed.

Ablative Variant	Batch size		Images scale		Temperature τ_t		EMA			mAP
	192	**64**	Mid	**Large**	0.05	**0.07**	0.99	0.996	**0.999**	
Base		✓	✓		✓			✓		26.7
Abl. Batch	✓		✓		✓			✓		26.5
Abl. Scale		✓		✓	✓			✓		27.4
Abl. Temp.		✓	✓			✓		✓		27
Abl. EMA		✓	✓		✓		✓			26.3
		✓	✓		✓				✓	26.8
Best		✓		✓		✓			✓	**27.8**

Table 5.6. *Ablation studies on different hyperparameters for the proposed method. The performance (mAP in %) are measured on Mini-COCO 5%.* ***Green and bold columns names*** *indicate a* positive *effect on the performance, and* red columns *a* negative *effect.*

5.5 Conclusion

In this chapter, we aim to use large unlabeled datasets for an unsupervised pretraining of the overall detection model to improve performance when having access to fewer labeled data. In this end, we propose *Proposal Selection Contrast (ProSeCo)*, a novel pretraining approach for Object Detection. Our method leverages the large number of object proposals generated by transformer-based detectors for contrastive learning, reducing the necessity of a large batch size, and introducing the location information of the objects for the selection of positive examples to contrast. We show from various experiments on standard and novel benchmarks in learning with few training data that ProSeCo outperforms previous pretraining methods. Throughout this chapter, we advocate for consistency in the level of information encoded in the features when pretraining. Indeed, learning object-level features during pretraining is more important than image-level when applied to a dense downstream task such as Object Detection.

Unsupervised pretraining allows to reduce the total amount of labels used. It improves the performance when fine-tuning on limited data by providing a better initialization of the weights *before the fine-tuning phase*. In the following chapter, we

Method	N	Performance
Supervised	100	23.1
	200	23.0
	300	**23.6**
	500	23.3
ProSeCo (Ours)	100	25.7
	200	26.5
	300	**27.8**
	500	27.2

Table 5.7. *Performance (mAP in %) comparison on Mini-COCO 5% when changing the number of object proposals in Def. DETR.*

Method	COCO		
	mAP	AP_{50}	AP_{75}
Supervised	37.4	55.5	40.5
DETReg [Bar et al., 2022]	38.9	56.6	42.3
ProSeCo (Ours)	38.9	56.2	42.4

Table 5.8. *Performance (mAP, AP_{50} and AP_{75} in %) comparison on the full COCO dataset with the $1\times$ training schedule.*

are interested in improving the performance *after the fine-tuning phase* through semi-supervision.

Method	IN epochs	Batch Size	Iterations	Time	It. / sec.	Hardware
SoCo [Wei et al., 2021b]	400	2048	240k	140h	0.5	16 V100 32G
ProSeCo (Ours)	10	64	187k	40h	1.4	8 A100 40G

Table 5.9. *Comparison of pretraining cost between overall pretraining methods. We compare the number of pretraining epochs on IN, the total batch size, the total number of iterations, the total training time (in hours), the number of iterations per seconds (It. / sec.), and the hardware used (number and type of GPUs). We can see that our pretraining is globally less costly than SoCo.*

Few Annotation Learning for Semi-Supervised Object Detection

6

Contents

In this chapter, we focus on the post-finetuning stage. We recall the positioning in the general training pipeline in Figure 6.1. We want to improve object detectors based on Transformers and trained on few annotated data by leveraging unlabeled data through semi-supervision. The present chapter is organized as follows. In Section 6.2, we review previous work in the topic of object detection and semi-supervised learning and discuss their relations with our proposed method. Then, in Section 6.3, we introduce our proposed MT-DETR and the ideas it is built upon. In Section 6.4, we present the results on different FAL benchmarks and an ablation study of the effect of the different architecture choices. Finally, we conclude the chapter in Section 6.5.

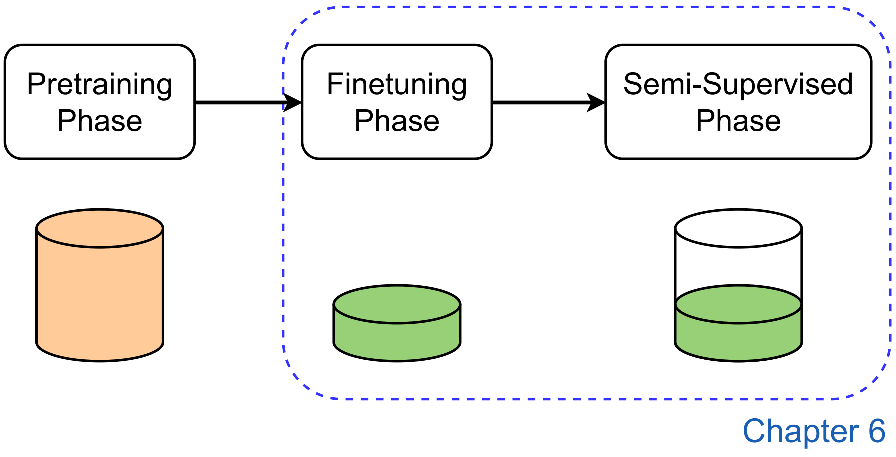


Figure 6.1. *Illustration of the position of our contributions from this chapter in the full training pipeline presented in Chapter 1. We are interested on Semi-Supervised learning for Object Detection.*

6.1 Introduction

In the previous chapter, we studied how unsupervised pre-training can be beneficial for object detection task. In this new chapter, we consider a complementary setting where we can guide the learning using only a handful of labeled examples while simultaneously leveraging a large amount of unlabeled data. This corresponds to a particular case of *semi-supervised learning* (SSL) called *few-annotation learning* (FAL) hereafter.

For the task of *Object Detection* (OD), methods in the literature that tackle this setting [Jeong et al., 2019, Sohn et al., 2020b, Liu et al., 2021, Xu et al., 2021, Zhou et al., 2021, Tang et al., 2021] have all considered object detectors based on traditional convolutional networks [Ren et al., 2015] with a set of specific post-processing heuristics required for them to work [Hosang et al., 2017, Zhang et al., 2020]. More recent object detectors are based on an encoder-decoder architecture using transformers [Vaswani et al., 2017] that allows for end-to-end OD without depending on this hand-crafted pipeline [Carion et al., 2020, Zhu et al., 2021]. However, they have not yet been tested in the SSL context.

The starting point of this contribution is the observation that current state-of-the-art transformer-based architecture [Zhu et al., 2021] performs much better than traditional object detectors in a data-scarce fully supervised learning setting, also called *Few-Shot Learning* (FSL), for an equal number of parameters. However, when plugging it into a state-of-the-art *Semi-Supervised Object Detection* (SSOD) method [Liu et al., 2021], we observe that the model fails to converge, meaning that when used as is (Figure 6.2), applying SSL methods from literature to transformer-based object detectors does not guarantee good results. Thus, we propose a novel SSL method tailored for transformer-based architectures in order to take advantage of the effectiveness of transformers in FSL, and upscale these methods for FAL. Our proposed method achieves state-of-the-art in several FAL benchmarks.

More precisely, our contributions are summarized as:

- After showcasing the strong performance of transformer-based detectors using few labeled data, we propose *Momentum Teaching DETR* (MT-DETR), an approach for SSOD that leverages the specificities of transformer-based architectures and outperforms previous semi-supervised approaches in FAL settings.

- Contrary to convolution-based OD methods, our approach does not rely on heuristics and post-processing for constructing pseudo-labels. Thus, it eliminates sensitive hyperparameters.

Before presenting these contributions in details, we provide an overview of related work close to the setting considered in this chapter.

6.2 Related Work

6.2.1 Fully Supervised Object Detection

We have seen in previous Chapters 2 and 5 that OD combines the tasks of object localization and classification. The most popular OD models have been based on fully convolutional neural networks [Girshick et al., 2014, Ren et al., 2015, Redmon et al., 2016]. These methods can be separated into *two-stage* [Girshick et al., 2014, Girshick, 2015, Ren et al., 2015, Lin et al., 2017a] or *one-stage* [Redmon et al., 2016, Liu et al., 2016, Tian et al., 2019] detectors. The former methods make predictions of boxes and their class labels based on region proposals, *e.g.* from a Region Proposal Network (RPN) [Ren et al., 2015], while the latter make predictions *w.r.t.* to anchors [Lin et al., 2017b] or a grid of possible reference points [Redmon et al., 2016, Zhou et al., 2019, Tian et al., 2019]. Both categories depend heavily on hand-designed heuristics, with the most prominent example being the Non-Maximal Suppression (NMS) post-processing, widely used in state-of-the-art OD methods [Hosang et al., 2017, Bodla et al., 2017]. Similarly to previous Chapter 5, we remain interested in this chapter to the recent transformer-based architectures. We found in our experiments that Def. DETR is a stronger baseline for FSL than the more popular Faster R-CNN [Ren et al., 2015] widely used in previous work, which motivated the focus on the transformer-based architectures.

Method	Params.	COCO				VOC07	
		0.5% (590)	1% (1180)	5% (5900)	10% (11800)	5% (250)	10% (500)
FRCNN + FPN[†]	42M	6.83 ± 0.15	9.05 ± 0.16	18.47 ± 0.22	23.86 ± 0.81	18.47 ± 0.39	25.23 ± 0.22
Def. DETR	40M	$\mathbf{8.95 \pm 0.51}$	$\mathbf{12.96 \pm 0.08}$	$\mathbf{23.59 \pm 0.21}$	$\mathbf{28.55 \pm 0.08}$	$\mathbf{22.87 \pm 0.38}$	$\mathbf{29.03 \pm 0.46}$
Δ		$+2.12$	$+3.91$	$+5.12$	$+4.69$	$+4.40$	$+3.80$

Table 6.1. *Performance (mAP in %) comparison between Faster R-CNN (FRCNN) [Ren et al., 2015] with Feature Pyramid Network (FPN) [Lin et al., 2017a], a two-stage detector commonly used in SSOD methods, and Deformable DETR (Def. DETR) [Zhu et al., 2021], a state-of-the-art transformer-based object detector, with the same ResNet-50 backbone model. The performances are reported for different percentages (and the corresponding number of images) of COCO and VOC07 labeled training data. See Section 6.4.1 for more details on the experiments. Def. DETR performs better than FRCNN + FPN with fewer labeled data for a similar amount of parameters. [†]: Results from [Liu et al., 2021] if available, from our reproduction otherwise.*

6.2.2 Semi-supervised Object Detection for Few-Annotation Learning

The goal of semi-supervised learning is to take advantage of unlabeled data along with labeled data during training. In the more specific case of FAL, it allows reducing the need of a large amount of labeled data by leveraging the use of unlabeled data. The problem of SSL in computer vision was historically tackled first for the image classification task as presented Chapter 3. Our work takes inspiration from recent *hybrid methods* [Sohn et al., 2020a, Chen et al., 2020c] adapted to OD, by training a student model to match the predicted *probability distributions* of proposals made by a teacher model.

Methods in the literature for SSOD are mainly relying on pseudo-labels provided by a teacher model after applying strong data augmentations on unlabeled data [Jeong et al., 2019, Sohn et al., 2020b, Liu et al., 2021, Xu et al., 2021, Zhou et al., 2021, Tang et al., 2021]. The use of geometric transformations in these strong augmentations is particularly important for OD [Sohn et al., 2020b], due to the localization task intrinsic to the problem. The most recent and best performing ones [Liu et al., 2021, Xu et al., 2021, Tang et al., 2021] are also updating the teacher through Exponential Moving Average (EMA) [Lillicrap et al., 2015] of the student's weights to continuously improve the teacher and, thus, the pseudo-labels given to the student. Although the use of

EMA has improved the performance of the models, we propose in our work to stabilize the teacher, by applying an updating strategy throughout training, inspired by recent advances in self-supervised learning [Grill et al., 2020, Caron et al., 2021]. Pseudo-labels are obtained, either by using a *hard labeling* [Jeong et al., 2019, Sohn et al., 2020b, Liu et al., 2021, Xu et al., 2021, Zhou et al., 2021] approach, which consists in applying an $\arg\max$ to the predictions, or a *soft labeling* [Tang et al., 2021] approach, by fully using the predicted distribution. All the previous methods are relying on NMS and thresholding the *confidence scores*, *i.e.* the $\mathrm{softmax}$ of the predictions, given by the teacher model. However, the above-mentioned post-processing steps are sensitive to hyperparameters and introduce a bias into the model incentivizing it to be highly confident in its predictions, which may be suboptimal, particularly when few labeled data are available. Therefore, we aim to remove all these post-processing steps in this work. Furthermore, SSOD methods in the literature have been exclusively built and evaluated using two-stage OD architectures, and we found that they do not work as is for the more recent detection models based on transformers.

In this chapter, we investigate SSOD through the lens of FAL, and we focus our experiments in this setting, in contrast to previous work that address FAL with only a limited number of experiments.

6.3 A semi-supervised learning approach for transformer-based object detection

In this section, we first motivate our main idea to use a recent state-of-the-art transformer-based OD method in an SSL context by providing several results on both FSL and FAL settings. Then, we present Momentum Teaching DETR (MT-DETR), our transformer-based SSOD method more adapted to FAL and illustrated in Figure 6.3. More specifically, we describe the construction of the pseudo-labels for unlabeled data, and the update scheduling for the teacher model.

6.3.1 How do object detectors handle data scarcity ?

From the results presented in Table 6.1, we can see that Deformable DETR (Def. DETR) [Zhu et al., 2021], a recent state-of-the-art detection model based on transformers, achieves consistently better performance than the most popular two-stage method in FSL. We refer the reader to Section 6.4.1 for all the implementation details.

These results motivated us to implement Def. DETR in a state-of-the-art SSOD method to see how it performs in FAL settings. We opted for the recent Unbiased Teacher (UBT) [Liu et al., 2021], as its strong results in FAL were easily reproducible with the provided code. Surprisingly, we observed that with Def. DETR detector, the

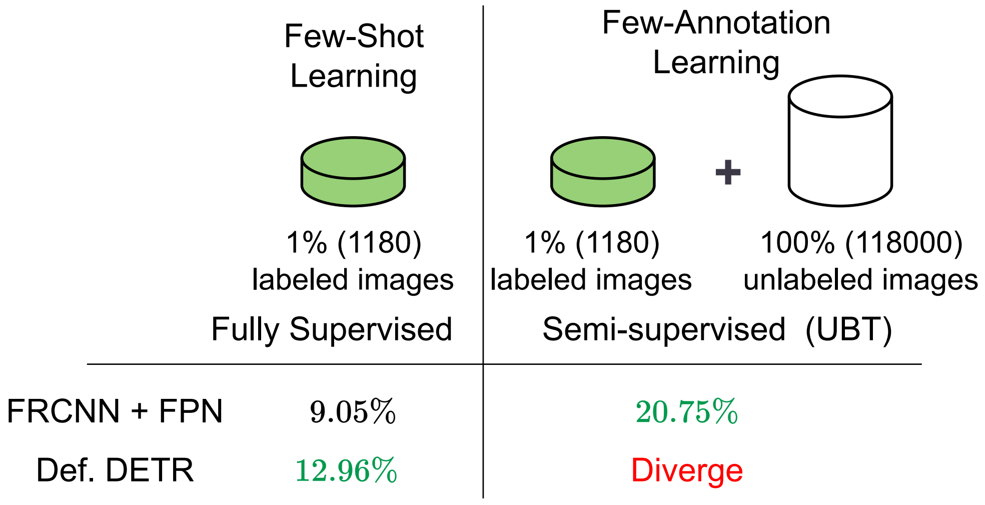


Figure 6.2. *Comparison of mean final performance (mAP in %) between Faster R-CNN (FRCNN) [Ren et al., 2015] with Feature Pyramid Network (FPN) [Lin et al., 2017a] and Deformable DETR (Def. DETR) [Zhu et al., 2021] in the Few-Shot and Few-Annotation Learning settings, using only 1% of labeled data on COCO (about 1180 images). See Section 6.4.1 for experimental details. In the fully supervised case, Def. DETR achieves better results than FRCNN. However, in the semi-supervised case implemented in Unbiased Teacher (UBT) [Liu et al., 2021], Def. DETR cannot converge.*

model does not converge in all the FAL settings tested: 1% of COCO as labeled data (*i.e.* about 1180 labeled images), 5% and 10% of VOC 07 (*i.e.* 250 and 500 labeled images respectively). Even though it passes by an early best (about 17% mAP on 1% of COCO) at the beginning of training, the model collapses soon after. This diverging behavior is not satisfying in practice, even more so that the same method used with a Faster R-CNN [Ren et al., 2015] architecture converges (it achieves about 20% final mAP on 1% of COCO) in similar settings (*c.f.* Figure 6.2). All of this shows that current state-of-the-art SSOD methods are not adapted to more recent transformer-based architectures.

Inspired by these results, we propose an SSL method tailored for transformer-based OD called *Momentum Teaching DETR (MT-DETR)*.

6.3.2 Overview of our approach

As shown in Figure 6.3, our approach is composed of a *student-teacher architecture*, which is common for semi-supervised learning [Tarvainen and Valpola, 2017, Sohn et al., 2020a]. Even though the general structure of the approach bears similarities

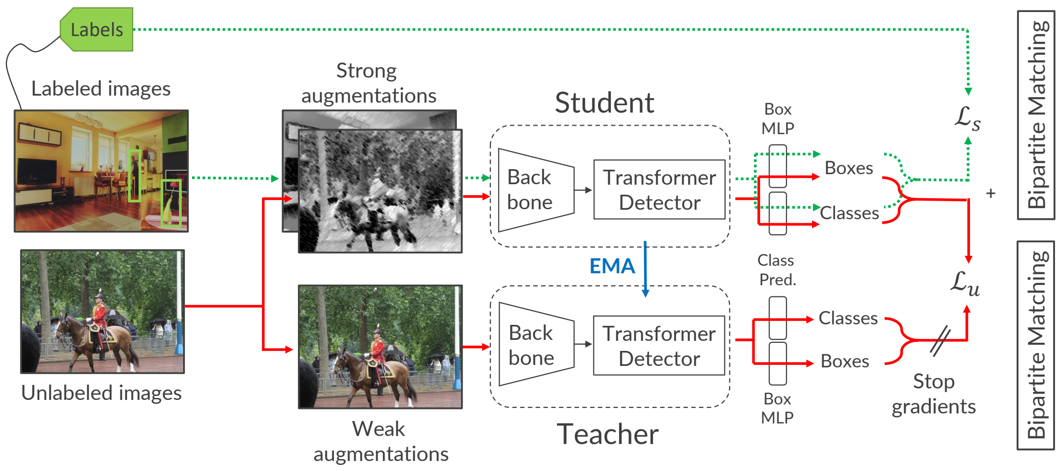


Figure 6.3. *Overview of our Momentum Teaching DETR (MT-DETR) approach for SSOD. The method follows a student-teacher architecture, with the teacher updated through an Exponential Moving Average (EMA) of the student. The keep rate parameter for the EMA follows a* cosine scheduling*. In the supervised branch (*in dotted and green*), the supervised loss $\mathcal{L}_s$ is computed with the predictions of the student on the labeled images. In the unsupervised branch (*in straight and red*), the* raw, i.e. unprocessed*, outputs of the teacher model for the weakly augmented unlabeled images are used as* soft *pseudo-labels without applying any heuristic like NMS or confidence thresholding. After finding the best corresponding detection proposals with bipartite matching, the student model learns from the strongly augmented images to match the distribution of class probabilities and the bounding boxes in these pseudo-labels through the unsupervised loss $\mathcal{L}_u$.*

with our ProSeCo presented in the previous Chapter 5, such as the student-teacher architecture, the strong and weak augmentations, and the dual bipartite matching, there are some important differences coming from the practical data conditions considered. With ProSeCo, the input data being fully unlabeled, both Hungarian algorithms are unsupervised, and the localization task has to be learned partly from region proposals given by an unsupervised algorithm [Uijlings et al., 2013]. This is important to avoid a collapse of the model if the student relies only on the teacher. In the FAL settings that we consider here, the supervision from the labels, even with few of them, stabilizes the training and avoids a collapse of the model. Furthermore, the model has also access to class information from the supervision, such as the total number of class and also reusing classification layers for pseudo-labeling. We describe in this section the full semi-supervised training process.

Both student and teacher models are initialized from a fully supervised model trained on the *few labeled data* available. Then, during the semi-supervised training, the method takes as inputs a batch of labeled images $\mathcal{B}^l = \{(\mathbf{x}_i^l, \mathbf{y}_i^l)\}_{i=1}^{N^l}$ and a batch of unlabeled images $\mathcal{B}^u = \{\mathbf{x}_i^u\}_{i=1}^{N^u}$. We define $\mathbf{x}_i^l$ and $\mathbf{x}_i^u$ as the i^{th} labeled and unlabeled image respectively, $\mathbf{y}_i^l = \{\mathbf{y}_{(i,j)}^l\}_{j=1}^{k_i} = \{(c_{(i,j)}^l, \mathbf{b}_{(i,j)}^l)\}_{j=1}^{k_i} \in \{\{1, 2, \dots, C\} \times \mathbb{R}^4\}_{j=1}^{k_i}$ as the corresponding k_i ground truth class labels and box coordinates, and finally, N^l and N^u are respectively the labeled and unlabeled batch sizes. The student model is updated by a weighted combination of a supervised loss $\mathcal{L}_s$ and an unsupervised loss $\mathcal{L}_u$ with weight $\lambda_u \in \mathbb{R}^+$:

$$\mathcal{L}(\mathcal{B}^l, \mathcal{B}^u) = \frac{1}{N^l}\mathcal{L}_s(\mathcal{B}^l) + \frac{\lambda_u}{N^u}\mathcal{L}_u(\mathcal{B}^u). \tag{6.1}$$

Below, we first describe the *supervised branch*, which computes the *supervised loss* using the batch of labeled data $\mathcal{B}^l$. Then, we detail the *unsupervised branch*, which computes the *unsupervised loss* with the batch of unlabeled data $\mathcal{B}^u$.

Supervised branch To compute the supervised loss, the supervised branch follows the supervised learning of Def. DETR [Zhu et al., 2021], which is an improved version of DETR [Carion et al., 2020]. For each image $\mathbf{x}_i^l$, the student model infers N predictions $\hat{\mathbf{y}}_i^l = \{\hat{\mathbf{y}}_{(i,j)}^l\}_{j=1}^N = \{(\hat{\mathbf{c}}_{(i,j)}^l, \hat{\mathbf{b}}_{(i,j)}^l)\}_{j=1}^N$ of boxes $\hat{\mathbf{b}}_{(i,j)}^l$ and their associated predicted labels *logits* $\hat{\mathbf{c}}_{(i,j)}^l \in \mathbb{R}^{C+1}$, with the $(C+1)^{\text{th}}$ logit representing the *no object* ($\varnothing$) class. Then, the Hungarian algorithm [Munkres, 1957] finds from all the permutations of N elements $\mathfrak{S}_N$, the optimal bipartite matching $\hat{\sigma}_i^l$ between the predictions $\hat{\mathbf{y}}_i^l$ of the student model and the ground truth labels $\mathbf{y}_i^l$: $\hat{\sigma}_i^l = \arg\min_{\sigma \in \mathfrak{S}_N} \sum_j^N \mathcal{L}_{\text{match}}(\mathbf{y}_{(i,j)}^l, \hat{\mathbf{y}}_{(i,\sigma(j))}^l)$. Thus, for each labeled image $\mathbf{x}_i^l$, the j^{th} ground truth $\mathbf{y}_{(i,j)}^l$ is associated to $\hat{\sigma}_i^l(j)$. Similarly to the loss used in object detectors, the matching cost $\mathcal{L}_{\text{match}}$ used in the Hungarian algorithm takes into account both class and bounding box predictions through a linear combination of the *Focal loss* [Lin et al., 2017b] $\mathcal{L}_{\text{focal}}$, the ℓ_1 loss of the box coordinates, and the generalized IoU loss [Rezatofighi et al., 2019] $\mathcal{L}_{\text{giou}}$, respectively. These loss functions are then used to compute the supervised loss $\mathcal{L}_s$ as well:

$$\begin{aligned}\mathcal{L}_{\text{match}}(\mathbf{y}_{(i,j)}^l, \hat{\mathbf{y}}_{(i,\sigma(j))}^l) = \mathbb{1}_{\{\hat{\mathbf{c}}_{(i,\sigma(j))}^l \neq \varnothing\}}\Big[&\lambda_{\text{class}}\mathcal{L}_{\text{focal}}\left(c_{(i,j)}^l, \hat{\mathbf{c}}_{(i,\sigma(j))}^l\right) \\ &+ \lambda_{\ell_1}\|\mathbf{b}_{(i,j)}^l - \hat{\mathbf{b}}_{(i,\sigma(j))}^l\|_1 + \lambda_{\text{giou}}\mathcal{L}_{\text{giou}}\left(\mathbf{b}_{(i,j)}^l, \hat{\mathbf{b}}_{(i,\sigma(j))}^l\right)\Big],\end{aligned} \tag{6.2}$$

$$\begin{aligned}\mathcal{L}_s(\mathcal{B}^l) = \sum_{i=1}^{N^l}\sum_{j=1}^{N}\Big[&\lambda_{\text{class}}\mathcal{L}_{\text{focal}}\left(c_{(i,j)}^l, \hat{\mathbf{c}}_{(i,\hat{\sigma}_i^l(j))}^l\right) \\ &+ \mathbb{1}_{\{\hat{\mathbf{c}}_{(i,\hat{\sigma}_i^l(j))}^l \neq \varnothing\}}\lambda_{\ell_1}\|\mathbf{b}_{(i,j)}^l - \hat{\mathbf{b}}_{(i,\hat{\sigma}_i^l(j))}^l\|_1 \\ &+ \mathbb{1}_{\{\hat{\mathbf{c}}_{(i,\hat{\sigma}_i^l(j))}^l \neq \varnothing\}}\lambda_{\text{giou}}\mathcal{L}_{\text{giou}}\left(\mathbf{b}_{(i,j)}^l, \hat{\mathbf{b}}_{(i,\hat{\sigma}_i^l(j))}^l\right)\Big].\end{aligned} \tag{6.3}$$

In the above equations, we define $\lambda_{\text{class}}, \lambda_{\ell_1}, \lambda_{\text{giou}} \in \mathbb{R}^+$ as the coefficients in the matching cost and $\mathbb{1}_{\mathcal{X}}$ the *indicator function*, such that $\forall x, \mathbb{1}_{\mathcal{X}}(x) = 1$ iff $x \in \mathcal{X}$.

Unsupervised branch Our main contribution is the unsupervised loss for transformer-based OD. In the unsupervised branch, we produce two different views for each unlabeled image $\mathbf{x}_i^u$: a *weakly augmented view* $\mathbf{x}_i^{u\prime}$ and a *strongly augmented view* $\mathbf{x}_i^{u\prime\prime}$[1]. Then, the teacher model provides *soft pseudo-labels* $\mathbf{y}_i^u = \{\mathbf{y}_{(i,j)}^u\}_{j=1}^N = \{(\mathbf{c}_{(i,j)}^u, \mathbf{b}_{(i,j)}^u)\}_{j=1}^N$, with $\mathbf{c}_{(i,j)}^u$ the predicted *logits*, for each *weakly augmented* unlabeled image $\mathbf{x}_i^{u\prime}$, and the student model infers predictions $\hat{\mathbf{y}}_i^u = \{\hat{\mathbf{y}}_{(i,j)}^u\}_{j=1}^N = \{(\hat{\mathbf{c}}_{(i,j)}^u, \hat{\mathbf{b}}_{(i,j)}^u)\}_{j=1}^N$ from the corresponding *strongly augmented* unlabeled view $\mathbf{x}_i^{u\prime\prime}$.

We apply the same Hungarian algorithm with the same matching cost $\mathcal{L}_{\text{match}}$ to obtain the best permutation $\hat{\sigma}_i^u = \arg\min_{\sigma \in \mathfrak{S}_N} \sum_j^N \mathcal{L}_{\text{match}}(\mathbf{y}_{(i,j)}^u, \hat{\mathbf{y}}_{(i,\sigma(j))}^u)$, that matches the predictions of the student with the closest pseudo-label. In the unsupervised loss $\mathcal{L}_u$, we follow the consistency regularization paradigm [Buciluă et al., 2006, Chen et al., 2020c, Caron et al., 2021]. We train the student network to match the probability distributions of the classes predicted by the student with the soft pseudo-labels proposed by the teacher. We learn to match these distributions by minimizing the cross-entropy between the two class distribution outputs normalized by a softmax function. We define respectively:

$$p_{(i,j)}^{s}{}^{(k)} = \text{softmax}(\mathbf{c}_{(i,j)}^u)^{(k)} = \frac{\exp(c_{(i,j)}^u{}^{(k)})}{\sum_{n=1}^{C+1} \exp(c_{(i,j)}^u{}^{(n)})}, \tag{6.4}$$

and $p_{(i,j)}^{t}{}^{(k)} = \text{softmax}(\hat{\mathbf{c}}_{(i,j)}^u)^{(k)}$, the student and teacher class distribution outputs, where $c^{(k)}$ is the k^{th} element of $\mathbf{c}, \forall \mathbf{c} \in \mathbb{R}^{C+1}$. Then the cross-entropy loss is defined as:

$$\mathcal{L}_{\text{CE}}(\mathbf{c}_{(i,j)}^u, \hat{\mathbf{c}}_{(i,j)}^u) = -\sum_{k=1}^{C+1} p_{(i,j)}^{s}{}^{(k)} \log p_{(i,j)}^{t}{}^{(k)}, \tag{6.5}$$

and finally, we compute the unsupervised loss $\mathcal{L}_u$ as:

$$\begin{aligned}\mathcal{L}_u(\mathcal{B}^u) = &\sum_{i=1}^{N^u} \sum_{j=1}^{N} \Big[\lambda_{\text{class}} \mathcal{L}_{\text{CE}} \left(\mathbf{c}_{(i,j)}^u, \hat{\mathbf{c}}_{(i,\hat{\sigma}_i^u(j))}^u \right) \\ &+ \mathbb{1}_{\{\hat{\mathbf{c}}_{(i,\hat{\sigma}_i^u(j))}^u \neq \varnothing\}} \lambda_{\ell_1} \|\mathbf{b}_{(i,j)}^u - \hat{\mathbf{b}}_{(i,\hat{\sigma}_i^u(j))}^u\|_1 \\ &+ \mathbb{1}_{\{\hat{\mathbf{c}}_{(i,\hat{\sigma}_i^u(j))}^u \neq \varnothing\}} \lambda_{\text{giou}} \mathcal{L}_{\text{giou}} \left(\mathbf{b}_{(i,j)}^u, \hat{\mathbf{b}}_{(i,\hat{\sigma}_i^u(j))}^u \right) \Big].\end{aligned} \tag{6.6}$$

For FAL, we have little information from the labeled data. Therefore, the quality of the pseudo-labels and their contained information play an important part in the training.

[1]The weak and strong augmentations are described in Section 6.4.1.

6.3.3 Construction of the pseudo-labels

As mentioned above, the unsupervised loss $\mathcal{L}_u$ takes into account the class predictions through a cross-entropy between the outputs of the student model and the matched outputs of the teacher model. We use the $\mathrm{softmax}$ of the outputs of the teacher model as *soft pseudo-labels* for the cross-entropy, as opposed to *hard pseudo-labels* obtained after taking the $\arg\max$.

Following the DETR philosophy [Carion et al., 2020], we give to the students the *raw soft pseudo-labels* obtained from the teacher, *i.e.* we remove all handmade heuristics to process the teacher outputs, namely, the NMS and confidence thresholding. Both of these post-processing steps are sensitive to hyperparameters and restrict the diversity in the pseudo-labels. By introducing a bias to keep the most confident proposals, they have the unwanted effect of encouraging the models to always be highly confident in their predictions. In the case of FAL, where we have access to only a few labeled examples for each class, the model might not be confident for some classes, leading them to be discarded early by the post-processing. Relying on the model's confidence in certain predictions can be tricky. Using the full distributions makes the model less prone to focus on being highly confident in their predictions, and forces the model to take into account the relations between classes. Furthermore, the Hungarian algorithm used in transformer-based OD methods leverages the diversity of proposals given by the model and benefits from the fact that the model is not overconfident on a single class thanks to the matching loss. Indeed, the bipartite matching can favor proposals with better localizations even if the model is less confident in its class predictions, making the use of raw soft pseudo-labels more suitable for transformer-based detectors.

To obtain strong and insightful pseudo-labels helping the student, the teacher must be updated throughout training. We describe the update process in the following section.

6.3.4 Updating the Teacher model

To avoid a poor supervision from the teacher, its weights θ_t are updated by an Exponential Moving Average (EMA) from the student's weights θ_s using a keep rate $\alpha \in [0, 1]$, which we recall here from Chapter 5:

$$\theta_t \leftarrow \alpha\theta_t + (1 - \alpha)\theta_s. \tag{6.7}$$

For $\alpha = 1$, the teacher is constant and for $\alpha = 0$ its weights are equal to the student's. Therefore, there is a trade-off between a too high and too low keep rate parameter. Inspired by the Self-supervised learning literature [Grill et al., 2020, Caron et al., 2021], we update α following a *cosine scheduling* from α_{start} to α_{end}:

$$\alpha \triangleq \alpha_{\text{end}} - (\alpha_{\text{end}} - \alpha_{\text{start}}) \cdot (\cos(\pi k/K) + 1)/2, \tag{6.8}$$

Augmentations	Probability	Parameters	Supervised branch	Unsupervised branch	
				Weak	Strong
Horizontal Flip	0.5	–	✓	✓	✓
Resize	1.0	short edge ∈ range(480,801,32)	✓	✓	✓
Color Jitter	0.8	(brightness, contrast, saturation, hue) = (0.4, 0.4, 0.4, 0.1)	✓		✓
Grayscale	0.2	–	✓		✓
Gaussian Blur	0.5	$\sigma \in [0.1, 2.0]$	✓		✓
CutOut	0.7	scale $\in [0.05, 0.2]$, ratio $\in [0.3, 3.3]$	✓		✓
	0.5	scale $\in [0.02, 0.2]$, ratio $\in [0.1, 6]$	✓		✓
	0.3	scale $\in [0.02, 0.2]$, ratio $\in [0.05, 8]$	✓		✓
Rotate	0.3	degrees $\in [-30, 30]$			✓
Shear	0.3	shear$_x \in [-30, 30]$, shear$_y \in [-30, 30]$			✓
Rescale + Pad + Translation	0.5	translate$_x \in [0, 0.25]$, translate$_y \in [0, 0.25]$ scale$_x \in [0.25, 0.75]$, scale$_y \in [0.25, 0.75]$			✓

Table 6.2. *The different sets of augmentations used during SSL for each branch. The Horizontal Flip and Resize augmentations follow standard supervised training [Carion et al., 2020, Zhu et al., 2021]. The Color Jitter, Grayscale, Gaussian Blur and CutOut augmentations follow Unbiased Teacher [Liu et al., 2021] training, and the geometric augmentations (Rotate, Shear, Rescale, Pad and Translation) follow Soft Teacher [Xu et al., 2021] training.*

with k the current *epoch* and K the maximum number of *epochs*. This scheduling stabilizes the teacher model, especially in the last training iterations, to make it converge at the end of training.

6.4 Experimental Results

In this section, we present a comparative study of the results of our method to the state-of-the-art on FAL benchmarks, as well as an ablative study on the most relevant parts. Before that, we detail the datasets, the evaluation and training settings used for the different experiments.

Method	Pretrain.	Arch.	FAL-COCO			
			0.5% (590)	1% (1180)	5% (5900)	10% (11800)
Supervised	Sup.	2S	6.83 ± 0.15	9.05 ± 0.16	18.47 ± 0.22	23.86 ± 0.81
STAC [Sohn et al., 2020b]	Sup.	2S	9.78 ± 0.53	13.97 ± 0.35	24.38 ± 0.12	28.64 ± 0.21
Instant-Teaching [Zhou et al., 2021]	Sup.	2S	–	18.05 ± 0.15	26.75 ± 0.05	30.40 ± 0.05
Humble Teacher [Tang et al., 2021]	Sup.	2S	–	16.96 ± 0.38	27.70 ± 0.15	31.61 ± 0.28
Unbiased Teacher [Liu et al., 2021]	Sup.	2S	16.94 ± 0.23	20.75 ± 0.12	28.27 ± 0.11	31.50 ± 0.10
Soft Teacher [Xu et al., 2021]	Sup.	2S	–	20.46 ± 0.39	30.74 ± 0.08	34.04 ± 0.14
Supervised	Sup.	T	8.95 ± 0.51	12.96 ± 0.08	23.59 ± 0.21	28.55 ± 0.08
Supervised	ProSeCo	T	11.37 ± 0.40	17.9 ± 0.08	28.33 ± 0.33	32.60 ± 0.28
MT-DETR *(Ours)*	Sup.	T	$\mathbf{17.84} \pm 0.54$	$\mathbf{22.03} \pm 0.17$	$\mathbf{31.00} \pm 0.11$	$\mathbf{34.52} \pm 0.07$
MT-DETR *(Ours)*	ProSeCo	T	14.33 ± 0.17	21.73 ± 0.12	$\mathbf{32.00} \pm 0.16$	$\mathbf{35.83} \pm 0.17$

Table 6.3. *Performance (mAP in %) of our proposed approach on FAL-COCO, using different percentage of labeled data (with the corresponding number of images reported) and 100% of the dataset as unlabeled data. We also report the* pretrained (Pretrain.) *backbone used, the* architecture (Arch.) *of the underlying object detector which is either* Two-Stage (2S) *or* Transformer-based (T)*.*

6.4.1 Datasets, evaluation and training details

Datasets and evaluation protocol We briefly recall the datasets introduced in Chapter 3 that we used here. To evaluate our proposed method, we use the MS-COCO (COCO) [Lin et al., 2014] and PASCAL VOC (VOC) [Everingham et al., 2010] datasets which are standard for object detection, following the settings of existing works [Jeong et al., 2019, Sohn et al., 2020b, Liu et al., 2021, Xu et al., 2021, Zhou et al., 2021, Tang et al., 2021]. COCO is a dataset with 80 classes, and VOC contains 20 classes. We are specifically interested in two *Few Annotation Learning* (FAL) settings:
On *FAL-COCO*, we randomly sample 0.5, 1, 5 or 10% (respectively about 590, 1180, 5900 and 11800 images) of the training set (*train2017*) used as the labeled set and use the full training set for the unlabeled set (about 118k images). Performance is evaluated on *val2017*.
On *FAL-VOC 07-12*, we restrict the labeled training set (VOC07 *trainval*) to a random sample of 5 or 10% (respectively 250 and 500 labeled images), and use the full VOC12 *trainval* (about 11k images) as unlabeled training set. We introduce this novel setting to evaluate our approach in a FAL setting on VOC. We also compare the results with previous SSOD methods using the full VOC07 *trainval* labeled training set (5k labeled

Method	Arch.	FAL-VOC 07-12		
		5% (250)	10% (500)	100% (5000)
STAC [Sohn et al., 2020b]	2S	–	–	44.64
Instant-Teaching [Zhou et al., 2021]	2S	–	–	50.00
Humble Teacher [Tang et al., 2021]	2S	–	–	53.04
Unbiased Teacher [github]	2S	–	–	54.48
Unbiased Teacher*	2S	35.98 ± 0.71	40.34 ± 0.95	54.61
MT-DETR *(Ours)*	T	$\mathbf{36.95} \pm 0.53$	$\mathbf{43.15} \pm 1.10$	**56.2**

Table 6.4. *Performance (mAP in %) of our proposed approach on VOC with fully labeled VOC07 and unlabeled VOC12 to compare with previous work, and in the novel FAL-VOC 07-12 settings. Different percentage of VOC07 are used as labeled data (5%, 10% or 100%, with the corresponding number of images reported), and the full VOC12 dataset is used as unlabeled data. We also report the* architecture (Arch.) *of the underlying object detector which is either* Two-Stage (2S) *or* Transformer-based (T)*. * indicates our implementation of Unbiased Teacher [Liu et al., 2021] in this novel setting to compare with our approach. [github] : updated results after publication [Liu et al., 2021] taken from their official code released.*[3]

images) and VOC12 *trainval* as unlabeled training set. Performance is evaluated on VOC07 *test* set.

In all settings, performance is reported and compared using the $AP_{50:95}$ (mAP, in %) evaluation metric using the official COCO and VOC evaluation codes, respectively.

Training For a fair comparison, a fully supervised ResNet-50 [He et al., 2016] pretrained on ImageNet [Russakovsky et al., 2015] is used as a backbone for all the methods. For fine-tuning Def. DETR [Zhu et al., 2021] on the few labeled data, we train the model with a batch size of 32 images on 8 GPUs until the validation performance stops increasing, *i.e.* for COCO, up to 2000 epochs for 1%, 500 epochs for 5%, 400 epochs for 10%, and for VOC, up to 2000 epochs for both 5% and 10%. For semi-supervised learning, we train MT-DETR for 50 (respectively, 250) epochs of the unlabeled data on COCO (respectively, VOC) with a batch size of 48 labeled images and 48 unlabeled images (respectively, 24 and 24) on 8 GPUs. All experiments

with less than 100% of labeled data are reproduced on 3 different random subsets[2]. The training hyperparameters, are defined as in Def. DETR [Zhu et al., 2021]. The coefficients for the losses are set as $\lambda_{\text{class}} = 2, \lambda_{\ell_1} = 5, \lambda_{\text{giou}} = 2$, and $\lambda_u = 4$. Following the training schedule of Def. DETR, we always decay the learning rates by a factor of 0.1 after about 80% of training. The keep rate parameter α follows a *cosine scheduling* from $\alpha_{\text{start}} = 0.9996$ to $\alpha_{\text{end}} = 1$, with the value of α_{start} chosen according to previous work [Liu et al., 2021].
When using Unbiased Teacher [Liu et al., 2021], we follow the official implementation[3] and the hyperparameters provided.

Augmentations For strong and weak data augmentations, we follow the common data augmentations used in previous works [Sohn et al., 2020b, Liu et al., 2021, Xu et al., 2021]. We apply a random resizing and random horizontal flip for weak augmentations. We randomly add color jittering, grayscale, Gaussian blur, CutOut patches for strong augmentations and also randomly add rescaling, translation with padding, shearing and rotating as geometric transformations [Sohn et al., 2020b] in strong augmentations. In the supervised branch, images are also randomly augmented using weak and strong augmentations without any geometric transformations following Soft Teacher [Xu et al., 2021] practices. It helps the student model to be augmentation-agnostic, to better predict pseudo-labels coming from non-augmented images in the unsupervised branch. We remove the CutOut augmentation in the supervised branch in the most difficult settings of FAL-COCO 0.5% and 1%, since it can cover the only labeled small boxes available and is counterproductive. All the parameters for the different augmentations can be found in Table 6.2.

6.4.2 Results of FAL on COCO and VOC

Tables 6.3 and 6.4 present the results (mAP in %) obtained by our method compared to previous methods in the literature on the FAL-COCO and FAL-VOC 07-12 benchmarks. As can be seen in both tables, our approach is the only one to consider a transformer-based OD architecture (Def. DETR), as opposed to the commonly used two-stage architecture (FRCNN + FPN). When we implemented Def. DETR into Unbiased Teacher [Liu et al., 2021] (UBT), we found that the model cannot converge in FAL settings (*c.f.* Figure 6.2).

First, we can see from both tables that our method always improve performance over the corresponding fully supervised FSL baseline (*c.f.* Table 6.1). With our method, we outperform state-of-the-art results on all labeled fractions of the dataset, and obtain even more strong results specifically when the annotations are scarce: globally about +1 performance point (p.p.) when using 1k or less labeled images, which is even more

[2] https://github.com/CEA-LIST/MT-DETR
[3] https://github.com/facebookresearch/unbiased-teacher

Ablative Variant	EMA Scheduling		Initialization		NMS	Confidence Thresholding				mAP (in %)
	Cosine	Constant	**After FT**	From scratch		**ø**	0.5	0.7	0.9	
Best	✓		✓			✓				**22.25**
Abl. Sched.		✓	✓			✓				21.48
Abl. Init.	✓			✓		✓				16.51
Abl. NMS	✓		✓		✓	✓				19.85
Abl. Thresh.	✓		✓				✓			10.26
	✓		✓					✓		17.34
	✓		✓						✓	12.37

Table 6.5. *Ablation studies of the different parts of our method.* ***Green and bold columns names*** *indicate a* positive *effect on the performance and red columns a* negative *effect. The use of* cosine scheduling, an initialization after fine-tuning (FT) *and* raw soft pseudo labels *corresponds to the best combination found.*

significant when the overall performance is low. For FAL-COCO with 1% of labeled images, our method achieves a mean of 22.03 mAP, which is about 1.2 p.p., or 6% of improvement over the state-of-the-art, UBT. Notably, on FAL-VOC with 10% of labeled images, we obtain mean performance of 43.15 mAP, corresponding to 2.81 p.p. or 7% of improvement over UBT. We note that our method also outperform the state-of-the-art when using more labeled data, such as with the 100% labeled VOC07 setting, where we improve of about 1.5 p.p. over UBT.

6.4.3 Ablation studies

In Table 6.5, we present an ablation study on the main parts of our approach. We review each ablation below.

EMA scheduling The effect of the EMA scheduling is compared between the *Best* and *Abl. Sched.* rows. We can see that using a *cosine scheduling* to gradually reduce the EMA keep rate parameter α leads to an improvement of about 0.7 p.p., as opposed to using a *constant* value for α as done in other SSL approaches [Liu et al., 2021, Xu et al., 2021, Tang et al., 2021].

Initialization In this ablation, we study the effect of *end-to-end semi-supervised learning* [Xu et al., 2021] in the row *Abl. Init.* which consists in starting the semi-supervised training *from scratch* compared to an initialization *after Fine-Tuning (FT)* in the row *Best*, in which we initialize both student and teacher models from the weights of the fine-tuned model on the few labeled data. As can be seen in Table 6.5 and contrary to Soft Teacher [Xu et al., 2021], starting the semi-supervised training from fine-tuned weights is much more effective (about 5.7 p.p. better) than starting from randomly initialized weights, since the teacher model will give useful pseudo-labels to the student from the start of training.

NMS The importance of removing NMS to avoid filtering interesting pseudo-labels and introducing bias is showcased between the rows *Best* and *Abl. NMS*. We can see that, contrary to the common practice when using other detectors [Liu et al., 2021, Xu et al., 2021, Tang et al., 2021], the introduction of NMS leads to a performance drop of about 2.5 p.p. This is why we used *raw pseudo labels*, *i.e.* without any post-processing.

Confidence Thresholding The effect of introducing a threshold to filter out the pseudo-labels given by the teacher with poor confidence is shown in the rows *Best* and *Abl. Thresh.*. We test the results using several common values in the literature (0.5, 0.7 and 0.9) [Sohn et al., 2020a, Liu et al., 2021, Xu et al., 2021]. A value of 0.7 seems to give the best final results (17.34 mAP) between the thresholding variants, but we can see that choosing the best threshold to apply is extremely sensitive. Similarly to Humble Teacher [Tang et al., 2021], we also found that removing the confidence threshold to use all the *soft pseudo-labels*, which corresponds to the column with $\emptyset$, leads to stronger results (22.24 mAP), less sensitivity and fewer hyperparameters.

Data augmentation In this ablation study, we are interested in the effect of the different sets of data augmentation to generate the *strong view* in the unsupervised branch. In Table 6.6a, we group similar augmentations together and then study the effects of each group in Table 6.6b. First, we can see that the model does not converge when using the same setting as Unbiased Teacher [Liu et al., 2021], confirming our observation in Figure 6.2, but achieves strong result when using *raw soft pseudo-labels*. Furthermore, using more augmentations to generate the *strong view* leads to better results.

6.4.4 Self-supervised Backbone

In Table 6.3, we also report results when initializing with a *Self-supervised* pretraining instead of a *supervised* one before the fine-tuning phase. As is common practice in Object Detection, detectors are usually initialized with the weights of the *backbone* obtained after a supervised training phase on ImageNet [Ren et al., 2015, He et al., 2019]. In these experiments, we use instead weights from our fully unsupervised *ProSeCo* pretraining on ImageNet, presented in Chapter 5. This reduces even more the total amount of labels used in the overall training.

We can see first that initializing with ProSeCo improves the fine-tuning performance for all labeled fractions of COCO considered, confirming the results from Chapter 5. Then, with semi-supervision with MT-DETR, we observe that the unsupervised pre-training improves over the supervised one on the 5% and 10% fractions, but is less effective on the scarcer 0.5% and 1% fractions. This might be due to an overlapping between the classes contained in both datasets. The model benefits from having seen more examples of some class of object during the supervised pretraining for the most label-scarce benchmarks, since they contain very few examples for each class.

6.5 Conclusion

In this work, we experimented in different data scarce settings with the state-of-the-art transformer-based object detector Def. DETR [Zhu et al., 2021] and showed that it performs much better than the most popular two-stage detector Faster R-CNN [Ren et al., 2015] with FPN [Lin et al., 2017a]. Surprisingly, we found that Unbiased Teacher [Liu et al., 2021], a state-of-the-art SSOD method, did not converge when applied with Def. DETR.

To address this issue, we propose Momentum Teaching DETR (MT-DETR), an SSL approach tailored for OD based on transformers, in order to leverage their good results with few labeled data. Our method is based on a student-teacher architecture and, contrary to common practice, discards all previously used handcrafted heuristics to process pseudo-labels generated by the teacher. These processing steps are sensitive to hyperparameters, and introduce biases with the unwanted effect of forcing the models to be overconfident in their predictions. We show that our proposed MT-DETR outperforms state-of-the-art methods, especially in FAL settings. Then, we combined our unsupervised pretraining with our semi-supervised method to reduce even more the number of labels used.

This concludes our contributions in this thesis. In the last chapter we give a summary of the different contributions presented and outline future perspectives for learning with limited labels.

Name	Augmentations
Basic	Horizontal Flip
	Resize
Photo.	Color Jitter
	Grayscale
	Gaussian Blur
CutOut	CutOut
Geom.	Rotate
	Shear
	Rescale + Pad

(a)

Augmentations used	mAP (in %)
Basic + Photo.	17.8
Basic + Photo. + CutOut	
| w/ NMS + Hard PL [Liu et al., 2021]	Div.
| w/o NMS + Soft PL *(Ours)*	21.1
Basic + Photo. + CutOut + Geom.	**21.6**
Basic + Photo. + CutOut + Geom. + Augmentations in Supervised branch	**22.3**

(b)

Table 6.6. *(a) The different sets of augmentations considered during SSL. The* Basic *augmentations follow standard supervised training [Carion et al., 2020, Zhu et al., 2021]. The* Photometric *augmentations are used in Self-supervised learning and in Unbiased Teacher [Liu et al., 2021] along with* CutOut *augmentations. The* Geometric *augmentations follow Soft Teacher [Xu et al., 2021] training.*
*(b) Performance comparison of different combination of data augmentations on FAL-COCO 1%. Using all data augmentations leads to the best results (**in bold**). Furthermore, we can see that the model do not converge when using NMS and confidence thresholding along with Basic, Photometric and CutOut augmentations, which corresponds to the same setting as Unbiased Teacher [Liu et al., 2021] (in red). Using* raw soft pseudo-labels (PL) *stabilizes the training and makes it converge to a strong solution (in green).*

Conclusions and Perspectives 7

Contents

7.1 Summary of the contributions

Although Machine Learning (ML) and Deep Learning (DL) systems are becoming increasingly used in research and industry, they require a huge amount of labeled data to be effective. Major breakthroughs have been possible thanks to large-scale labeled datasets such as ImageNet [Russakovsky et al., 2015], containing 14M labeled images divided into 21k classes, or MS COCO [Lin et al., 2014], with 2.5M labeled instances of 91 classes in 328k images. While images in these datasets represent common objects in natural scenes and considerable performance improvements have been achieved by iterating on them, they can be really semantically far from more specialized tasks which would require specific annotations. Annotating such large-scale datasets are incredibly time-consuming and costly, which represents a high entry cost to apply DL algorithms in real-world settings where there may not be a large amount of labeled data available. This explains why in this thesis, we are working towards *Few-Annotation Learning* (FAL), *i.e.* learning efficient models with limited labels while having access to unlabeled data, such that they can still perform well on new data, even if they have only seen a few *labeled* examples of that data during training. Our contributions are organized in three parts, each detailed in the previous chapters that we summarize below.

In Chapter 4, we investigate the *Meta-Learning* paradigm that is increasingly popular for *Few-Shot Learning* (FSL). We are specifically interested in theoretical justifications of the good empirical results, observed through the lens of *Multi-task Representation Learning*. We provide proofs and a better understanding of the behavior of meta-learning algorithms as well as practical ways to force them to follow the best learning settings. To do so, we propose spectral-based regularization terms and normalization schemes to enforce important theoretical assumptions. This leads to more efficient meta-learning as supported by better empirical results.

Then, in Chapter 5, we develop a novel approach for pretraining Transformer-based Object Detectors, Proposal Selection Contrast (ProSeCo), such that every part of the detector can be properly initialized. The pretraining is based on a Proposal-Contrastive Learning framework. Object proposals given by a teacher model are contrasted with predictions of a student model. To introduce the location information that is important in Object Detection, we compute the *Intersection over Union* between proposals and consider the overlapping ones as positives during contrastive learning. This allows for more sample-efficient learning, as pretrained detectors achieve better performance for fewer annotated samples.

Finally, in Chapter 6, we focus on Semi-Supervised Learning for Object Detection to leverage large-scale unlabeled data along with few annotated samples. We propose the first method specifically designed for Transformer-based detectors, since training with previous approaches does not converge when using few annotated data. We observe that previous semi-supervised methods do not converge in FAL when applied with these detectors. To fix this converging issue, we investigate into different parts of the model and improve methodological aspects such as data augmentation, soft pseudo-labels without counterproductive post-processing. When comparing performance on different few annotation learning benchmarks, our proposed method outperforms state of the art with a more significant gap when labels are scarce.

7.2 Perspectives and discussions

As mentioned in Chapter 3, research in learning with limited labels can be tackled in various ways. There is still a large room for improvements, and open problems for future works. We discuss potential research directions in this section.

7.2.1 Meta-Learning

While the work presented in Chapter 4 proposes an initial approach to bridging the gap between theory and practice for Meta-Learning and Multi-Task Representation Learning, some questions remain open on the inner workings of these algorithms. In particular, being able to take better advantage of the particularities of the training tasks during meta-training could help improve the effectiveness of these approaches. Furthermore, the similarity between source and test tasks was not taken into account in this work, which is an additional assumption in the theory developed in [Du et al., 2020]. Even though we provide a preliminary experimental study using different datasets between meta-training and meta-testing to foster future work on this topic, the theoretical learning bounds have to be updated to take into account the difference in the assumptions.

However, in light of recent works challenging the effectiveness of meta-learning compared to fine-tuning approaches [Chen et al., 2019, Tian et al., 2020c, Wang et al., 2020a], which require less computation, the benefit of meta-learning and episodic training for few-shot learning remains to be proven. While meta-learning can be interesting for specific problems such as Reinforcement Learning [Finn et al., 2017] or Continual Learning [Gupta et al., 2020], it is still unsure if it is more advantageous for Few-Shot Learning.

7.2.2 Unsupervised pretraining

Future work for pretraining transformer-based object detectors, as presented in Chapter 5, could update the backbone during pretraining to further improve the consistency between the backbone and the detection heads. Even though it would likely be more costly to do so, it might help to improve performance in the large-scale annotated data settings. However, the benefits of pretrained backbone for downstream tasks are also contested. While they are common practice in object detection [Ren et al., 2015] and allows achieving good performance with lower supervised training iterations [He et al., 2019], similar performance can still be achieved with random initialization and more supervised training time [He et al., 2019, Newell and Deng, 2020]. This implies that pretrained backbone do not lead to intrinsically better representations. Moreover, the best backbones are currently selected according to ImageNet classification performance, which might not be suited for all downstream problems [Ericsson et al., 2021].

Even though unsupervised pretraining does not require labeled data, it still needs large-scale datasets to be effective. Pretraining on ImageNet currently remains better than pretraining on the target dataset, thanks to the huge amount of diverse data available in ImageNet [Wei et al., 2021b]. However, pretraining on different datasets one after the other can help for target tasks with smaller-scale data [Reed et al., 2022].

7.2.3 Semi-Supervised Object Detection

In SSOD, future works could push the data scarcity in OD even further to consider very few labeled examples for each class, and better understand how to match the performance of SSL methods for image classification in this setting [Sohn et al., 2020a].

Better taking into account the uncertainty in the localization part might be an interesting step forward [Liu et al., 2022b]. Regarding transformer-based detectors, we found that improvements from semi-supervision are less significant than with convolutional methods, which would require more understanding. For instance in Table 6.3, we achieve an improvement of about 9 p.p. with our MT-DETR compared to the supervised baseline in the FAL-COCO 1% setting, whereas the best convolutional methods achieve 11 p.p. of improvements. Furthermore, we also found in Table 6.3 that the fewer the annotated data, the fewer the benefits of unsupervised pretraining in semi-supervision with transformer-based detectors. More experiments are needed to fully understand how to leverage unsupervised pretraining for semi-supervised learning in FAL, as this can reduce the total amount of label used.

7.3 Broader Impacts

We propose to conclude this thesis by discussing the potential benefits and implications of research towards Few-Annotation Learning. Three themes are thus addressed in this section covering the accessibility of Deep Learning, the impact on computational costs and on environmental footprints which represent major issues nowadays.

7.3.1 Accessibility of Deep Learning

Improving the ability of DL methods to learn with fewer labeled data would make DL accessible to a wider audience, since it can be currently a technical barrier for applying DL methods to specialized problems such as medical or geological imaging. These tasks might not have enough data to begin with, since the gathering of data itself can be a challenge because of the cost of the procedures. Furthermore, annotating these data requires expertise and thus, qualified workers, which is expensive and time-consuming. These practical problems have motivated research in learning with limited data and labels, but the means found to alleviate the necessity of labels comes with an additional computational cost which can also restrict accessibility of such techniques.

7.3.2 Computational costs

Unfortunately, few annotation learning does not imply few computations, and it's often the opposite. To compensate for the scarcity in labels, one must rely on costly unsupervised trainings, along or before supervised training. Compared to supervised pretraining, unsupervised pretraining require bigger batch size and longer training time [Chen et al., 2020c]. Whereas supervised training on the fully labeled ImageNet is done with a batch size of 256 and 90 epochs [Akiba et al., 2017], recent self-supervised methods train for about 800 epochs with a batch size of 4096 [Chen et al., 2020c]. While self-supervised backbones achieve the best performance faster in terms of training time on the target dataset [He et al., 2019, Newell and Deng, 2020] compared to supervised backbones, the full training might not be globally faster if we also take into account the time of the pretraining phase. Thus, using off-the-shelf weights pretrained on large-scale common datasets can save a lot of training time, but in general, pretraining on novel datasets might not be interesting in practice. The information contained in the labels makes supervised training very efficient.

Even in few-shot learning, *i.e.* without access to large-scale unlabeled data, the popular Meta-learning framework, and in particular *episodic training*, requires to split the training dataset into small, independent, but overlapping tasks and to train on a large amount of such tasks [Finn et al., 2017]. Recent methods [Ye et al., 2020, Ye and Chao, 2021] even initialize the episodic training phase with fully supervised weights, making the full meta-learning training much more computationally expensive than simpler fine-tuning approaches.

These computational perspectives have to be taken into account when considering FSL and FAL methods. However, with a high computational cost comes an environmental cost, but so does the process of annotating a dataset.

7.3.3 Environmental footprints

Looking through the lens of CO2 emissions, even though unsupervised trainings are obviously expensive because of hardware resources and computing time, the process of annotating a large-scale dataset is not green either. We generally only measure the cost of the training phase [Luccioni et al., 2022], but rarely look at the annotation part. All numerical details from the following discussion can be found in Appendix A.5.

Let us consider the ILSVRC-2012 [Russakovsky et al., 2015] dataset from ImageNet, having 1.2M labeled images and widely used for supervised pretraining of backbones. The annotation tasks were given to Amazon Mechanical Turk (AMT) to parallelize the process, using about 5 independent annotators to verify each image [Russakovsky et al., 2015] which results in about *2000 hours of annotation*. From the carbon intensity of the resulting electrical consumption, detailed in Appendix A.5, we obtain an estimate of the total carbon footprint of this annotation of *286.4 kgCO2eq*.

To put this footprint in perspective, it is equivalent to about 55 days of compute of a node with 8 GPUs A100 used at 100% in a cluster based in France, or 10 days of compute if the cluster is based in the USA (*c.f.* Appendix A.5 for more details). Considering that a full self-supervised pretraining of a regular sized CNN (*e.g.* a ResNet-50) on the same dataset takes about 2 days with the same computational budget, this might represent a lot of possible unsupervised experiments depending on the country.

While these numbers might seem low, we considered here an image classification dataset. The annotation process is simple and can be done quickly since there is a single label per image. For more complicated annotations like in Object Detection tasks, each image can contain multiple instance of objects, and each one of them have to be segmented to provide precise location information. If we consider the COCO dataset [Lin et al., 2014], the full annotation process took about *85 000 hours*. This represents a total carbon footprint of about *12 tons of CO2eq*, or about *12 transatlantic round trips by plane* (from France to New-York), which is on another scale than ImageNet. This footprint is equivalent to *51 years* of computation of a node with 8 A100 GPUs running at 100% of their capacity located in France, and about *9 years* if the node is located in the USA. Once again, the numerical details can be found in Appendix A.5.

One might argue that the annotation process of these two popular large-scale datasets has since then been largely profitable both to the community and from an ecological point of view, considering the quantity of experiments using either these labels or pretrained weights resulting from them. Nevertheless, this might not be the case for *private* or single-use datasets, on which a small-scale annotation process along with Few-Annotation Learning methods would be more beneficial, both ecologically and economically. In the end, the amount of data to annotate must take into account the annotation cost and time, the complexity of the target task, the distance of the data to other already available datasets, and the number of experiments planned both in supervised and unsupervised settings.

Appendix A

Contents

The Appendix is organized as follows. Appendix A.1 provides an additional review on Multi-task Representation Learning theory. Appendix A.2 details two more advanced meta-learning algorithms used in experiments. Appendix A.3 provides the full proofs of the theoretical results presented in Chapter 4. Appendix A.4 gives the detailed performance results of experiments from Section 4.4. Appendix A.5 details the computation of carbon footprint resulting from labelization of popular dataset used in the literature.

A.1 Review of Multi-task Representation Learning Theory

We formulate the main results of the three main theoretical analyses of Multi-task Representation (MTR) Learning Theory provided in [Maurer et al., 2016, Du et al., 2020, Tripuraneni et al., 2020] in Table A.1 to give additional details for Sections 4.2.2 and 4.4.5.

Paper	Assumptions	Φ	Bound
[Maurer et al., 2016]	**A1**. $\forall t \in [[T+1]],\ \mu_t \sim \eta$	–	$O\left(\frac{1}{\sqrt{n_1}} + \frac{1}{\sqrt{T}}\right)$
[Du et al., 2020]	**A2.0**. $\forall t,\ \|\mathbf{w}_t^*\| = \Theta(1)$		
	A2.1. $\forall t,\ \bar{\mathbf{x}}$ is ρ^2-subgaussian		
	A2.2. $\forall t \in [[T]], \exists c > 0 : \Sigma_t \succeq c\Sigma_{T+1}$	**A2.1-2.4**, linear, $k \ll d$	$O\left(\frac{kd}{cn_1T} + \frac{k}{n_2}\right)$
	A2.3. $\frac{\sigma_1(\mathbf{W}^*)}{\sigma_k(\mathbf{W}^*)} = O(1)$	**A2.3-2.5**, general, $k \ll d$	$O\left(\frac{\mathcal{C}(\Phi)}{n_1T} + \frac{k}{n_2}\right)$
	A2.4. $\mathbf{w}_{T+1}^* \sim \mu_{\mathbf{w}} : \|\| \mathbb{E}_{\mathbf{w}\sim\mu_{\mathbf{w}}}[\mathbf{w}\mathbf{w}^T]\|\| \leq O(\frac{1}{k})$	**A2.1,2.5,2.6**, linear + ℓ_2 regul., $k \gg d$	$\sigma\bar{R}\tilde{O}\left(\frac{\sqrt{\mathrm{Tr}(\Sigma)}}{\sqrt{n_1T}} + \frac{\sqrt{\|\|\Sigma\|\|_2}}{\sqrt{n_2}}\right)$
	A2.5. $\forall t,\ p_t = p, \Sigma_t = \Sigma$	**A2.1,2.5,2.6,2.7**, two-layer NN (ReLUs+ ℓ_2 regul.)	$\sigma\bar{R}\tilde{O}\left(\frac{\sqrt{\mathrm{Tr}(\Sigma)}}{\sqrt{n_1T}} + \frac{\sqrt{\|\|\Sigma\|\|_2}}{\sqrt{n_2}}\right)$
	A2.6. Point-wise+unif. cov. convergence		
	A2.7. Teacher network		
[Tripuraneni et al., 2020]	**A3.1**. $\forall t,\ \mathbf{x} \sim \mu_{\mathcal{X}_t}$ is ρ^2-subgaussian		
	A3.2. $\frac{\sigma_1(\mathbf{W}^*)}{\sigma_k(\mathbf{W}^*)} = O(1)$ and $\forall t, \|\mathbf{w}_t\| = \Theta(1)$	**A1-4**, linear, $k \ll d$	$\tilde{O}\left(\frac{kd}{n_1T} + \frac{k}{n_2}\right)$
	A3.3. $\widehat{\mathbf{W}}$ learned using the Method of Moments		
	A3.4. $\mathbf{w}_{T+1}^*$ is learned using Linear Regression		

Table A.1. *Overview of main theoretical contributions related to MTR learning with their assumptions, considered classes of representations and the obtained bounds on the excess risk. Here $\tilde{O}(\cdot)$ hides logarithmic factors.*

One may note that all the assumptions presented in this table can be roughly categorized into two groups. First one consists of the assumptions related to the data generating process (A1, A2.1, A2.4-7 and A3.1), technical assumptions required for the manipulated empirical quantities to be well-defined (A2.6) and assumptions specifying the learning setting (A3.3-4). We put them together as they are not directly linked to the quantities that we optimize over in order to solve the meta-learning problem. The second group of assumptions includes A2.2 and A3.2: both defined as a measure of diversity between source tasks' predictors that are expected to cover all the directions of $\mathbb{R}^k$ evenly. These assumptions are of primary interest as it involves the matrix of predictors optimized in Equation (4.1) as thus one can attempt to force it in order for $\widehat{\mathbf{W}}$ to have the desired properties.

Finally, we note that assumption A2.2 related to the covariance dominance can be seen as being at the intersection between the two groups. On the one hand, this assumption is related to the population covariance and thus is related to the data generating process that is supposed to be fixed. On the other hand, we can think about a pre-processing step that precedes the meta-train step of the algorithm and transforms the source and target tasks' data so that their sample covariance matrices satisfy A2.2. While presenting a potentially interesting research direction, it is not clear how this can be done in practice especially under a constraint of the largest value of c required to minimize the bound. [Du et al., 2020] circumvent this problem by adding A2.5, stating that the task data marginal distributions are similar.

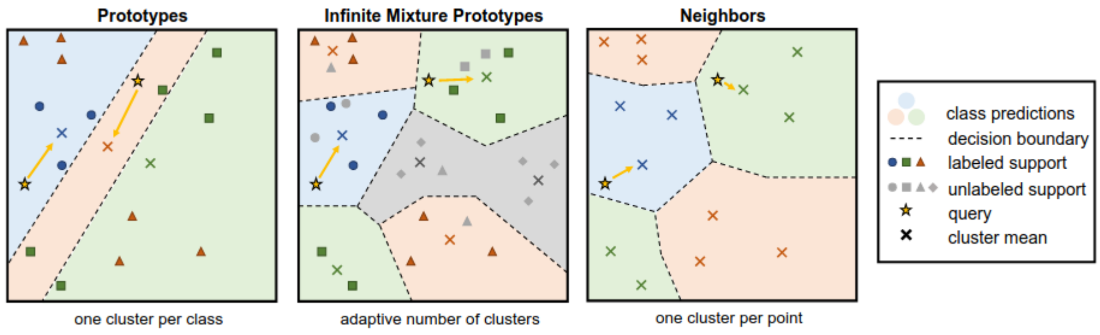
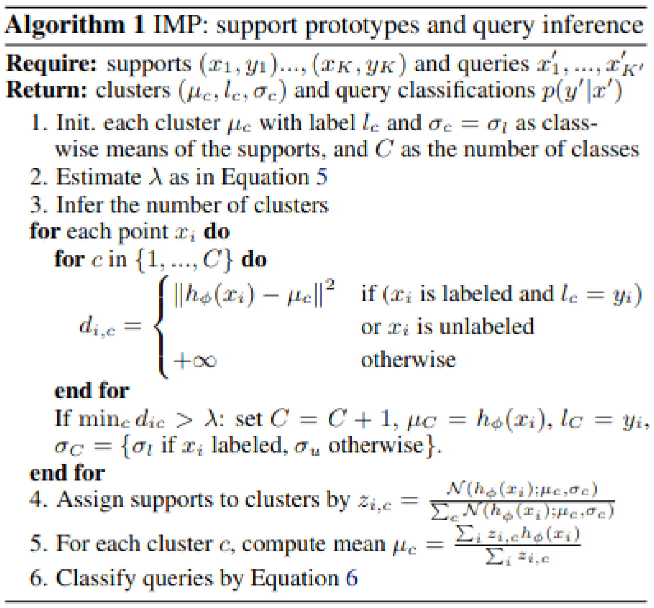


Figure A.1. *Illustration of the IMP methods [Allen et al., 2019]. IMP represents each class by a set of clusters, and infers the number of clusters from the data to adjust its modeling capacity.*



```
Algorithm 1 IMP: support prototypes and query inference
Require: supports $(x_1, y_1), ..., (x_K, y_K)$ and queries $x'_1, ..., x'_{K'}$
Return: clusters $(\mu_c, l_c, \sigma_c)$ and query classifications $p(y'|x')$
  1. Init. each cluster $\mu_c$ with label $l_c$ and $\sigma_c = \sigma_l$ as class-
     wise means of the supports, and $C$ as the number of classes
  2. Estimate $\lambda$ as in Equation 5
  3. Infer the number of clusters
  for each point $x_i$ do
    for $c$ in $\{1, ..., C\}$ do
      $d_{i,c} = \begin{cases} \|h_\phi(x_i) - \mu_c\|^2 & \text{if } (x_i \text{ is labeled and } l_c = y_i) \\ & \text{or } x_i \text{ is unlabeled} \\ +\infty & \text{otherwise} \end{cases}$
    end for
    If $\min_c d_{ic} > \lambda$: set $C = C + 1$, $\mu_C = h_\phi(x_i)$, $l_C = y_i$,
    $\sigma_C = \{\sigma_l$ if $x_i$ labeled, $\sigma_u$ otherwise$\}$.
  end for
  4. Assign supports to clusters by $z_{i,c} = \frac{\mathcal{N}(h_\phi(x_i); \mu_c, \sigma_c)}{\sum_c \mathcal{N}(h_\phi(x_i); \mu_c, \sigma_c)}$
  5. For each cluster $c$, compute mean $\mu_c = \frac{\sum_i z_{i,c} h_\phi(x_i)}{\sum_i z_{i,c}}$
  6. Classify queries by Equation 6
```

Figure A.2. *Pseudocode algorithm of IMP from [Allen et al., 2019].*

A.2 Introduction to IMP and MC algorithms

A.2.1 Infinite Mixture Prototypes

The *Infinite Mixture Prototypes (IMP)* [Allen et al., 2019] algorithm represents each class by a set of clusters, instead of a single cluster like ProtoNet [Snell et al., 2017]. By inferring the number of clusters, IMP interpolates between nearest neighbor and prototypical representations. IMP can adapt its capacity to avoid underfitting by learning the cluster variance and by multi-modal clustering. Figure A.1 gives a schematic

```
Algorithm 1 Training MAML with the meta-curvature for few-shot supervised learning
  Input: task distribution $p(\mathcal{T})$, learning rate $\alpha$, $\beta$
  Initialize $\mathbf{M}_o, \mathbf{M}_i, \mathbf{M}_f = \mathbf{I}$
  while not converged do
    Sample batch of tasks $\tau_i \sim p(\mathcal{T})$
    for all $\tau_i$ do do
      $\theta^{\tau_i} = \theta - \alpha \mathbf{M}_{mc} \nabla \mathcal{L}_{\text{tr}}^{\tau_i}(\theta)$ {Assuming one gradient step}
    end for
    $\theta \leftarrow \text{ADAM}(\theta, \beta, \nabla_\theta \sum_{\tau_i} \mathcal{L}_{\text{val}}^{\tau_i}(\theta^{\tau_i}))$
    $\mathbf{M}_o \leftarrow \text{ADAM}(\mathbf{M}_o, \beta, \nabla_{\mathbf{M}_o} \sum_{\tau_i} \mathcal{L}_{\text{val}}^{\tau_i}(\theta^{\tau_i}))$
    $\mathbf{M}_i \leftarrow \text{ADAM}(\mathbf{M}_i, \beta, \nabla_{\mathbf{M}_i} \sum_{\tau_i} \mathcal{L}_{\text{val}}^{\tau_i}(\theta^{\tau_i}))$
    $\mathbf{M}_f \leftarrow \text{ADAM}(\mathbf{M}_f, \beta, \nabla_{\mathbf{M}_f} \sum_{\tau_i} \mathcal{L}_{\text{val}}^{\tau_i}(\theta^{\tau_i}))$
  end while
```

Figure A.3. *Pseudocode algorithm of MC from [Park and Oliva, 2019]. The model considered here has only one layer, it is straightforward to extend to multiple layers.*

view of the multi-modal representation.

Figure A.2 describes the IMP algorithm in pseudocode. Suppose we are given episodes with *support set* $\mathcal{S} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_K, y_K)\} \in \mathbb{X}^K \times \mathbb{R}^K$ of K labeled points and a *query set* $\mathcal{Q} = \{(\mathbf{x}'_1, y'_1), \ldots, (\mathbf{x}'_{K'}, y'_{K'})\} \in \mathbb{X}^{K'} \times \mathbb{R}^{K'}$ of K' labeled testing points. Within an episode, the cluster are initialized with class-wise means. Then, during inference, the distance to all existing clusters are computed for all support points. If the distance exceeds a threshold λ, a new cluster is made with mean equal to that point. IMP then updates soft cluster assignments $z_{i,c}$ as the normalized Gaussian density for cluster membership. Finally, cluster means μ_c are computed by the weighted mean of their members. Since each class can have multiple clusters, query points $\mathbf{x}'$ are classified by the softmax over distances to the closes cluster in each class n:

$$P_\phi(y' = n|\mathbf{x}') = \frac{\exp(-d(h_\phi(\mathbf{x}'), \mu_{c_n^*}))}{\sum_{n'} \exp(-d(h_\phi(\mathbf{x}'), \mu_{c_{n'}^*}))}, \tag{A.1}$$

with $c_n^* = \arg\min_{c:l_c=n} d(h_\phi(\mathbf{x}'), \mu_c)$ indexing the clusters, where each cluster c has label l_c, and $d(\cdot,\cdot)$ is the distance function of choice. IMP optimizes the embedding function h_ϕ with parameters ϕ and cluster variances σ by stochastic gradient descent across episodes.

A.2.2 Meta-Curvature

The *Meta-Curvature (MC)* [Park and Oliva, 2019] method learns a curvature for better generalization and faster model adaptation in the meta-learning framework. The curvature is learned along with the model's initial parameters simultaneously via the

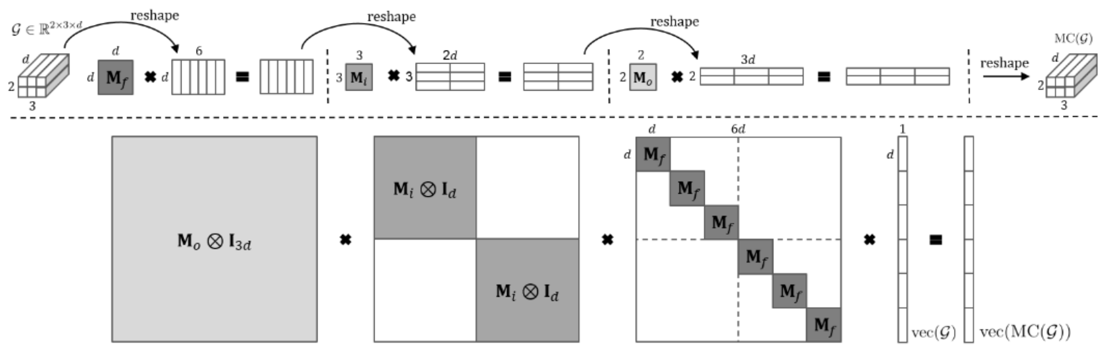


Figure A.4. *Illustration from [Park and Oliva, 2019] of meta-curvature computation with $G \in \mathbb{R}^{2\times3\times d}$.* (Top) *Tensor algebra view.* (Bottom) *Matrix-vector product view.*

meta-learning process. The goal is that the meta-learned curvature works collaboratively with the meta-learned model's initial parameters to produce good generalization performance on new tasks with fewer gradient steps. The meta-curvature is computed efficiently through tensor algebra.

If we consider convolutional neural networks as our model, we can formally define meta-curvature matrices $\mathbf{M}_o \in \mathbb{R}^{C_{out}\times C_{out}}$, $\mathbf{M}_i \in \mathbb{R}^{C_{in}\times C_{in}}$, and $\mathbf{M}_f \in \mathbb{R}^{d\times d}$, where C_{out}, C_{in} and d are the number of output channels, the number of input channels and the filter size respectively. Then the meta-curvature function for a multidimensional tensor G is computed through n-mode products and has all matrices as learnable parameters:

$$\mathrm{MC}(G) = G \times_3 \mathbf{M}_f \times_2 \mathbf{M}_i \times_1 \mathbf{M}_o. \tag{A.2}$$

Then, we can expand the matrices as follows:

$$\widehat{\mathbf{M}}_o = \mathbf{M}_o \otimes \mathbf{I}_{C_{in}} \otimes \mathbf{I}_d, \quad \widehat{\mathbf{M}}_i = \mathbf{I}_{C_{out}} \otimes \mathbf{M}_i \otimes \mathbf{I}_d, \quad \widehat{\mathbf{M}}_f = \mathbf{I}_{C_{out}} \otimes \mathbf{I}_{C_{in}} \otimes \mathbf{M}_f, \tag{A.3}$$

where $\otimes$ is the Kronecker product, $\mathbf{I}_k$ is the k dimensional identity matrix, and the three expanded matrices are all the same size $\widehat{\mathbf{M}}_o, \widehat{\mathbf{M}}_i, \widehat{\mathbf{M}}_f \in \mathbb{R}^{C_{out}C_{in}d\times C_{out}C_{in}d}$. Finally, the gradients can be transformed with the meta-curvature learned as:

$$\mathrm{vec}(\mathrm{MC}(G)) = \mathbf{M}_{mc}\mathrm{vec}(G), \tag{A.4}$$

where $\mathbf{M}_{mc} = \widehat{\mathbf{M}}_o\widehat{\mathbf{M}}_i\widehat{\mathbf{M}}_f$. Figure A.4 shows an example of computation with $G \in \mathbb{R}^{2\times3\times d}$, and Figure A.3 shows the details of the algorithm to train meta-curvature matrices and the initial model parameters.

A.3 Full proofs from Chapter 4

A.3.1 Proof of Theorem 4.3.1

Prototypical Loss We start by recalling the prototypical loss $\mathcal{L}_{proto}$ used during training of Prototypical Networks for a single episode with support set S and query set Q:

$$\begin{aligned}\mathcal{L}_{proto}(S, Q, \phi) &= \mathbb{E}_{(\mathbf{q},i)\sim Q}\left[-\log \frac{\exp(-d(\phi(\mathbf{q}), \mathbf{c}_i))}{\sum_j \exp\left(-d(\phi(\mathbf{q}), \mathbf{c}_j)\right)}\right] \\ &= \underbrace{\mathbb{E}_{(\mathbf{q},i)\sim Q}\left[d(\phi(\mathbf{q}), \mathbf{c}_i)\right]}_{(1)} \\ &\quad \underbrace{+\, \mathbb{E}_{\mathbf{q}\sim Q} \log \sum_{j=1}^{n} \exp\left(-d(\phi(\mathbf{q}), \mathbf{c}_j)\right)}_{(2)}\end{aligned}$$

with $\mathbf{c}_i = \frac{1}{k}\sum_{\mathbf{s}\in S_i} \phi(\mathbf{s})$ the prototype for class i, $S_i \subseteq S$ being the subset containing instances of S labeled with class i.

Distance For ProtoNet, we consider the Euclidean distance between the representation of a query example $\phi(\mathbf{q})$ and the prototype of a class i $\mathbf{c}_i$:

$$\begin{aligned}-d(\phi(\mathbf{q}), \mathbf{c}_i) &= -\|\phi(\mathbf{q}) - \mathbf{c}_i\|_2^2 && \text{(A.5)}\\ &= -\phi(\mathbf{q})^\top\phi(\mathbf{q}) + 2\mathbf{c}_i^\top\phi(\mathbf{q}) - \mathbf{c}_i^\top\mathbf{c}_i. && \text{(A.6)}\end{aligned}$$

Then, with respect to class i, the first term is constant and do not affect the softmax probabilities. The remaining terms are:

$$\begin{aligned}-d(\phi(\mathbf{q}), \mathbf{c}_i) &= 2\mathbf{c}_i^\top\phi(\mathbf{q}) - \|\mathbf{c}_i\|_2^2 && \text{(A.7)}\\ &= \frac{2}{|S_i|}\sum_{\mathbf{s}\in S_i} \phi(\mathbf{s})^\top\phi(\mathbf{q}) - \|\mathbf{c}_i\|_2^2. && \text{(A.8)}\end{aligned}$$

Now we can recall and prove Theorem 4.3.1:

Theorem (4.3.1). *(Normalized ProtoNet)*
If $\forall i\ \|\mathbf{c}_i\| = 1$, then $\forall\hat{\phi} \in \arg\min_\phi \mathcal{L}_{proto}(S, Q, \phi)$, the matrix of the optimal prototypes $\mathbf{W}^*$ is well-conditioned, *i.e.* $\kappa(\mathbf{W}^*) = O(1)$.

Proof. We can rewrite the first term in $\mathcal{L}_{proto}$ as

$$\mathbb{E}_{(\mathbf{q},i)\sim Q}\left[d(\phi(\mathbf{q}),\mathbf{c}_i)\right] \tag{A.9}$$

$$= -\mathbb{E}_{(\mathbf{q},i)\sim Q}\left[\frac{2}{|S_i|}\sum_{\mathbf{s}\in S_i}\phi(\mathbf{s})^\top\phi(\mathbf{q}) - \|\mathbf{c}_i\|_2^2\right] \tag{A.10}$$

$$= -\mathbb{E}_{(\mathbf{q},i)\sim Q}\left[\frac{2}{|S_i|}\sum_{\mathbf{s}\in S_i}\phi(\mathbf{s})^\top\phi(\mathbf{q})\right] \tag{A.11}$$

$$+ \mathbb{E}_{(\mathbf{q},i)\sim Q}\left[\|\mathbf{c}_i\|_2^2\right], \tag{A.12}$$

and the second term as

$$\mathbb{E}_{\mathbf{q}\sim Q}\left[\log\sum_{j=1}^{n}\exp\left(-d(\phi(\mathbf{q}),\mathbf{c}_j)\right)\right] \tag{A.13}$$

$$= \mathbb{E}_{\mathbf{q}\sim Q}\left[\log\sum_{j=1}^{n}\exp\left(\frac{2}{|S_j|}\sum_{\mathbf{s}\in S_j}\phi(\mathbf{s})^\top\phi(\mathbf{q}) - \|\mathbf{c}_j\|_2^2\right)\right] \tag{A.14}$$

$$= \mathbb{E}_{\mathbf{q}\sim Q}\left[\log\sum_{j=1}^{n}\exp\left(2\mathbf{c}_j^\top\phi(\mathbf{q}) - \|\mathbf{c}_j\|_2^2\right)\right] \tag{A.15}$$

$$= \mathbb{E}_{\mathbf{q}\sim Q}\left[\log\left(n\sum_{j=1}^{n}\frac{1}{n}\left[\exp\left(2\mathbf{c}_j^\top\phi(\mathbf{q}) - \|\mathbf{c}_j\|_2^2\right)\right]\right)\right] \tag{A.16}$$

$$= \mathbb{E}_{\mathbf{q}\sim Q}\left[\log\sum_{j=1}^{n}\frac{1}{n}\left[\exp\left(2\mathbf{c}_j^\top\phi(\mathbf{q}) - \|\mathbf{c}_j\|_2^2\right)\right] + \log n\right]. \tag{A.17}$$

$$\tag{A.18}$$

By dropping the constant part in the loss, we obtain:

$$\mathcal{L}_{proto}(S,Q,\phi) = -\mathbb{E}_{(\mathbf{q},i)\sim Q}\left[\frac{2}{|S_i|}\sum_{\mathbf{s}\in S_i}\phi(\mathbf{s})^\top\phi(\mathbf{q})\right] \tag{A.19}$$

$$+ \mathbb{E}_{\mathbf{q}\sim Q}\left[\log\sum_{j=1}^{n}\frac{1}{n}\left[\exp\left(2\mathbf{c}_j^\top\phi(\mathbf{q})\right)\right]\right]. \tag{A.20}$$

Let us note $\mathcal{S}^d$ the hypersphere of dimension d, and $\mathcal{M}(\mathcal{S}^d)$ the set of all possible Borel probability measures on $\mathcal{S}^d$. $\forall\mu\in\mathcal{M}(\mathcal{S}^d), u\in\mathcal{S}^d$, we further define the continuous and Borel measurable function:

$$U_\mu(u) := \int_{\mathcal{S}^d}\exp(2u^\top v)d\mu(v).$$

Then, we can write the second term as

$$\mathbb{E}_{\mathbf{q}\sim Q}\left[\log \mathbb{E}_{\mathbf{c}\sim C\circ\phi^{-1}}\left[\exp\left(2\phi(\mathbf{c})^{\top}\phi(\mathbf{q})\right)\right]\right] \tag{A.21}$$
$$= \mathbb{E}_{\mathbf{q}\sim Q}\left[\log U_{C\circ\phi^{-1}}(\phi(\mathbf{q}))\right], \tag{A.22}$$

where C is the distribution of prototypes of S, *i.e.*each data point in C is the mean of all the points in S that share the same label, and $C\circ\phi^{-1}$ is the probability measure of prototypes, *i.e.*the pushforward measure of C via ϕ.

We now consider the following problem:

$$\min_{\mu\in\mathcal{M}(\mathcal{S}^d)} \int_{\mathcal{S}^d} \log U_{\mu}(u)d\mu(u). \tag{A.23}$$

The unique minimizer of Eq. A.23 is the *uniform distribution on* $\mathcal{S}^d$, as shown in [Wang and Isola, 2020]. This means that learning with $\mathcal{L}_{proto}$ leads to prototypes uniformly distributed in the embedding space. By considering $\mathbf{W}^*$ the matrix of the optimal prototypes for each task then $\mathbf{W}^*$ is *well-conditioned*, *i.e.*$\kappa(\mathbf{W}^*) = O(1)$.

∎

A.3.2 Proof of Proposition 4.3.1

Let us first recall the learning model of interest:

$$\hat{y}_t = \langle \mathbf{w}_t, \mathbf{x}_t\rangle, \quad \ell_t = \mathbb{E}_{p(\mathbf{x}_t,y_t|\boldsymbol{\theta}_t)}(y_t - \langle \mathbf{w}_t, \mathbf{x}_t\rangle)^2. \tag{A.24}$$

We can now recall Proposition 4.3.1:

Proposition (4.3.1)**.** Let $\forall t \in [[T]]$, $\boldsymbol{\theta}_t \sim \mathcal{N}(\mathbf{0}_d, \boldsymbol{I}_d)$, $\mathbf{x}_t \sim \mathcal{N}(\mathbf{0}_d, \boldsymbol{I}_d)$ and $y_t \sim \mathcal{N}(\langle\boldsymbol{\theta}_t, \mathbf{x}_t\rangle, 1)$. Consider the learning model from Equation (A.24), let $\boldsymbol{\Theta}_i := [\boldsymbol{\theta}_i, \boldsymbol{\theta}_{i+1}]^T$, and denote by $\widehat{\mathbf{W}}_2^i$ the matrix of last two predictors learned by MAML at iteration i starting from $\widehat{\mathbf{w}}_0 = \mathbf{0}_d$. Then, we have that:

$$\forall i, \quad \kappa(\widehat{\mathbf{W}}_2^{i+1}) \geq \kappa(\widehat{\mathbf{W}}_2^{i}), \quad \text{if } \sigma_{\mathsf{min}}(\boldsymbol{\Theta}_i) = 0. \tag{A.25}$$

Proof. We follow [Arnold et al., 2021] and note that in the considered setup the gradient of the loss for each task is given by

$$\frac{\partial \ell_t(\widehat{\mathbf{w}} - \alpha\nabla\ell_t(\boldsymbol{\theta}))}{\partial\widehat{\mathbf{w}}} \propto (1-\alpha)^2(\widehat{\mathbf{w}}_t - \boldsymbol{\theta}_t)$$

so that the meta-training update for a single gradient step becomes:

$$\widehat{\mathbf{w}}_t \leftarrow \widehat{\mathbf{w}}_{t-1} - \beta(1-\alpha)^2(\widehat{\mathbf{w}}_{t-1} - \boldsymbol{\theta}_t),$$

where β is the meta-training update learning rate. Starting at $\widehat{\mathbf{w}}_0 = \mathbf{0}_d$, we have that

$$\begin{aligned}
&\widehat{\mathbf{w}}_1 = c\boldsymbol{\theta}_1,\\
&\widehat{\mathbf{w}}_2 = c((c-1)\boldsymbol{\theta}_1 + \boldsymbol{\theta}_2),\\
&\dots\\
&\widehat{\mathbf{w}}_n = c\sum_{i=1}^{n} \boldsymbol{\theta}_i (c-1)^{n-i},
\end{aligned}$$

where $c := \beta(1-\alpha)^2$. We can now define matrices $\widehat{\mathbf{W}}_2^i$ as follows:

$$\begin{aligned}
&\widehat{\mathbf{W}}_2^1 = \begin{pmatrix} c\boldsymbol{\theta}_1, \\ c((c-1)\boldsymbol{\theta}_1 + \boldsymbol{\theta}_2) \end{pmatrix},\\
&\widehat{\mathbf{W}}_2^2 = \begin{pmatrix} c((c-1)\boldsymbol{\theta}_1 + \boldsymbol{\theta}_2), \\ c((c-1)^2\boldsymbol{\theta}_1 + (c-1)\boldsymbol{\theta}_2 + \boldsymbol{\theta}_3) \end{pmatrix},\\
&\dots\\
&\widehat{\mathbf{W}}_2^n = \begin{pmatrix} c\sum_{i=1}^{n} \boldsymbol{\theta}_i (c-1)^{n-i}, \\ c\sum_{i=1}^{n+1} \boldsymbol{\theta}_i (c-1)^{n-i} \end{pmatrix}.
\end{aligned}$$

We can note that for all $i > 1$:

$$\widehat{\mathbf{W}}_2^{i+1} = (c-1)\widehat{\mathbf{W}}_2^i + c\boldsymbol{\Theta}_i.$$

Now, we can write:

$$\begin{aligned}
\kappa(\widehat{\mathbf{W}}_2^{i+1}) = \frac{\sigma_1(\widehat{\mathbf{W}}_2^{i+1})}{\sigma_2(\widehat{\mathbf{W}}_2^{i+1})} &= \frac{\sigma_1((c-1)\widehat{\mathbf{W}}_2^i + c\boldsymbol{\Theta}_i)}{\sigma_2((c-1)\widehat{\mathbf{W}}_2^i + c\boldsymbol{\Theta}_i)}\\
&\geq \frac{\sigma_1((c-1)\widehat{\mathbf{W}}_2^i) - \sigma_2(c\boldsymbol{\Theta}_i)}{\sigma_2((c-1)\widehat{\mathbf{W}}_2^i + c\boldsymbol{\Theta}_i)}\\
&\geq \frac{\sigma_1((c-1)\widehat{\mathbf{W}}_2^i) - \sigma_2(c\boldsymbol{\Theta}_i)}{\sigma_2((c-1)\widehat{\mathbf{W}}_2^i) + \sigma_2(c\boldsymbol{\Theta}_i)}\\
&\geq \kappa(\widehat{\mathbf{W}}_2^i).
\end{aligned}$$

where the second and third lines follow from the inequalities for singular values $\sigma_1(A+B) \leq \sigma_1(A) + \sigma_2(B)$ and $\sigma_i(A+B) \geq \sigma_i(A) - \sigma_{\min}(B)$ and the desired result is obtained by setting $\sigma_{\min}(\boldsymbol{\theta}_i) = 0$. ■

A.3.3 Proof of Proposition 4.3.2

Let us first recall Proposition 4.3.2:

> **Proposition** (4.3.2)**.** If $\forall t \in [[T]], \|\mathbf{w}_t^*\| = O(1)$ and $\kappa(\mathbf{W}^*) = O(1)$, and $\mathbf{w}_{T+1}$ follows a distribution ν such that $\|\mathbb{E}_{\mathbf{w}\sim\nu}[\mathbf{w}\mathbf{w}^\top]\| \leq O\left(\frac{1}{k}\right)$, then
>
> $$\mathrm{ER}(\hat{\phi}, \hat{\mathbf{w}}_{T+1}) \leq O\left(\frac{C(\Phi)}{n_1 T} \cdot \kappa(\mathbf{W}^*) + \frac{k}{n_2}\right).$$

Proof. Du et al. [Du et al., 2020] assume that $\sigma_k(\mathbf{W}^*) \gtrsim \frac{T}{k}$ (Assumption 4.3 in their work). However, since we also have $\|\mathbf{w}_t^*\| = O(1)$, it is equivalent to $\frac{\sigma_1(\mathbf{W}^*)}{\sigma_k(\mathbf{W}^*)} = O(1)$. We have $\sigma_1(\mathbf{W}^*) \gtrsim \sigma_k(\mathbf{W}^*) \gtrsim \frac{T}{k}$ and then $\frac{\sigma_1(\mathbf{W}^*)}{T\cdot\sigma_k(\mathbf{W}^*)} = \frac{1}{T}\cdot\kappa(\mathbf{W}^*) \gtrsim \frac{1}{k\cdot\sigma_k(\mathbf{W}^*)}$ which we use in their proof of Theorem 5.1 instead of $\frac{1}{T} \gtrsim \frac{1}{k\cdot\sigma_k(\mathbf{W}^*)}$ to obtain the desired result. ■

A.3.4 Proof of Proposition 4.3.3

Let us recall the data generating process and Proposition 4.3.3:

$$\forall t \in [[T+1]] \text{ and } (\mathbf{x}, y) \sim \mu_t, \quad y = \langle \mathbf{w}_t^*, \phi^*(\mathbf{x})\rangle + \varepsilon, \quad \varepsilon \sim \mathcal{N}(0, \sigma^2). \tag{A.26}$$

> **Proposition** (4.3.3)**.** Let $T = 2$, $\mathbb{X} \subseteq \mathbb{R}^d$ be the input space and $\mathbb{Y} = \{-1, 1\}$ be the output space. Then, there exist distributions μ_1 and μ_2 over $\mathbb{X}\times\mathbb{Y}$, representations $\widehat{\phi} \neq \phi^*$ and matrices of predictors $\widehat{\mathbf{W}} \neq \mathbf{W}^*$ that satisfy the data generating model (Equation (A.26)) with $\kappa(\widehat{\mathbf{W}}) \approx 1$ and $\kappa(\mathbf{W}^*) \gg 1$.

Proof. Let us define two uniform distributions μ_1 and μ_2 parametrized by a scalar $\varepsilon > 0$ satisfying the data generating process from Equation (A.26):

1. μ_1 is uniform over $\{1 - k\varepsilon, k, 1, \underbrace{\ldots}_{d-3}\} \times \{1\} \cup \{1 + k\varepsilon, k, -1, \underbrace{\ldots}_{d-3}\} \times \{-1\}$;
2. μ_2 is uniform over $\{1 + k\varepsilon, k, \frac{k-1}{\varepsilon}, \underbrace{\ldots}_{d-3}\} \times \{1\} \cup \{-1 + k\varepsilon, k, \frac{1+k}{\varepsilon}, \underbrace{\ldots}_{d-3}\} \times \{-1\}$.

where last $d - 3$ coordinates of the generated instances are arbitrary numbers. We now define the optimal representation and two optimal predictors for each distribution as the solution to the MTR problem over the two data generating distributions and $\Phi = \{\phi |\ \phi(\mathbf{x}) = \mathbf{\Phi}^T\mathbf{x},\ \mathbf{\Phi} \in \mathbb{R}^{d\times 2}\}$:

$$\phi^*, \mathbf{W}^* = \underset{\phi\in\Phi, \mathbf{W}\in\mathbb{R}^{2\times 2}}{\arg\min} \sum_{i=1}^{2} \mathop{\mathbb{E}}_{(\mathbf{x},y)\sim\mu_i} \ell(y, \langle \mathbf{w}_i, \phi(\mathbf{x})\rangle), \tag{A.27}$$

One solution to this problem can be given as follows:

$$\mathbf{\Phi}^* = \begin{pmatrix} 1 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & \dots & 0 \end{pmatrix}^T, \quad \mathbf{W}^* = \begin{pmatrix} 1 & \varepsilon \\ 1 & -\varepsilon \end{pmatrix},$$

where $\mathbf{\Phi}^*$ projects the data generated by μ_i to a two-dimensional space by discarding its $d-2$ last dimensions and the linear predictors satisfy the data generating process from Eq. 4.3 with $\varepsilon = 0$. One can verify that in this case $\mathbf{W}^*$ have singular values equal to $\sqrt{2}$ and $\sqrt{2}\varepsilon$, and $\kappa(\mathbf{W}^*) = \frac{1}{\varepsilon}$. When $\varepsilon \to 0$, the optimal predictors make the ratio arbitrary large thus violating Assumption 1.

Let us now consider a different problem where we want to solve Eq. A.27 with constraints that force linear predictors to satisfy both assumptions:

$$\begin{aligned} \widehat{\phi}, \widehat{\mathbf{W}} = \underset{\phi\in\Phi, \mathbf{W}\in\mathbb{R}^{2\times 2}}{\arg\min} \sum_{i=1}^{2} \mathop{\mathbb{E}}_{(\mathbf{x},y)\sim\mu_i} \ell(y, \langle \mathbf{w}_i, \phi(\mathbf{x})\rangle), \\ \text{s.t. } \kappa(\mathbf{W}) \approx 1 \quad \text{and} \quad \forall i, \quad \|\mathbf{w}_i\| \approx 1. \end{aligned} \tag{A.28}$$

Its solution is different and is given by

$$\widehat{\mathbf{\Phi}} = \begin{pmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \end{pmatrix}^T, \quad \widehat{\mathbf{W}} = \begin{pmatrix} 0 & 1 \\ 1 & -\varepsilon \end{pmatrix}.$$

Similarly to $\mathbf{\Phi}^*$, $\widehat{\mathbf{\Phi}}$ projects to a two-dimensional space by discarding the first and last $d-3$ dimensions of the data generated by μ_i. The learned predictors in this case also satisfy Eq. 4.3 with $\varepsilon = 0$, but contrary to $\mathbf{W}^*$, $\kappa(\widehat{\mathbf{W}}) = \sqrt{\frac{2+\varepsilon^2+\varepsilon\sqrt{\varepsilon^2+4}}{2+\varepsilon^2-\varepsilon\sqrt{\varepsilon^2+4}}}$ tends to 1 when $\varepsilon \to 0$. ■

A.4 Detailed Performance Comparisons from Section 4.4

Table A.2 provides the detailed performance of our reproduced methods with and without our regularization for gradient-based methods in Table A.2b, or normalization for metric-based methods in Table A.2a. Figure A.5 shows the performance gap throughout training for all methods on miniImageNet. We can see on both Table A.2 and Figure A.5 that the gap is globally positive, which shows the increased generalization capabilities of enforcing the assumptions.

Method	Dataset	Episodes	without Reg./Norm.	with Reg./Norm.
ProtoNet	Omniglot	1-shot	$95.56 \pm 0.10\%$	$\mathbf{95.89 \pm 0.10\%}$
		5-shot	$\mathbf{98.80 \pm 0.04\%}$	$\mathbf{98.80 \pm 0.04\%}$
	miniImageNet	1-shot	$49.53 \pm 0.41\%$	$\mathbf{50.29 \pm 0.41\%}$
		5-shot	$65.10 \pm 0.35\%$	$\mathbf{67.13 \pm 0.34\%}$
	tieredImageNet	1-shot	$51.95 \pm 0.45\%$	$\mathbf{54.05 \pm 0.45\%}$
		5-shot	$\mathbf{71.61 \pm 0.38\%}$	$\mathbf{71.84 \pm 0.38\%}$
IMP	Omniglot	1-shot	$\mathbf{95.77 \pm 0.20\%}$	$\mathbf{95.85 \pm 0.20\%}$
		5-shot	$\mathbf{98.77 \pm 0.08\%}$	$\mathbf{98.83 \pm 0.07\%}$
	miniImageNet	1-shot	$48.85 \pm 0.81\%$	$\mathbf{50.69 \pm 0.80\%}$
		5-shot	$66.43 \pm 0.71\%$	$\mathbf{67.29 \pm 0.68\%}$
	tieredImageNet	1-shot	$52.16 \pm 0.89\%$	$\mathbf{53.46 \pm 0.89\%}$
		5-shot	$\mathbf{71.79 \pm 0.75\%}$	$\mathbf{72.38 \pm 0.75\%}$

***(a)** Metric-based methods*

A.5 Detailed computations of CO2 emissions from popular dataset labelization

We measure in this section the carbon footprint resulting from the labelization of two popular datasets widely used in the literature, namely ILSVRC-2012 [Russakovsky et al., 2015] and COCO [Lin et al., 2014].

The ILSVRC-2012 [Russakovsky et al., 2015] dataset, is composed of about 1.2M labeled images. The annotation tasks were given to Amazon Mechanical Turk (AMT) to parallelize the process, using about 5 independent annotators (turkers) to verify

Method	Dataset	Episodes	without Reg./Norm.	with Reg./Norm.
MAML	Omniglot	1-shot	$91.72 \pm 0.29\%$	$\mathbf{95.67 \pm 0.20\%}$
		5-shot	$97.07 \pm 0.14\%$	$\mathbf{98.24 \pm 0.10\%}$
	miniImageNet	1-shot	$47.93 \pm 0.83\%$	$\mathbf{49.16 \pm 0.85\%}$
		5-shot	$64.47 \pm 0.69\%$	$\mathbf{66.43 \pm 0.69\%}$
	tieredImageNet	1-shot	$50.08 \pm 0.91\%$	$\mathbf{51.5 \pm 0.90\%}$
		5-shot	$67.5 \pm 0.79\%$	$\mathbf{70.16 \pm 0.76\%}$
MC	Omniglot	1-shot	$\mathbf{96.56 \pm 0.18\%}$	$95.95 \pm 0.20\%$
		5-shot	$\mathbf{98.88 \pm 0.08\%}$	$98.78 \pm 0.08\%$
	miniImageNet	1-shot	$\mathbf{49.28 \pm 0.83\%}$	$\mathbf{49.64 \pm 0.83\%}$
		5-shot	$63.74 \pm 0.69\%$	$\mathbf{65.67 \pm 0.70\%}$
	tieredImageNet	1-shot	$55.16 \pm 0.94\%$	$\mathbf{55.85 \pm 0.94\%}$
		5-shot	$71.95 \pm 0.77\%$	$\mathbf{73.34 \pm 0.76\%}$

(b) *Gradient-based methods*

Table A.2. *Exact performance of the meta-learning setting considered in Section 4.4, without and with our regularization (or normalization in the case of ProtoNet and IMP) to enforce the theoretical assumptions. All accuracy results (in %) are averaged over 2400 test episodes and 4 different seeds and are reported with 95% confidence interval. Episodes are 20-way classification for Omniglot and 5-way classification for miniImageNet and tieredImageNet.*

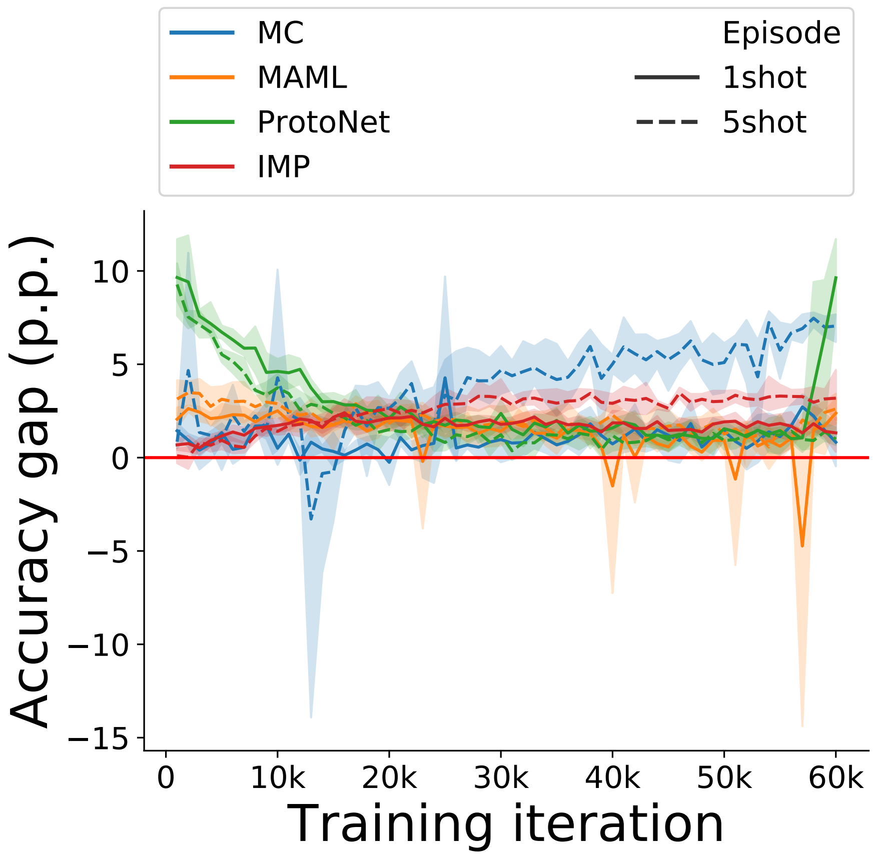


Figure A.5. *Performance gap (in p.p.) when applying regularization for gradient-based and normalization for metric-based methods throughout the training process on 5-way 1-shot and 5-shot episodes on miniImageNet (*better viewed in color*). Each data point is averaged over 2400 validation episodes and 4 different seeds and shaded areas report 95% confidence interval. We can see that the gap is globally positive throughout training and generally higher at the beginning of training. The increase in the gap at the end of training is linked to a lower overfitting.*

Country	AMT demography	Carbon intensity of electricity
USA	47%	379
India	34%	633
Other	19%	442
France	–	68

Table A.3. *General demography of AMT (in % of total AMT users) with the corresponding carbon intensity of the electricity grid (in gCO2eq/kWh) of each country. The carbon intensity of the* other *category is taken as the world average, and the value for France is given for reference.*

each image [Russakovsky et al., 2015], leading to about 6M total labels given. Since the average user can identify about 250 images in five minutes[1], this results in about 2000 worker hours. Turkers are using their own computers to complete the annotation tasks, and with an average computer requiring about 300 W, we obtain a total electrical consumption for the workers of about 600 kWh, without taking into account the Amazon servers running the services behind. Then, from the average demography of AMT [Ipeirotis, 2010] and the carbon intensity of the electricity grid for each country[2] given in Table A.3, we can compute an estimate of the total electric consumption of about 1200 kWh and a resulting carbon footprint of the annotation process of about *286.4 kgCO2eq*. To compare with an actual training experiment, we suppose that we have a node with 8 A100 GPUs, and we estimate that each GPU require 400 W[3]. Thus, from Table A.3, the annotation process is equivalent to the emissions from *55 days of computation* if the node is located in France, and about *10 days* if it is located in the USA.

For the COCO dataset [Lin et al., 2014], the authors report the time taken for each stage of the annotation process. The *Category Labeling* stage took about 20 000 worker hours, the *Instance Spotting* stage about 10 000 worker hours, and the *Instance Segmentation* stage about 55 000 worker hours, resulting in a total of *85 000 worker hours*. Using the values given in Table A.3, we obtain a total electric consumption of about 25 500 kWh and a carbon footprint of about *12 tons of CO2eq*. This footprint is equivalent to *51 years* of computation of a node with 8 A100 GPUs running at 100% located in France, and about *9 years* if the node is located in the USA.

[1] https://www.nytimes.com/2012/11/20/science/for-web-images-creating-new-technology-to-seek-and-find.html

[2] https://ourworldindata.org/grapher/carbon-intensity-electricity

[3] https://www.nvidia.com/en-us/data-center/a100/

Bibliography

[Akiba et al., 2017] Akiba, T., Suzuki, S., and Fukuda, K. (2017). Extremely large minibatch SGD: training resnet-50 on imagenet in 15 minutes. *CoRR*, abs/1711.04325. 130

[Alet et al., 2020] Alet, F., Schneider, M. F., Lozano-Perez, T., and Kaelbling, L. P. (2020). Meta-learning curiosity algorithms. In *International Conference on Learning Representations*. 39

[Allen et al., 2019] Allen, K., Shelhamer, E., Shin, H., and Tenenbaum, J. (2019). Infinite mixture prototypes for few-shot learning. In *Proceedings of the 36th International Conference on Machine Learning*. 39, 41, 59, 63, 64, 82, III, XLIX

[Amit and Meir, 2018] Amit, R. and Meir, R. (2018). Meta-learning by adjusting priors based on extended PAC-Bayes theory. In *International Conference on Machine Learning*. 59, 61

[Antoniou et al., 2019] Antoniou, A., Edwards, H., and Storkey, A. (2019). How to train your MAML. In *International Conference on Learning Representations*. 39, 40

[Arnold et al., 2021] Arnold, S., Iqbal, S., and Sha, F. (2021). When MAML can adapt fast and how to assist when it cannot. In *AISTATS*. 40, 68, VIII

[Ba et al., 2016] Ba, J. L., Kiros, J. R., and Hinton, G. E. (2016). Layer normalization. *arXiv preprint arXiv:1607.06450*. 26

[Balaji et al., 2018] Balaji, Y., Sankaranarayanan, S., and Chellappa, R. (2018). MetaReg: Towards domain generalization using meta-regularization. In *Advances in Neural Information Processing Systems*, pages 998–1008. 73

[Balcan et al., 2019] Balcan, M., Khodak, M., and Talwalkar, A. (2019). Provable guarantees for gradient-based meta-learning. In *International Conference on Machine Learning*. 59

[Bar et al., 2022] Bar, A., Wang, X., Kantorov, V., Reed, C. J., Herzig, R., Chechik, G., Rohrbach, A., Darrell, T., and Globerson, A. (2022). Detreg: Unsupervised pretraining with region priors for object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 47, 51, 52, 81, 88, 90, 91, 93, 98, 99, 101, 103, 106, XLVI

[Bardes et al., 2022] Bardes, A., Ponce, J., and LeCun, Y. (2022). Vicreg: Variance-invariance-covariance regularization for self-supervised learning. In *ICLR*. OpenReview.net. 49

[Bartlett and Mendelson, 2002] Bartlett, P. L. and Mendelson, S. (2002). Rademacher and gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research*, 3(Nov):463–482. 16

[Bateni et al., 2020] Bateni, P., Goyal, R., Masrani, V., Wood, F., and Sigal, L. (2020). Improved few-shot visual classification. In *CVPR*, pages 14481–14490. Computer Vision Foundation / IEEE. 41

[Baxter, 2000] Baxter, J. (2000). A model of inductive bias learning. *Journal of Artificial Intelligence Research*, 12:149–198. 59, 61

[Berthelot et al., 2020] Berthelot, D., Carlini, N., Cubuk, E. D., Kurakin, A., Sohn, K., Zhang, H., and Raffel, C. (2020). Remixmatch: Semi-supervised learning with distribution matching and augmentation anchoring. In *ICLR*. OpenReview.net. 54

[Berthelot et al., 2019] Berthelot, D., Carlini, N., Goodfellow, I., Papernot, N., Oliver, A., and Raffel, C. A. (2019). Mixmatch: A holistic approach to semi-supervised learning. *Advances in neural information processing systems*, 32. 54

[Bertinetto et al., 2019] Bertinetto, L., Henriques, J. F., Torr, P. H. S., and Vedaldi, A. (2019). Meta-learning with differentiable closed-form solvers. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. 39, 40, 43, 59, 63

[Beyer et al., 2019] Beyer, L., Zhai, X., Oliver, A., and Kolesnikov, A. (2019). S4L: self-supervised semi-supervised learning. In *ICCV*, pages 1476–1485. IEEE. 54

[Bochkovskiy et al., 2020] Bochkovskiy, A., Wang, C.-Y., and Liao, H.-Y. M. (2020). Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*. 35

[Bodla et al., 2017] Bodla, N., Singh, B., Chellappa, R., and Davis, L. S. (2017). Soft-nms–improving object detection with one line of code. In *Proceedings of the IEEE international conference on computer vision*, pages 5561–5569. 111

[Boser et al., 1992] Boser, B. E., Guyon, I. M., and Vapnik, V. N. (1992). A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152. 18

[Bottou et al., 1998] Bottou, L. et al. (1998). Online learning and stochastic approximations. *On-line learning in neural networks*, 17(9):142. 22

[Bouniot et al., 2023a] Bouniot, Q., Audigier, R., Loesch, A., and Habrard, A. (2023a). Proposal-contrastive pretraining for object detection from fewer data. In *The Eleventh International Conference on Learning Representations*. 11, 12

[Bouniot et al., 2023b] Bouniot, Q., Loesch, A., Audigier, R., and Habrard, A. (2023b). Towards few-annotation learning for object detection: Are transformer-based models more efficient? In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 75–84. 11, 12

[Bouniot and Redko, 2022] Bouniot, Q. and Redko, I. (2022). Understanding few-shot multi-task representation learning theory. In *ICLR Blog Track*. 12

[Bouniot et al., 2020] Bouniot, Q., Redko, I., Audigier, R., Loesch, A., and Habrard, A. (2020). Putting theory to work: From learning bounds to meta-learning. In *NeurIPS Workshop on Meta-Learning (MetaLearn)*. 11, 12

[Bouniot et al., 2021] Bouniot, Q., Redko, I., Audigier, R., Loesch, A., and Habrard, A. (2021). Vers une meilleure compréhension des méthodes de méta-apprentissage à travers la théorie de l'apprentissage de réprésentations multi-tâches. In *CAp*. 12

[Bouniot et al., 2022] Bouniot, Q., Redko, I., Audigier, R., Loesch, A., and Habrard, A. (2022). Improving few-shot learning through multi-task representation learning theory. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XX*, pages 435–452. Springer. 11, 12

[Buciluǎ et al., 2006] Buciluǎ, C., Caruana, R., and Niculescu-Mizil, A. (2006). Model compression. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 535–541. 117

[Cai et al., 2018] Cai, Q., Pan, Y., Yao, T., Yan, C., and Mei, T. (2018). Memory matching networks for one-shot image recognition. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 4080–4088. Computer Vision Foundation / IEEE Computer Society. 41

[Cai et al., 2020] Cai, T. T., Frankle, J., Schwab, D. J., and Morcos, A. S. (2020). Are all negatives created equal in contrastive instance discrimination? *arXiv preprint arXiv:2010.06682*. 49

[Cao et al., 2020] Cao, T., Law, M. T., and Fidler, S. (2020). A theoretical analysis of the number of shots in few-shot learning. In *ICLR*. 73

[Carion et al., 2020] Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., and Zagoruyko, S. (2020). End-to-end object detection with transformers. In *European conference on computer vision*, pages 213–229. Springer. 3, 25, 27, 31, 89, 90, 92, 99, 110, 116, 118, 119, 126, XLIV, LII

[Caron et al., 2020] Caron, M., Misra, I., Mairal, J., Goyal, P., Bojanowski, P., and Joulin, A. (2020). Unsupervised learning of visual features by contrasting cluster assignments. *Advances in Neural Information Processing Systems*. 48, 49, 51, 88, 101, XLVI

[Caron et al., 2021] Caron, M., Touvron, H., Misra, I., Jégou, H., Mairal, J., Bojanowski, P., and Joulin, A. (2021). Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9650–9660. 48, 49, 54, 113, 117, 118

[Chapelle et al., 2009] Chapelle, O., Scholkopf, B., and Zien, A. (2009). Semi-supervised learning (chapelle, o. et al., eds.; 2006)[book reviews]. *IEEE Transactions on Neural Networks*, 20(3):542–542. 3

[Chawla et al., 2002] Chawla, N. V., Bowyer, K. W., Hall, L. O., and Kegelmeyer, W. P. (2002). Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357. 35

[Chen et al., 2018] Chen, D., Wang, W., Gao, W., and Zhou, Z. (2018). Tri-net for semi-supervised deep learning. In *IJCAI*, pages 2014–2020. ijcai.org. 54

[Chen et al., 2020a] Chen, J., Wu, X.-M., Li, Y., Li, Q., Zhan, L.-M., and Chung, F.-l. (2020a). A closer look at the training strategy for modern meta-learning. *Advances in Neural Information Processing Systems*. 44, 63

[Chen et al., 2020b] Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. (2020b). A simple framework for contrastive learning of visual representations. In *International conference on machine learning*. PMLR. 35, 49, 95, 99, XLIV

[Chen et al., 2020c] Chen, T., Kornblith, S., Swersky, K., Norouzi, M., and Hinton, G. E. (2020c). Big self-supervised models are strong semi-supervised learners. *Advances in neural information processing systems*, 33:22243–22255. 48, 49, 54, 88, 112, 117, 130

[Chen et al., 2019] Chen, W.-Y., Wang, Y.-C. F., Liu, Y.-C., Kira, Z., and Huang, J.-B. (2019). A closer look at few-shot classification. In *ICLR*. 47, 48, 56, 73, 75, 129, XLV

[Chen et al., 2020d] Chen, X., Fan, H., Girshick, R. B., and He, K. (2020d). Improved baselines with momentum contrastive learning. *CoRR*, abs/2003.04297. 49

[Chen et al., 2020e] Chen, X., Wang, Z., Tang, S., and Muandet, K. (2020e). MATE: Plugging in model awareness to task embedding for meta learning. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. F., and Lin, H., editors, *Advances in neural information processing systems*, volume 33, pages 11865–11877. Curran Associates, Inc. 59, 63, 82

[Chen et al., 2021a] Chen, X., Xie, S., and He, K. (2021a). An empirical study of training self-supervised vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 49

[Chen et al., 2021b] Chen, Y., Liu, Z., Xu, H., Darrell, T., and Wang, X. (2021b). Meta-baseline: Exploring simple meta-learning for few-shot learning. In *ICCV*, pages 9042–9051. IEEE. 44

[Chuang et al., 2020] Chuang, C., Robinson, J., Lin, Y., Torralba, A., and Jegelka, S. (2020). Debiased contrastive learning. In *NeurIPS*. 49

[Cortes and Vapnik, 1995] Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine learning*, 20(3):273–297. 18

[Cubuk et al., 2019] Cubuk, E. D., Zoph, B., Mane, D., Vasudevan, V., and Le, Q. V. (2019). Autoaugment: Learning augmentation strategies from data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 113–123. 34

[Dai et al., 2016] Dai, J., Li, Y., He, K., and Sun, J. (2016). R-fcn: Object detection via region-based fully convolutional networks. *Advances in neural information processing systems*, 29. 29

[Dai et al., 2021a] Dai, X., Chen, Y., Yang, J., Zhang, P., Yuan, L., and Zhang, L. (2021a). Dynamic detr: End-to-end object detection with dynamic attention. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 31, 90

[Dai et al., 2021b] Dai, Z., Cai, B., Lin, Y., and Chen, J. (2021b). Up-detr: Unsupervised pre-training for object detection with transformers. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 51, 88, 91, 103

[Dalal and Triggs, 2005] Dalal, N. and Triggs, B. (2005). Histograms of oriented gradients for human detection. In *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, volume 1, pages 886–893. Ieee. 29

[David and Siegel, 1956] David, F. N. and Siegel, S. (1956). Nonparametric statistics for the behavioral sciences. *Biometrika*, 44:538. 77, 78

[Denevi et al., 2019] Denevi, G., Ciliberto, C., Grazzi, R., and Pontil, M. (2019). Learning-to-learn stochastic gradient descent with biased regularization. In *International Conference on Machine Learning*. 59, 73

[Denevi et al., 2018a] Denevi, G., Ciliberto, C., Stamos, D., and Pontil, M. (2018a). Incremental learning-to-learn with statistical guarantees. In *Conference on Uncertainty in Artificial Intelligence*, pages 457–466. 73

[Denevi et al., 2018b] Denevi, G., Ciliberto, C., Stamos, D., and Pontil, M. (2018b). Learning to learn around a common mean. In *Advances in Neural Information Processing Systems*, pages 10169–10179. 73

[Deng et al., 2009] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee. 28

[Denize et al., 2023] Denize, J., Rabarisoa, J., Orcesi, A., Hérault, R., and Canu, S. (2023). Similarity contrastive estimation for self-supervised soft contrastive learning. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 2706–2716. 49, 91, 93, 95, 98, 103

[DeVries and Taylor, 2017a] DeVries, T. and Taylor, G. W. (2017a). Dataset augmentation in feature space. *arXiv preprint arXiv:1702.05538*. 35

[DeVries and Taylor, 2017b] DeVries, T. and Taylor, G. W. (2017b). Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*. 35

[Dhillon et al., 2020] Dhillon, G. S., Chaudhari, P., Ravichandran, A., and Soatto, S. (2020). A baseline for few-shot image classification. In *ICLR*. 48

[Doersch et al., 2015] Doersch, C., Gupta, A., and Efros, A. A. (2015). Unsupervised visual representation learning by context prediction. In *ICCV*, pages 1422–1430. IEEE Computer Society. 48

[Dosovitskiy et al., 2021] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. (2021). An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*. 3, 25, 27

[Dosovitskiy et al., 2016] Dosovitskiy, A., Fischer, P., Springenberg, J. T., Riedmiller, M. A., and Brox, T. (2016). Discriminative unsupervised feature learning with exemplar convolutional neural networks. *IEEE Trans. Pattern Anal. Mach. Intell.*, 38(9):1734–1747. 48

[Du et al., 2020] Du, S. S., Hu, W., Kakade, S. M., Lee, J. D., and Lei, Q. (2020). Few-shot learning via learning the representation, provably. In *arXiv:2002.09434*. 56, 59, 62, 63, 69, 129, I, II, X

[Dwibedi et al., 2021] Dwibedi, D., Aytar, Y., Tompson, J., Sermanet, P., and Zisserman, A. (2021). With a little help from my friends: Nearest-neighbor contrastive learning of visual representations. In *ICCV*, pages 9568–9577. IEEE. 49

[Ericsson et al., 2021] Ericsson, L., Gouk, H., and Hospedales, T. M. (2021). How well do self-supervised models transfer? In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*, pages 5414–5423. Computer Vision Foundation / IEEE. 129

[Everingham et al., 2010] Everingham, M., Van Gool, L., Williams, C. K., Winn, J., and Zisserman, A. (2010). The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338. 29, 46, 98, 120

[Fan et al., 2020a] Fan, Q., Zhuo, W., Tang, C.-K., and Tai, Y.-W. (2020a). Few-shot object detection with attention-rpn and multi-relation detector. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 45, 46, 98, L

[Fan et al., 2021] Fan, Z., Ma, Y., Li, Z., and Sun, J. (2021). Generalized few-shot object detection without forgetting. In *CVPR*, pages 4527–4536. Computer Vision Foundation / IEEE. 52

[Fan et al., 2020b] Fan, Z., Yu, J., Liang, Z., Ou, J., Gao, C., Xia, G., and Li, Y. (2020b). FGN: fully guided network for few-shot instance segmentation. In *CVPR*, pages 9169–9178. Computer Vision Foundation / IEEE. 45

[Finn et al., 2017] Finn, C., Abbeel, P., and Levine, S. (2017). Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*. 39, 40, 44, 47, 56, 59, 63, 75, 82, 129, 131, XLV

[Finn and Levine, 2018] Finn, C. and Levine, S. (2018). Meta-learning and universality: Deep representations and gradient descent can approximate any learning algorithm. In *ICLR (Poster)*. OpenReview.net. 40, 41

[Finn et al., 2019] Finn, C., Rajeswaran, A., Kakade, S. M., and Levine, S. (2019). Online meta-learning. In *International Conference on Machine Learning*. 59

[Franceschi et al., 2018] Franceschi, L., Frasconi, P., Salzo, S., Grazzi, R., and Pontil, M. (2018). Bilevel programming for hyperparameter optimization and meta-learning. In *International Conference on Machine Learning*, pages 1568–1577. PMLR. 39

[Fu et al., 2015] Fu, Y., Hospedales, T. M., Xiang, T., and Gong, S. (2015). Transductive multi-view zero-shot learning. *IEEE transactions on pattern analysis and machine intelligence*, 37(11):2332–2345. 35

[Gao et al., 2019] Gao, J., Wang, J., Dai, S., Li, L., and Nevatia, R. (2019). NOTE-RCNN: noise tolerant ensemble RCNN for semi-supervised object detection. In *ICCV*, pages 9507–9516. IEEE. 54

[Gehring et al., 2017] Gehring, J., Auli, M., Grangier, D., Yarats, D., and Dauphin, Y. N. (2017). Convolutional sequence to sequence learning. In *International conference on machine learning*, pages 1243–1252. PMLR. 26

[Gidaris et al., 2019] Gidaris, S., Bursuc, A., Komodakis, N., Pérez, P., and Cord, M. (2019). Boosting few-shot visual learning with self-supervision. In *ICCV*, pages 8058–8067. IEEE. 48

[Gidaris and Komodakis, 2018] Gidaris, S. and Komodakis, N. (2018). Dynamic Few-Shot Visual Learning Without Forgetting. In *CVPR*, pages 4367–4375. 41, 48, 73

[Gidaris et al., 2018] Gidaris, S., Singh, P., and Komodakis, N. (2018). Unsupervised representation learning by predicting image rotations. In *International Conference on Learning Representations (ICLR)*. 48

[Girshick, 2015] Girshick, R. (2015). Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448. 29, 90, 111

[Girshick et al., 2014] Girshick, R., Donahue, J., Darrell, T., and Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587. 29, 90, 111

[Gladwell, 2009] Gladwell, M. (2009). *Outliers*. Back Bay Books. 1

[Goldberger et al., 2005] Goldberger, J., Hinton, G. E., Roweis, S., and Salakhutdinov, R. R. (2005). Neighbourhood components analysis. In Saul, L., Weiss, Y., and Bottou, L., editors, *NeurIPS*, pages 513–520. 63

[Goldblum et al., 2020] Goldblum, M., Reich, S., Fowl, L., Ni, R., Cherepanova, V., and Goldstein, T. (2020). Unraveling meta-learning: Understanding feature representations for few-shot tasks. In *International Conference on Machine Learning*. 72, 73

[Goodfellow et al., 2016] Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. `http://www.deeplearningbook.org`. 2, 3, 14, 17, 18, 24, XLIII, XLIV

[Grandvalet and Bengio, 2004] Grandvalet, Y. and Bengio, Y. (2004). Semi-supervised learning by entropy minimization. In *NIPS*, pages 529–536. 54

[Graves et al., 2014] Graves, A., Wayne, G., and Danihelka, I. (2014). Neural turing machines. *CoRR*, abs/1410.5401. 41

[Grill et al., 2020] Grill, J.-B., Strub, F., Altché, F., Tallec, C., Richemond, P., Buchatskaya, E., Doersch, C., Avila Pires, B., Guo, Z., Gheshlaghi Azar, M., et al. (2020). Bootstrap your own latent-a new approach to self-supervised learning. *Advances in neural information processing systems*, 33:21271–21284. 48, 49, 51, 91, 93, 113, 118

[Guo et al., 2020] Guo, Y., Codella, N. C., Karlinsky, L., Codella, J. V., Smith, J. R., Saenko, K., Rosing, T., and Feris, R. (2020). A broader study of cross-domain few-shot learning. In *Computer Vision – ECCV 2020*. Springer Int. Publishing. 80

[Gupta et al., 2020] Gupta, G., Yadav, K., and Paull, L. (2020). Look-ahead meta learning for continual learning. *Advances in Neural Information Processing Systems*, 33:11588–11598. 56, 129

[Han et al., 2022] Han, G., Huang, S., Ma, J., He, Y., and Chang, S. (2022). Meta faster R-CNN: towards accurate few-shot object detection with attentive feature alignment. In *AAAI*, pages 780–789. AAAI Press. 45

[Hariharan and Girshick, 2017] Hariharan, B. and Girshick, R. (2017). Low-shot visual recognition by shrinking and hallucinating features. In *ICCV*, pages 3037–3046. 37

[Hastie et al., 2009] Hastie, T., Tibshirani, R., Friedman, J. H., and Friedman, J. H. (2009). *The elements of statistical learning: data mining, inference, and prediction*, volume 2. Springer. 16

[He et al., 2020] He, K., Fan, H., Wu, Y., Xie, S., and Girshick, R. (2020). Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 48, 49, 88, 95, 103

[He et al., 2019] He, K., Girshick, R., and Dollar, P. (2019). Rethinking imagenet pre-training. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 28, 29, 50, 124, 129, 131

[He et al., 2015] He, K., Zhang, X., Ren, S., and Sun, J. (2015). Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE transactions on pattern analysis and machine intelligence*, 37(9):1904–1916. 29

[He et al., 2016] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778. 3, 24, 25, 26, 29, 121, XLIV

[Hénaff et al., 2021] Hénaff, O. J., Koppula, S., Alayrac, J.-B., Van den Oord, A., Vinyals, O., and Carreira, J. (2021). Efficient visual pretraining with contrastive detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 88

[Hochreiter et al., 2001] Hochreiter, S., Bengio, Y., Frasconi, P., Schmidhuber, J., et al. (2001). Gradient flow in recurrent nets: the difficulty of learning long-term dependencies. 23

[Hoffer and Ailon, 2015] Hoffer, E. and Ailon, N. (2015). Deep metric learning using triplet network. In Feragen, A., Pelillo, M., and Loog, M., editors, *Similarity-Based Pattern Recognition*, Lecture Notes in Computer Science, pages 84–92, Cham. Springer International Publishing. 63

[Hosang et al., 2017] Hosang, J., Benenson, R., and Schiele, B. (2017). Learning non-maximum suppression. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4507–4515. 110, 111

[Hospedales et al., 2021] Hospedales, T., Antoniou, A., Micaelli, P., and Storkey, A. (2021). Meta-learning in neural networks: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 44(9):5149–5169. 39, 40, XLV

[Hu et al., 2021] Hu, Q., Wang, X., Hu, W., and Qi, G. (2021). Adco: Adversarial contrast for efficient learning of unsupervised representations from self-trained negative adversaries. In *CVPR*, pages 1074–1083. Computer Vision Foundation / IEEE. 49

[Huang et al., 2021] Huang, G., Laradji, I. H., Vázquez, D., Lacoste-Julien, S., and Rodríguez, P. (2021). A survey of self-supervised and few-shot object detection. *CoRR*, abs/2110.14711. 44, 45, 50, XLV

[Inoue, 2018] Inoue, H. (2018). Data augmentation by pairing samples for images classification. *arXiv preprint arXiv:1801.02929*. 35

[Ipeirotis, 2010] Ipeirotis, P. G. (2010). Demographics of mechanical turk. XV

[Izmailov et al., 2018] Izmailov, P., Podoprikhin, D., Garipov, T., Vetrov, D. P., and Wilson, A. G. (2018). Averaging weights leads to wider optima and better generalization. In *UAI*, pages 876–885. AUAI Press. 54

[Jamal and Qi, 2019] Jamal, M. A. and Qi, G.-J. (2019). Task agnostic meta-learning for few-shot learning. In *CVPR*. 73

[Jarrett et al., 2009] Jarrett, K., Kavukcuoglu, K., Ranzato, M., and LeCun, Y. (2009). What is the best multi-stage architecture for object recognition? In *2009 IEEE 12th international conference on computer vision*, pages 2146–2153. IEEE. 20

[Jeong et al., 2019] Jeong, J., Lee, S., Kim, J., and Kwak, N. (2019). Consistency-based semi-supervised learning for object detection. *Advances in neural information processing systems*, 32. 54, 110, 112, 113, 120

[Jiao et al., 2019] Jiao, L., Zhang, F., Liu, F., Yang, S., Li, L., Feng, Z., and Qu, R. (2019). A survey of deep learning-based object detection. *IEEE access*, 7:128837–128868. 29, 30, XLIV

[Kaiser et al., 2017] Kaiser, L., Nachum, O., Roy, A., and Bengio, S. (2017). Learning to remember rare events. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. 41

[Kang et al., 2019] Kang, B., Liu, Z., Wang, X., Yu, F., Feng, J., and Darrell, T. (2019). Few-shot object detection via feature reweighting. In *ICCV*, pages 8419–8428. IEEE. 44, 45, 46, XLV

[Karlinsky et al., 2019] Karlinsky, L., Shtok, J., Harary, S., Schwartz, E., Aides, A., Feris, R. S., Giryes, R., and Bronstein, A. M. (2019). Repmet: Representative-based metric learning for classification and few-shot object detection. In *CVPR*, pages 5197–5206. Computer Vision Foundation / IEEE. 45, XLV

[Kaufman, 2013] Kaufman, J. (2013). *The First 20 Hours: How to Learn Anything...Fast*. Portfolio. 1

[Ke et al., 2019] Ke, Z., Wang, D., Yan, Q., Ren, J. S. J., and Lau, R. W. H. (2019). Dual student: Breaking the limits of the teacher in semi-supervised learning. In *ICCV*, pages 6727–6735. IEEE. 54

[Khodak et al., 2019] Khodak, M., Balcan, M., and Talwalkar, A. S. (2019). Adaptive gradient-based meta-learning methods. In *Advances in Neural Information Processing Systems*. 59

[Kingma and Ba, 2015] Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization. In *ICLR*. 22, 75

[Koch et al., 2015] Koch, G., Zemel, R., and Salakhutdinov, R. (2015). Siamese neural networks for one-shot image recognition. In *International Conference on Machine Learning deep learning workshop*. 41, 44, 59, 63

[Kornblith et al., 2019] Kornblith, S., Shlens, J., and Le, Q. V. (2019). Do better imagenet models transfer better? In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2661–2671. 28

[Krizhevsky, 2009] Krizhevsky, A. (2009). Learning multiple layers of features from tiny images. Technical report, University of Toronto. 2, 25, 28, 43, XLIV

[Krizhevsky et al., 2017] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2017). Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90. 24

[Krogh and Hertz, 1992] Krogh, A. and Hertz, J. A. (1992). A simple weight decay can improve generalization. In *Advances in Neural Information Processing Systems*. 17, 73

[Kuzborskij and Orabona, 2017] Kuzborskij, I. and Orabona, F. (2017). Fast rates by transferring from auxiliary hypotheses. *Machine Learning*, 106(2):171–195. 73

[Laine and Aila, 2017] Laine, S. and Aila, T. (2017). Temporal ensembling for semi-supervised learning. In *International Conference on Learning Representations, ICLR*. 54

[Lake et al., 2015] Lake, B. M., Salakhutdinov, R., and Tenenbaum, J. B. (2015). Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338. 43, 65, 74

[Law and Deng, 2018] Law, H. and Deng, J. (2018). Cornernet: Detecting objects as paired keypoints. In *Proceedings of the European conference on computer vision (ECCV)*, pages 734–750. 29, 31

[LeCun et al., 2015] LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *nature*, 521(7553):436–444. 2, 3

[LeCun et al., 1989] LeCun, Y., Boser, B., Denker, J., Henderson, D., Howard, R., Hubbard, W., and Jackel, L. (1989). Handwritten digit recognition with a back-propagation network. *Advances in neural information processing systems*, 2. 3, 23

[LeCun et al., 1998] LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324. 28

[Lee et al., 2013] Lee, D.-H. et al. (2013). Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on challenges in representation learning, ICML*, volume 3, page 896. 54

[Lee et al., 2019] Lee, K., Maji, S., Ravichandran, A., and Soatto, S. (2019). Meta-learning with differentiable convex optimization. In *CVPR*. 39, 40, 59, 63, 64, 76, 82

[Lewis and Sendov, 2005] Lewis, A. S. and Sendov, H. S. (2005). Nonsmooth Analysis of Singular Values. Part I: Theory. *Set-Valued Analysis*, 13(3):213–241. 72

[Li and Li, 2021] Li, A. and Li, Z. (2021). Transformation invariant few-shot object detection. In *CVPR*, pages 3094–3102. Computer Vision Foundation / IEEE. 45

[Li et al., 2021] Li, C.-L., Sohn, K., Yoon, J., and Pfister, T. (2021). Cutpaste: Self-supervised learning for anomaly detection and localization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9664–9674. 35

[Li et al., 2022] Li, F., Zhang, H., Liu, S., Guo, J., Ni, L. M., and Zhang, L. (2022). Dn-detr: Accelerate detr training by introducing query denoising. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 31, 90, 103

[Li et al., 2019] Li, H., Eigen, D., Dodge, S., Zeiler, M., and Wang, X. (2019). Finding Task-Relevant Features for Few-Shot Learning by Category Traversal. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–10. ISSN: 2575-7075. 59, 63, 82

[Li et al., 2020] Li, J., Socher, R., and Hoi, S. C. H. (2020). Dividemix: Learning with noisy labels as semi-supervised learning. In *ICLR*. OpenReview.net. 54

[Li et al., 2017] Li, Z., Zhou, F., Chen, F., and Li, H. (2017). Meta-SGD: Learning to Learn Quickly for Few-Shot Learning. *arXiv:1707.09835 [cs]*. arXiv: 1707.09835. 39, 59, 63, 82

[Lifchitz et al., 2019] Lifchitz, Y., Avrithis, Y., Picard, S., and Bursuc, A. (2019). Dense classification and implanting for few-shot learning. In *CVPR*, pages 9258–9267. Computer Vision Foundation / IEEE. 48

[Lillicrap et al., 2015] Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. (2015). Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*. 112

[Lin et al., 2017a] Lin, T.-Y., Dollár, P., Girshick, R., He, K., Hariharan, B., and Belongie, S. (2017a). Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125. 29, 90, 111, 112, 114, 125, XLVIII, LI

[Lin et al., 2017b] Lin, T.-Y., Goyal, P., Girshick, R., He, K., and Dollár, P. (2017b). Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988. 29, 111, 116

[Lin et al., 2014] Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. (2014). Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer. 4, 5, 29, 46, 97, 120, 127, 132, XII, XV, XLIII

[Liu et al., 2019] Liu, H., Simonyan, K., and Yang, Y. (2019). DARTS: Differentiable architecture search. In *International Conference on Learning Representations*. 39

[Liu et al., 2022a] Liu, S., Li, F., Zhang, H., Yang, X., Qi, X., Su, H., Zhu, J., and Zhang, L. (2022a). DAB-DETR: Dynamic anchor boxes are better queries for DETR. In *International Conference on Learning Representations*. 31, 90, 99, 102, LI

[Liu et al., 2016] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., and Berg, A. C. (2016). Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer. 29, 31, 90, 111

[Liu et al., 2022b] Liu, Y., Ma, C., and Kira, Z. (2022b). Unbiased teacher v2: Semi-supervised object detection for anchor-free and anchor-based detectors. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*, pages 9809–9818. IEEE. 130

[Liu et al., 2021] Liu, Y.-C., Ma, C.-Y., He, Z., Kuo, C.-W., Chen, K., Zhang, P., Wu, B., Kira, Z., and Vajda, P. (2021). Unbiased teacher for semi-supervised object detection. In *ICLR*. 55, 56, 91, 110, 112, 113, 114, 119, 120, 121, 122, 123, 124, 125, 126, XLVI, XLVIII, LI, LII

[Loshchilov and Hutter, 2019] Loshchilov, I. and Hutter, F. (2019). Decoupled weight decay regularization. In *International Conference on Learning Representations*. 22

[Lowe, 2004] Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110. 29

[Luccioni et al., 2022] Luccioni, A. S., Viguier, S., and Ligozat, A. (2022). Estimating the carbon footprint of bloom, a 176b parameter language model. *CoRR*, abs/2211.02001. 131

[Maurer et al., 2016] Maurer, A., Pontil, M., and Romera-Paredes, B. (2016). The benefit of multitask representation learning. *Journal of Machine Learning Research*, 17:81:1–81:32. 59, 61, I, II

[Meng et al., 2021] Meng, D., Chen, X., Fan, Z., Zeng, G., Li, H., Yuan, Y., Sun, L., and Wang, J. (2021). Conditional detr for fast training convergence. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 31, 90, 99, 102, LI

[Mishra et al., 2018] Mishra, N., Rohaninejad, M., Chen, X., and Abbeel, P. (2018). A simple neural attentive meta-learner. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. 41

[Misra and Maaten, 2020] Misra, I. and Maaten, L. v. d. (2020). Self-supervised learning of pretext-invariant representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 49

[Miyato et al., 2018a] Miyato, T., Kataoka, T., Koyama, M., and Yoshida, Y. (2018a). Spectral normalization for generative adversarial networks. In *ICLR*. 73

[Miyato et al., 2018b] Miyato, T., Maeda, S.-i., Koyama, M., and Ishii, S. (2018b). Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *IEEE transactions on pattern analysis and machine intelligence*, 41(8):1979–1993. 54

[Mohanty et al., 2016] Mohanty, S. P., Hughes, D. P., and Salathé, M. (2016). Using deep learning for image-based plant disease detection. *Frontiers in Plant Science*, page 1419. 80

[Mohri et al., 2018] Mohri, M., Rostamizadeh, A., and Talwalkar, A. (2018). *Foundations of machine learning*. MIT press. 3, 16

[Munkhdalai and Yu, 2017] Munkhdalai, T. and Yu, H. (2017). Meta networks. In *International Conference on Machine Learning*, pages 2554–2563. PMLR. 41

[Munkres, 1957] Munkres, J. (1957). Algorithms for the assignment and transportation problems. *Journal of the society for industrial and applied mathematics*, 5(1):32–38. 31, 93, 116

[Nakamura and Harada, 2019] Nakamura, A. and Harada, T. (2019). Revisiting fine-tuning for few-shot learning. *CoRR*, abs/1910.00216. 48

[Newell and Deng, 2020] Newell, A. and Deng, J. (2020). How useful is self-supervised pretraining for visual tasks? In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 7343–7352. Computer Vision Foundation / IEEE. 129, 131

[Nichol et al., 2018] Nichol, A., Achiam, J., and Schulman, J. (2018). On first-order meta-learning algorithms. *arXiv:1803.02999 [cs]*. 59, 63

[Noroozi and Favaro, 2016] Noroozi, M. and Favaro, P. (2016). Unsupervised learning of visual representations by solving jigsaw puzzles. In *European conference on computer vision*. Springer. 48

[Noroozi et al., 2017] Noroozi, M., Pirsiavash, H., and Favaro, P. (2017). Representation learning by learning to count. In *ICCV*, pages 5899–5907. IEEE Computer Society. 48

[O Pinheiro et al., 2020] O Pinheiro, P. O., Almahairi, A., Benmalek, R., Golemo, F., and Courville, A. C. (2020). Unsupervised learning of dense visual representations. *Advances in Neural Information Processing Systems*. 50

[Oh et al., 2021] Oh, J., Yoo, H., Kim, C., and Yun, S. (2021). BOIL: towards representation change for few-shot learning. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. 39

[Oliver et al., 2018] Oliver, A., Odena, A., Raffel, C., Cubuk, E. D., and Goodfellow, I. J. (2018). Realistic evaluation of deep semi-supervised learning algorithms. In *NeurIPS*, pages 3239–3250. 53

[Oord et al., 2018] Oord, A. v. d., Li, Y., and Vinyals, O. (2018). Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*. 49, 95

[Oreshkin et al., 2018] Oreshkin, B., Rodríguez López, P., and Lacoste, A. (2018). TADAM: Task dependent adaptive metric for improved few-shot learning. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R., editors, *Advances in neural information processing systems*, volume 31. Curran Associates, Inc. 59, 63, 82

[Pan and Yang, 2009] Pan, S. J. and Yang, Q. (2009). A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359. 47

[Park and Oliva, 2019] Park, E. and Oliva, J. B. (2019). Meta-curvature. In Wallach, H. M., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E. B., and Garnett, R., editors, *NeurIPS*, pages 3309–3319. 59, 63, 64, 82, IV, V, XLIX

[Park et al., 2018] Park, S., Park, J., Shin, S., and Moon, I. (2018). Adversarial dropout for supervised and semi-supervised learning. In *AAAI*, pages 3917–3924. AAAI Press. 54

[Pentina and Lampert, 2014] Pentina, A. and Lampert, C. H. (2014). A PAC-Bayesian bound for lifelong learning. In *International Conference on Machine Learning*. 59, 61, 73

[Pezeshki et al., 2016] Pezeshki, M., Fan, L., Brakel, P., Courville, A. C., and Bengio, Y. (2016). Deconstructing the ladder network architecture. In *ICML*, volume 48 of *JMLR Workshop and Conference Proceedings*, pages 2368–2376. JMLR.org. 54

[Pham et al., 2021] Pham, H., Dai, Z., Xie, Q., and Le, Q. V. (2021). Meta pseudo labels. In *CVPR*, pages 11557–11568. Computer Vision Foundation / IEEE. 54

[Qi et al., 2018] Qi, H., Brown, M., and Lowe, D. G. (2018). Low-Shot Learning with Imprinted Weights. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5822–5830, Salt Lake City, UT. IEEE. 73

[Qiao et al., 2021] Qiao, L., Zhao, Y., Li, Z., Qiu, X., Wu, J., and Zhang, C. (2021). Defrcn: Decoupled faster R-CNN for few-shot object detection. In *ICCV*, pages 8661–8670. IEEE. 45

[Qiao et al., 2018a] Qiao, S., Liu, C., Shen, W., and Yuille, A. L. (2018a). Few-shot image recognition by predicting parameters from activations. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 7229–7238. 39, 41, 76

[Qiao et al., 2018b] Qiao, S., Shen, W., Zhang, Z., Wang, B., and Yuille, A. L. (2018b). Deep co-training for semi-supervised image recognition. In *ECCV (15)*, volume 11219 of *Lecture Notes in Computer Science*, pages 142–159. Springer. 54

[Raghu et al., 2020] Raghu, A., Raghu, M., Bengio, S., and Vinyals, O. (2020). Rapid learning or feature reuse? Towards understanding the effectiveness of MAML. In *ICLR*. 39, 47, 56, 59, 63, 64, 72, 73, 82

[Rasmus et al., 2015] Rasmus, A., Berglund, M., Honkala, M., Valpola, H., and Raiko, T. (2015). Semi-supervised learning with ladder networks. In *NIPS*, pages 3546–3554. 54

[Ravi and Larochelle, 2017] Ravi, S. and Larochelle, H. (2017). Optimization as a model for few-shot learning. In *ICLR*. 39, 40, 43, 47, 59, 63, 65, 75

[Redmon et al., 2016] Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *Proceedings of the IEEE* *conference on computer vision and pattern recognition*, pages 779–788. 29, 31, 90, 111

[Redmon and Farhadi, 2017] Redmon, J. and Farhadi, A. (2017). Yolo9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7263–7271. 29, 31

[Reed et al., 2022] Reed, C. J., Yue, X., Nrusimha, A., Ebrahimi, S., Vijaykumar, V., Mao, R., Li, B., Zhang, S., Guillory, D., Metzger, S., et al. (2022). Self-supervised pretraining improves self-supervised pretraining. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 52, 129

[Ren et al., 2018] Ren, M., Triantafillou, E., Ravi, S., Snell, J., Swersky, K., Tenenbaum, J. B., Larochelle, H., and Zemel, R. S. (2018). Meta-learning for semi-supervised few-shot classification. In *ICLR*. 43, 65, 75

[Ren et al., 2015] Ren, S., He, K., Girshick, R., and Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28. 28, 29, 50, 90, 110, 111, 112, 114, 124, 125, 129, XLVIII, LI

[Rezatofighi et al., 2019] Rezatofighi, H., Tsoi, N., Gwak, J., Sadeghian, A., Reid, I., and Savarese, S. (2019). Generalized intersection over union: A metric and a loss for bounding box regression. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 658–666. 94, 116

[Ridnik et al., 2021] Ridnik, T., Ben-Baruch, E., Noy, A., and Zelnik, L. (2021). Imagenet-21k pretraining for the masses. In Vanschoren, J. and Yeung, S., editors, *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1. 28

[Roh et al., 2021] Roh, B., Shin, W., Kim, I., and Kim, S. (2021). Spatially consistent representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 50, 98, 100, 101

[Rosenblatt, 1958] Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386. 19

[Rumelhart et al., 1986] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *nature*, 323(6088):533–536. 23

[Russakovsky et al., 2015] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. (2015). ImageNet large scale visual recognition challenge. *International Journal of Computer Vision*. 4, 28, 43, 49, 75, 97, 121, 127, 131, XII, XV

[Rusu et al., 2018] Rusu, A. A., Rao, D., Sygnowski, J., Vinyals, O., Pascanu, R., Osindero, S., and Hadsell, R. (2018). Meta-learning with latent embedding optimization. In *International Conference on Learning Representations*. 39, 48, 76

[Sajjadi et al., 2016] Sajjadi, M., Javanmardi, M., and Tasdizen, T. (2016). Regularization with stochastic transformations and perturbations for deep semi-supervised learning. In *NIPS*, pages 1163–1171. 54

[Santoro et al., 2016] Santoro, A., Bartunov, S., Botvinick, M., Wierstra, D., and Lillicrap, T. (2016). Meta-learning with memory-augmented neural networks. In Balcan, M. F. and Weinberger, K. Q., editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1842–1850, New York, New York, USA. PMLR. 39, 41

[Satorras and Estrach, 2018] Satorras, V. G. and Estrach, J. B. (2018). Few-shot learning with graph neural networks. In *International Conference on Learning Representations*. 39

[Schwartz et al., 2018] Schwartz, E., Karlinsky, L., Shtok, J., Harary, S., Marder, M., Kumar, A., Feris, R., Giryes, R., and Bronstein, A. (2018). Delta-encoder: an effective sample synthesis method for few-shot object recognition. *Advances in neural information processing systems*, 31. 37

[Shalev-Shwartz and Ben-David, 2014] Shalev-Shwartz, S. and Ben-David, S. (2014). *Understanding machine learning: From theory to algorithms*. Cambridge university press. 3, 16

[Shen et al., 2021] Shen, Z., Liu, Z., Qin, J., Savvides, M., and Cheng, K. (2021). Partial is better than all: Revisiting fine-tuning strategy for few-shot learning. In *AAAI*, pages 9594–9602. AAAI Press. 48

[Shorten and Khoshgoftaar, 2019] Shorten, C. and Khoshgoftaar, T. M. (2019). A survey on image data augmentation for deep learning. *Journal of big data*, 6(1):1–48. 34, 35, 37, XLIV

[Simon et al., 2020] Simon, C., Koniusz, P., Nock, R., and Harandi, M. (2020). Adaptive Subspaces for Few-Shot Learning. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4135–4144, Seattle, WA, USA. IEEE. 59, 63, 82

[Snell et al., 2017] Snell, J., Swersky, K., and Zemel, R. S. (2017). Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems*. 39, 41, 42, 59, 63, 65, 82, III, XLV

[Sohn et al., 2020a] Sohn, K., Berthelot, D., Carlini, N., Zhang, Z., Zhang, H., Raffel, C. A., Cubuk, E. D., Kurakin, A., and Li, C.-L. (2020a). Fixmatch: Simplifying semi-supervised learning with consistency and confidence. *Advances in neural information processing systems*, 33:596–608. 54, 55, 112, 114, 124, 129, XLVI

[Sohn et al., 2020b] Sohn, K., Zhang, Z., Li, C.-L., Zhang, H., Lee, C.-Y., and Pfister, T. (2020b). A simple semi-supervised learning framework for object detection. *arXiv preprint arXiv:2005.04757*. 56, 110, 112, 113, 120, 121, 122

[Summers and Dinneen, 2019] Summers, C. and Dinneen, M. J. (2019). Improved mixed-example data augmentation. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1262–1270. IEEE. 35, 36, XLIV

[Sung et al., 2018] Sung, F., Yang, Y., Zhang, L., Xiang, T., Torr, P. H., and Hospedales, T. M. (2018). Learning to compare: Relation network for few-shot learning. In *CVPR*, pages 1199–1208. 41, 59, 63, 82

[Takahashi et al., 2019] Takahashi, R., Matsubara, T., and Uehara, K. (2019). Data augmentation using random image cropping and patching for deep cnns. *IEEE Transactions on Circuits and Systems for Video Technology*, 30(9):2917–2931. 35, 36, XLIV

[Tang et al., 2021] Tang, Y., Chen, W., Luo, Y., and Zhang, Y. (2021). Humble teachers teach better students for semi-supervised object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3132–3141. 56, 110, 112, 113, 120, 121, 123, 124

[Tang et al., 2016] Tang, Y., Wang, J., Gao, B., Dellandréa, E., Gaizauskas, R. J., and Chen, L. (2016). Large scale semi-supervised object detection using visual and semantic knowledge transfer. In *CVPR*, pages 2119–2128. IEEE Computer Society. 54

[Tarvainen and Valpola, 2017] Tarvainen, A. and Valpola, H. (2017). Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. *Advances in neural information processing systems*, 30. 54, 55, 114, XLVI

[Taylor and Nitschke, 2018] Taylor, L. and Nitschke, G. (2018). Improving deep learning with generic data augmentation. In *2018 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1542–1547. IEEE. 35

[Thrun and Pratt, 1998] Thrun, S. and Pratt, L. (1998). Learning to learn: Introduction and overview. In *Learning to learn*, pages 3–17. Springer. 38

[Tian et al., 2020a] Tian, Y., Krishnan, D., and Isola, P. (2020a). Contrastive multiview coding. In *ECCV (11)*, volume 12356 of *Lecture Notes in Computer Science*, pages 776–794. Springer. 48

[Tian et al., 2020b] Tian, Y., Sun, C., Poole, B., Krishnan, D., Schmid, C., and Isola, P. (2020b). What makes for good views for contrastive learning? In *NeurIPS*. 49

[Tian et al., 2020c] Tian, Y., Wang, Y., Krishnan, D., Tenenbaum, J. B., and Isola, P. (2020c). 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*, pages 266–282. Springer. 44, 48, 73, 81, 129

[Tian et al., 2019] Tian, Z., Shen, C., Chen, H., and He, T. (2019). Fcos: Fully convolutional one-stage object detection. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9627–9636. 29, 31, 90, 111

[Tokozume et al., 2018] Tokozume, Y., Ushiku, Y., and Harada, T. (2018). Between-class learning for image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5486–5494. 35

[Triantafillou et al., 2020] Triantafillou, E., Zhu, T., Dumoulin, V., Lamblin, P., Evci, U., Xu, K., Goroshin, R., Gelada, C., Swersky, K., Manzagol, P., and Larochelle, H. (2020). Meta-dataset: A dataset of datasets for learning to learn from few examples. In *ICLR*. OpenReview.net. 43, XLV

[Tripuraneni et al., 2020] Tripuraneni, N., Jin, C., and Jordan, M. I. (2020). Provable meta-learning of linear representations. In *arXiv:2002.11684*. 56, 59, 62, I, II

[Tseng et al., 2020] Tseng, H.-Y., Lee, H.-Y., Huang, J.-B., and Yang, M.-H. (2020). Cross-domain few-shot classification via learned feature-wise transformation. In *International Conference on Learning Representations*. 37, XLV

[Uijlings et al., 2013] Uijlings, J. R., Van De Sande, K. E., Gevers, T., and Smeulders, A. W. (2013). Selective search for object recognition. *International journal of computer vision*. 29, 51, 89, 91, 93, 115, XLVI

[van Engelen and Hoos, 2020] van Engelen, J. E. and Hoos, H. H. (2020). A survey on semi-supervised learning. *Mach. Learn.*, 109(2):373–440. 53

[Vapnik, 1999] Vapnik, V. (1999). *The nature of statistical learning theory*. Springer science & business media. 8, 14, 15, 16

[Vapnik and Chervonenkis, 2015] Vapnik, V. N. and Chervonenkis, A. Y. (2015). On the uniform convergence of relative frequencies of events to their probabilities. In *Measures of complexity*, pages 11–30. Springer. 16

[Vaswani et al., 2017] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30. 3, 24, 25, 26, 27, 88, 90, 110, XLIV

[Verma et al., 2019] Verma, V., Lamb, A., Kannala, J., Bengio, Y., and Lopez-Paz, D. (2019). Interpolation consistency training for semi-supervised learning. In *IJCAI*, pages 3635–3641. ijcai.org. 54

[Vinyals et al., 2016] Vinyals, O., Blundell, C., Lillicrap, T., Kavukcuoglu, K., and Wierstra, D. (2016). Matching networks for one shot learning. In *Advances in Neural Information Processing Systems*, pages 3630–3638. 39, 41, 44, 47, 59, 63

[Wang et al., 2021a] Wang, G., Wang, K., Wang, G., Torr, P. H. S., and Lin, L. (2021a). Solving inefficiency of self-supervised representation learning. In *ICCV*, pages 9485–9495. IEEE. 49

[Wang et al., 2021b] Wang, H., Zhao, H., and Li, B. (2021b). Bridging multi-task learning and meta-learning: Towards efficient training and effective adaptation. In Meila, M. and Zhang, T., editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 10991–11002. PMLR. 56, 59, 60, 64, 65, 73, 79, 84, 85, L

[Wang et al., 2018a] Wang, K., Yan, X., Zhang, D., Zhang, L., and Lin, L. (2018a). Towards human-machine cooperation: Self-supervised sample mining for object detection. In *CVPR*, pages 1605–1613. Computer Vision Foundation / IEEE Computer Society. 54

[Wang and Isola, 2020] Wang, T. and Isola, P. (2020). Understanding contrastive representation learning through alignment and uniformity on the hypersphere. In *ICML*, Proceedings of machine learning research. 49, 67, 79, VIII

[Wang et al., 2020a] Wang, X., Huang, T. E., Gonzalez, J., Darrell, T., and Yu, F. (2020a). Frustratingly simple few-shot object detection. In *ICML*, volume 119 of *Proceedings of Machine Learning Research*, pages 9919–9928. PMLR. 47, 52, 56, 81, 129, XLVI

[Wang et al., 2021c] Wang, X., Kihara, D., Luo, J., and Qi, G. (2021c). Enaet: A self-trained framework for semi-supervised and supervised learning with ensemble transformations. *IEEE Trans. Image Process.*, 30:1639–1647. 54

[Wang et al., 2021d] Wang, X., Zhang, R., Shen, C., Kong, T., and Li, L. (2021d). Dense contrastive learning for self-supervised visual pre-training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 50, 52, 88

[Wang et al., 2019] Wang, Y., Chao, W.-L., Weinberger, K. Q., and van der Maaten, L. (2019). SimpleShot: Revisiting Nearest-Neighbor Classification for Few-Shot Learning. *arXiv:1911.04623 [cs]*. arXiv: 1911.04623. 48, 73, 82

[Wang et al., 2020b] Wang, Y., Yao, Q., Kwok, J. T., and Ni, L. M. (2020b). Generalizing from a few examples: A survey on few-shot learning. *ACM computing surveys (csur)*, 53(3):1–34. 34, 39, 41, XLV

[Wang et al., 2021e] Wang, Y., Zhang, X., Yang, T., and Sun, J. (2021e). Anchor detr: Query design for transformer-based object detection. *arXiv preprint arXiv:2109.07107*. 31, 90

[Wang et al., 2018b] Wang, Y.-X., Girshick, R., Hebert, M., and Hariharan, B. (2018b). Low-Shot Learning from Imaginary Data. In *CVPR*, pages 7278–7286. 37

[Wei et al., 2021a] Wei, C., Wang, H., Shen, W., and Yuille, A. L. (2021a). CO2: consistent contrast for unsupervised visual representation learning. In *ICLR*. OpenReview.net. 49

[Wei et al., 2021b] Wei, F., Gao, Y., Wu, Z., Hu, H., and Lin, S. (2021b). Aligning pretraining for detection via object-level contrastive learning. *Advances in Neural Information Processing Systems*. 51, 52, 88, 91, 93, 98, 101, 102, 103, 104, 107, 129, XLV

[Wilcoxon, 1945] Wilcoxon, F. (1945). Individual comparisons by ranking methods. *Biometrics Bulletin*, 1(6):80–83. 77, 78

[Wong et al., 2016] Wong, S. C., Gatt, A., Stamatescu, V., and McDonnell, M. D. (2016). Understanding data augmentation for classification: when to warp? In *2016 international conference on digital image computing: techniques and applications (DICTA)*, pages 1–6. IEEE. 37

[Wu et al., 2020a] Wu, J., Liu, S., Huang, D., and Wang, Y. (2020a). Multi-scale positive sample refinement for few-shot object detection. In *ECCV (16)*, volume 12361 of *Lecture Notes in Computer Science*, pages 456–472. Springer. 52

[Wu et al., 2020b] Wu, X., Sahoo, D., and Hoi, S. C. H. (2020b). Meta-rcnn: Meta learning for few-shot object detection. In *ACM Multimedia*, pages 1679–1687. ACM. 45

[Wu et al., 2018] Wu, Z., Xiong, Y., Yu, S. X., and Lin, D. (2018). Unsupervised feature learning via non-parametric instance discrimination. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 49

[Xiao et al., 2021] Xiao, T., Reed, C. J., Wang, X., Keutzer, K., and Darrell, T. (2021). Region similarity representation learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 50, 88

[Xiao and Marlet, 2020] Xiao, Y. and Marlet, R. (2020). Few-shot object detection and viewpoint estimation for objects in the wild. In *ECCV (17)*, volume 12362 of *Lecture Notes in Computer Science*, pages 192–210. Springer. 45

[Xie et al., 2020a] Xie, Q., Dai, Z., Hovy, E. H., Luong, T., and Le, Q. (2020a). Unsupervised data augmentation for consistency training. In *NeurIPS*. 54

[Xie et al., 2020b] Xie, Q., Luong, M., Hovy, E. H., and Le, Q. V. (2020b). Self-training with noisy student improves imagenet classification. In *CVPR*, pages 10684–10695. Computer Vision Foundation / IEEE. 54

[Xie et al., 2021] Xie, Z., Lin, Y., Zhang, Z., Cao, Y., Lin, S., and Hu, H. (2021). Propagate yourself: Exploring pixel-level consistency for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 50, 88

[Xu et al., 2021] Xu, M., Zhang, Z., Hu, H., Wang, J., Wang, L., Wei, F., Bai, X., and Liu, Z. (2021). End-to-end semi-supervised object detection with soft teacher. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3060–3069. 56, 110, 112, 113, 119, 120, 122, 123, 124, 126, LII

[Yaeger et al., 1996] Yaeger, L., Lyon, R., and Webb, B. (1996). Effective training of a neural network character classifier for word recognition. *Advances in neural information processing systems*, 9. 34

[Yan et al., 2019] Yan, X., Chen, Z., Xu, A., Wang, X., Liang, X., and Lin, L. (2019). Meta R-CNN: towards general solver for instance-level low-shot learning. In *ICCV*, pages 9576–9585. IEEE. 45

[Yang et al., 2022] Yang, C., Huang, Z., and Wang, N. (2022). Querydet: Cascaded sparse query for accelerating high-resolution small object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 31, 90

[Yang et al., 2021a] Yang, C., Wu, Z., Zhou, B., and Lin, S. (2021a). Instance localization for self-supervised detection pretraining. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 50, 52

[Yang et al., 2021b] Yang, S., Liu, L., and Xu, M. (2021b). Free lunch for few-shot learning: Distribution calibration. In *International Conference on Learning Representations ICLR*. OpenReview.net. 81

[Yang et al., 2021c] Yang, X., Song, Z., King, I., and Xu, Z. (2021c). A survey on deep semi-supervised learning. *CoRR*, abs/2103.00550. 53, XLVI

[Ye and Chao, 2021] Ye, H.-J. and Chao, W.-L. (2021). How to Train Your MAML to Excel in Few-Shot Classification. *arXiv:2106.16245 [cs]*. arXiv: 2106.16245. 39, 48, 73, 76, 131

[Ye et al., 2020] Ye, H.-J., Hu, H., Zhan, D.-C., and Sha, F. (2020). Few-shot learning via embedding adaptation with set-to-set functions. In *Computer vision and pattern recognition (CVPR)*. 48, 76, 77, 131

[Yin et al., 2020] Yin, M., Tucker, G., Zhou, M., Levine, S., and Finn, C. (2020). Meta-learning without memorization. In *ICLR*. 39, 59, 61, 73

[Yun et al., 2019] Yun, S., Han, D., Oh, S. J., Chun, S., Choe, J., and Yoo, Y. (2019). Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6023–6032. 35

[Zaidi et al., 2022] Zaidi, S. S. A., Ansari, M. S., Aslam, A., Kanwal, N., Asghar, M., and Lee, B. (2022). A survey of modern deep learning based object detection models. *Digital Signal Processing*, page 103514. 29

[Zbontar et al., 2021] Zbontar, J., Jing, L., Misra, I., LeCun, Y., and Deny, S. (2021). Barlow twins: Self-supervised learning via redundancy reduction. In *ICML*, volume 139 of *Proceedings of Machine Learning Research*, pages 12310–12320. PMLR. 49

[Zhang et al., 2017] Zhang, H., Cisse, M., Dauphin, Y. N., and Lopez-Paz, D. (2017). mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*. 35

[Zhang and Qi, 2020] Zhang, L. and Qi, G. (2020). WCP: worst-case perturbations for semi-supervised deep learning. In *CVPR*, pages 3911–3920. Computer Vision Foundation / IEEE. 54

[Zhang et al., 2016] Zhang, R., Isola, P., and Efros, A. A. (2016). Colorful image colorization. In *ECCV (3)*, volume 9907 of *Lecture Notes in Computer Science*, pages 649–666. Springer. 48

[Zhang et al., 2020] Zhang, S., Chi, C., Yao, Y., Lei, Z., and Li, S. Z. (2020). Bridging the gap between anchor-based and anchor-free detection via adaptive training sample selection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9759–9768. 110

[Zheng et al., 2021] Zheng, M., You, S., Wang, F., Qian, C., Zhang, C., Wang, X., and Xu, C. (2021). Ressl: Relational self-supervised learning with weak augmentation. *Advances in Neural Information Processing Systems*. 49, 95

[Zhong et al., 2020] Zhong, Z., Zheng, L., Kang, G., Li, S., and Yang, Y. (2020). Random erasing data augmentation. In *Proceedings of the AAAI conference on artificial intelligence*, pages 13001–13008. 35, 36, XLIV

[Zhou et al., 2021] Zhou, Q., Yu, C., Wang, Z., Qian, Q., and Li, H. (2021). Instant-teaching: An end-to-end semi-supervised object detection framework. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4081–4090. 110, 112, 113, 120, 121

[Zhou et al., 2019] Zhou, X., Wang, D., and Krähenbühl, P. (2019). Objects as points. *arXiv preprint arXiv:1904.07850*. 29, 31, 111

[Zhou and Li, 2010] Zhou, Z. and Li, M. (2010). Semi-supervised learning by disagreement. *Knowl. Inf. Syst.*, 24(3):415–439. 54

[Zhu et al., 2021] Zhu, X., Su, W., Lu, L., Li, B., Wang, X., and Dai, J. (2021). Deformable detr: Deformable transformers for end-to-end object detection. In *International Conference on Learning Representations*. 31, 89, 90, 92, 98, 99, 110, 112, 113, 114, 116, 119, 121, 122, 125, 126, XLVIII, LI, LII

List of Figures

List of Tables