université PARIS-SACLAY

Learning with Limited Labeled Data

Apprentissage avec peu de données étiquetées

Thèse de doctorat de l'université Paris-Saclay

École doctorale n°573 : interfaces : matériaux, systèmes, usages (INTERFACES)
Spécialité de doctorat : INFORMATIQUE
Graduate School : Sciences de l'ingénierie et des systèmes
Référent : CentraleSupélec

Thèse préparée dans l'unité de recherche MICS (Université Paris-Saclay, CentraleSupélec), sous la direction de Céline HUDELOT, Professeure des Universités, et le co-encadrement de Myriam TAMI, Maître de conférences.

Thèse soutenue à Paris-Saclay, le JJ Avril 2023, par

Yassine OUALI

Composition du jury

Ismail Ben Ayed Professeur des Universités, ÉTS Montréal (Canada)	Rapporteur & Examinateur
Matthieu Cord Professeur des Universités, Sorbonne Université	Rapporteur & Examinateur
Vincent Lepetit Professeur des Universités, École des Ponts ParisTech	Examinateur
Stéphane Herbin Directeur de Recherche, ONERA	Examinateur
Clément Rambour Maître de conférences, Conservatoire National des Arts et Métiers	Examinateur
Myriam Tami Maître de conférences, Université Paris-Saclay	Co-encadrante de thèse
Céline HUDELOT Professeure des Universités, Université Paris-Saclay	Directrice de thèse

THESE DE DOCTORAT
NNT : 2020UPASA001

université PARIS-SACLAY | ÉCOLE DOCTORALE Interfaces: matériaux, systèmes, usages

Titre : Apprentissage avec peu de données étiquetées
Mots clés : Apprentissage profond, Apprentissage efficace, Vision par ordinateur, Traitement du langage naturel.

Résumé : Depuis ses débuts, l'objectif de l'intelligence artificielle est de concevoir des systèmes capables d'apprendre aussi efficacement que les humains pour résoudre ou aider à résoudre des problèmes difficiles qui nécessitent une certaine forme d'intelligence humaine. La discipline a connu récemment un essor spectaculaire grâce aux réseaux de neurones profonds et ses extensions qui ont montré des performances sur tout un ensemble de tâches jusqu'alors considérées complexes. Cependant, ce paradigme dominant nécessite une grande quantité de données étiquetées, qui sont souvent coûteuses et difficiles à acquérir. Ces données peuvent également contenir des biais cachés et des erreurs d'annotation, ce qui limite l'application de tels systèmes dans de nombreux domaines. Pourtant, les humains font preuve d'une remarquable capacité à apprendre efficacement dans de nouveaux et divers contextes, en tirant en grande parti de leur expérience à s'adapter à de nouveaux cas et acquérir rapidement de nouvelles compétences. Cette divergence soulève une question évidente : pouvons-nous concevoir des systèmes dotés de capacités similaires ? Dans cette thèse, notre objectif est de développer des algorithmes d'apprentissage efficaces avec une quantité limitée d'étiquettes pour résoudre de diverses tâches pour différentes modalités. A cette fin, cette thèse couvre des travaux qui : i) développent des méthodes d'apprentissage pour des paradigmes avec différents degrés de supervision, ii) présentent des résultats pour différentes modalités, notamment l'image et le texte, et iii) qui gèrent différentes tâches.

Title : Learning with Limited Labeled Data
Keywords : Deep Learning, Label Efficient Learning, Computer Vision, Natural Language Processing.

Abstract : Since its inception, the north star of artificial intelligence was to design systems capable of learning as efficiently (*i.e.* with limited training signal) and effectively (*i.e.* demonstrating good performances) as humans to solve challenging problems that require human-like intelligence. Deep neural networks and the collection of popular deep learning ingredients used to produce systems usable in the real world, such as optimization algorithms, novel architectures, objective functions, and large annotated datasets, have shown remarkable performances across various tasks in recent years. However, this dominant paradigm requires a large amount of fully labeled data, which is often expensive and difficult to acquire. It might also contain annotation errors and hidden biases, which limits the applicability and adoption of such systems. Yet humans demonstrate a remarkable ability to learn effectively across diverse settings, using limited supervision and leveraging prior experience to adapt to novel cases and gain new skills quickly. This discrepancy raises an obvious question, can we design systems with similar capabilities ? In this thesis, we aim to develop label-efficient learning algorithms that are effective with a limited or no amount of annotated examples for various tasks, over different modalities and multiple levels of abstraction. To this end, this thesis cover works that : i) develop learning methods for paradigms with varying degrees of supervision, ii) present results for different modalities, notably vision and text, and iii) different tasks across various levels of abstraction (*e.g.* image level and pixel level). We hope these works can help further advance the state of the field and aid in developing systems capable of learning efficiently and adapting effectively across a wide range of environments.

Thèse de doctorat de l'université Paris-Saclay

Learning with Limited Labeled Data

Apprentissage avec peu de données étiquetées

by

Yassine Ouali

Director: Prof. Dr. Céline Hudelot
Co-supervisor: Asst. Prof. Dr. Myriam Tami

École doctorale n°573 : interfaces : matériaux, systèmes, usages (INTERFACES)
Spécialité de doctorat : INFORMATIQUE
Graduate School : Sciences de l'ingénierie et des systèmes
Référent : CentraleSupélec

Thèse préparée dans l'unité de recherche MICS (Université Paris-Saclay, CentraleSupélec), sous la direction de Céline HUDELOT, Professeure des Universités, et le co-encadrement de Myriam TAMI, Maître de conférences.

Université Paris-Saclay, CentraleSupélec, MICS, 91190, Gif-sur-Yvette, France.

Résumé

Depuis ses débuts, l'objectif de l'intelligence artificielle est de concevoir des systèmes capables d'apprendre aussi efficacement que les humains pour résoudre ou aider à résoudre des problèmes difficiles qui nécessitent une certaine forme d'intelligence humaine. La discipline a connu récemment un essor spectaculaire grâce aux réseaux de neurones profonds et ses extensions qui ont montré des performances sur tout un ensemble de tâches jusqu'alors considérées complexes. Cependant, ce paradigme dominant nécessite une grande quantité de données étiquetées, qui sont souvent coûteuses et difficiles à acquérir. Ces données peuvent également contenir des biais cachés et des erreurs d'annotation, ce qui limite l'application de tels systèmes dans de nombreux domaines. Pourtant, les humains font preuve d'une remarquable capacité à apprendre efficacement dans de nouveaux et divers contextes, en tirant en grande parti de leur expérience à s'adapter à de nouveaux cas et acquérir rapidement de nouvelles compétences. Cette divergence soulève une question évidente : pouvons-nous concevoir des systèmes dotés de capacités similaires ? Dans cette thèse, notre objectif est de développer des algorithmes d'apprentissage efficaces avec une quantité limitée d'étiquettes pour résoudre de diverses tâches pour différentes modalités. A cette fin, cette thèse couvre des travaux qui : i) développent des méthodes d'apprentissage pour des paradigmes avec différents degrés de supervision, ii) présentent des résultats pour différentes modalités, notamment l'image et le texte, et iii) qui gèrent différentes tâches.

Abstract

Since its inception, the north star of artificial intelligence was to design systems capable of learning as efficiently (*i.e.* with limited training signal) and effectively (*i.e.* demonstrating good performances) as humans to solve challenging problems that require human-like intelligence. Deep neural networks and the collection of popular deep learning ingredients used to produce systems usable in the real world, such as optimization algorithms, novel architectures, objective functions, and large annotated datasets, have shown remarkable performances across various tasks in recent years. However, this dominant paradigm requires a large amount of fully labeled data, which is often expensive and difficult to acquire. It might also contain annotation errors and hidden biases, which limits the applicability and adoption of such systems. Yet humans demonstrate a remarkable ability to learn effectively across diverse settings, using limited supervision and leveraging prior experience to adapt to novel cases and gain new skills quickly. This discrepancy raises an obvious question, can we design systems with similar capabilities? In this thesis, we aim to develop label-efficient learning algorithms that are effective with a limited or no amount of annotated examples for various tasks, over different modalities and multiple levels of abstraction. To this end, this thesis cover works that: i) develop learning methods for paradigms with varying degrees of supervision, ii) present results for different modalities, notably vision and text, and iii) different tasks across various levels of abstraction (*e.g.* image level and pixel level). We hope these works can help further advance the state of the field and aid in developing systems capable of learning efficiently and adapting effectively across a wide range of environments.

Acknowledgements

To ...

Advisors

... Céline, so without her, this work would not exist. I am very fortunate to have met Céline during the interview process, and I am very grateful to her for allowing me to start a Ph.D. within her lab. During the interview, I was not the best candidate, nor the one with the better credentials or the preferred experience. Yet, Céline took a considerable risk on me, created a new offer and a position within her lab, and offered it to me, and for that, I will always owe her my scientific career. On top of that, Céline gave me full liberty to pursue the research that interested me, created a pleasant working environment, and was always incredibly supportive and inspiring and the source of much guidance. To say that I am grateful to her would be an understatement.

... Myriam for her consistent support, guidance, and positive spirit throughout my Ph.D. I am tremendously thankful for her much-needed help, rigor, and clarity when developing new ideas and for her invaluable advice and encouragement, which always resulted in better outcomes and contributions.

I could neither have hoped nor wished for better advisors.

Colleagues

... all my lab colleagues that made my Ph.D. journey enjoyable. I always enjoyed every chance of interaction in the coffee lounge, a curious discussion at the end of an informative presentation, and an engaging chat over the launch table. Victor B., Victor P., Thomas and Jun, and the rest of the MICS members, I express to all of you my deepest gratitude.

Administrative & technical staff

... the administrative and technical staff who make our research work possible. I am very thankful to the school's highly competent administrative staff that helped me countless times with various administrative tasks and enabled me to focus on my research topics. Fabienne, Xavière, Jana, Raphaëlle, Suzanne, Dany, Vincent, and Pascale, you made it a pleasure to be part of the MICS lab. I must also not forget the talented technical staff of Mésocentre Moulan and LabIA clusters, without whom all of the experimental results presented in this work would not have been possible.

Institutions

... every public and private institution that financed and founded, either directly or indirectly, my research and provided me with the luxury of being able to conduct scientific work while also getting paid to do so, an opportunity many students from other countries cannot afford. Specifically, I would be remiss if I did not recognize the help and assistance provided by CentraleSupélec, Université Paris-Saclay, CNRS, Région Île-de-France and Randstad corporate research chair, all of

which were essential in every facet of my Ph.D., from the working environment to administrative assistance, the financial support and the availability of computational resources.

... AND FINALLY

I am immensely grateful to my family, who have always supported me for as long as I can remember. And to all who made this work possible, I am very grateful to you for contributing to the wonderful years of my experience as a Ph.D. student.

Contents

List of Figures xi

List of Tables xi

1 Introduction 1
- 1.1 Concrete Motivations . . . 3
- 1.2 Objectives . . . 8
- 1.3 Scope . . . 9
- 1.4 Outline and Contributions . . . 12

I Part One: Foundations 17

2 A Learning Machine 19
- 2.1 Problem statement and terminology . . . 19
  - 2.1.1 The Standard Framework: Supervised Learning . . . 20
  - 2.1.2 A More Flexible Framework . . . 21
- 2.2 Scenarios and Variations . . . 22
  - 2.2.1 Generalization . . . 25
- 2.3 In Practice . . . 25

Conclusion . . . 28

3 Learning Paradigms 31
- 3.1 Supervised Learning . . . 31
  - 3.1.1 A Related Setting: Transfer Learning . . . 32
  - 3.1.2 A Related Setting: Few-Sample Fine-Tuning . . . 33
- 3.2 Semi-supervised Learning . . . 34
  - 3.2.1 Main Assumptions. . . . 35
  - 3.2.2 Dominant Approaches. . . . 35
  - 3.2.3 Consistency Training . . . 36
  - 3.2.4 Pseudo-Labeling . . . 38
  - 3.2.5 A Related Setting: Weakly-Supervised Learning . . . 39
- 3.3 Unsupervised Learning . . . 40
  - 3.3.1 Generative UL . . . 41
  - 3.3.2 Discriminative UL . . . 42
- 3.4 Unsupervised Domain Adaptation . . . 46
- 3.5 Few-shot Learning . . . 50

Conclusion . . . 53

4 Tasks, Models, and Data 55
4.1 Tasks . . . 55
4.1.1 Image Classification . . . 55
4.1.2 Image Segmentation . . . 57
4.1.3 NLP tasks . . . 59
4.2 Models . . . 66
4.2.1 Image Classification . . . 66
4.2.2 Image Segmentation . . . 67
4.2.3 NLP . . . 71
4.3 Data . . . 73
Conclusion . . . 75

II Part Two: Contributions 77

5 Cross-Consistency Training 79
5.1 Introduction . . . 80
5.2 Related Work . . . 81
5.3 Preliminaries . . . 82
5.4 Method . . . 82
5.4.1 Problem Definition . . . 82
5.4.2 Proposed Method . . . 83
5.4.3 Method Details . . . 85
5.4.4 Extensions . . . 86
5.5 Experimental Results . . . 88
5.5.1 Experimental Details . . . 88
5.5.2 Ablation Results . . . 89
5.5.3 Quantitative Results . . . 92
5.5.4 Qualitative Results . . . 93
Conclusion . . . 94

6 Autoregressive Segmentation 97
6.1 Introduction . . . 98
6.2 Related Work . . . 99
6.3 Method . . . 100
6.3.1 Problem Definition . . . 100
6.3.2 Proposed Method . . . 101
6.3.3 Method Details . . . 104
6.4 Experimental Results . . . 107
6.4.1 Experimental Details . . . 107
6.4.2 Ablation Results . . . 107
6.4.3 Quantitative Results . . . 111
6.4.4 Qualitative Results . . . 111
Conclusion . . . 112

7 Target Consistency 115
7.1 Introduction . . . 116
7.2 Related Work . . . 117
7.3 Preliminaries . . . 117
7.3.1 Problem Definition . . . 117
7.3.2 Target Sensitivity . . . 118
7.4 Method . . . 120
7.4.1 Proposed Method . . . 120
7.4.2 Extensions . . . 121
7.5 Experimental Results . . . 123
7.5.1 Experimental Details . . . 123
7.5.2 Ablation Results . . . 124
7.5.3 Quantitative Results . . . 125
7.5.4 Qualitative Results . . . 126
Conclusion . . . 127

8 Spatial Contrastive Learning 129
8.1 Introduction . . . 129
8.2 Related Work . . . 131
8.3 Preliminaries . . . 133
8.3.1 Problem Definition . . . 133
8.3.2 Transfer Learning Baseline . . . 134
8.3.3 Analysis of the Learned Representations . . . 134
8.4 Method . . . 135
8.4.1 Proposed Method . . . 135
8.4.2 Extensions . . . 138
8.5 Experimental Results . . . 139
8.5.1 Experimental Details . . . 139
8.5.2 Ablation Results . . . 140
8.5.3 Quantitative Results . . . 142
8.5.4 Cross-Domain Few-shot Classification . . . 144
8.5.5 Qualitative Results . . . 147
Conclusion . . . 147

9 Relative Bias Fine-tuning 151
9.1 Introduction . . . 152
9.2 Related Work . . . 153
9.3 Preliminaries . . . 154
9.4 Method . . . 156
9.5 Experimental Results . . . 158
9.5.1 Experimental Details . . . 159
9.5.2 Ablation Results . . . 161
9.5.3 Quantitative Results . . . 162
9.5.4 Qualitative Results . . . 163

Conclusion 163

10 Conclusion and perspectives 165

Acronyms 171

Bibliography 173

List of Figures

1.1 Electronic Health Records. . . . 4
1.2 EHRs Analysis System. . . . 5
1.3 An illustrative map of the different learning paradigms . . . 11
1.4 Thesis contributions and their corresponding chapters . . . 12

2.1 Learning paradigms . . . 24
2.2 DL Pipeline . . . 26

3.1 Effect of SSL on the learned decision boundary . . . 34
3.2 An example of an autoregressive model . . . 42
3.3 Contrastive Learning . . . 45
3.4 An example of a UDA benchmark. . . . 47
3.5 Domain-Invariant Representations . . . 48
3.6 The structure of data in FSL . . . 50
3.7 Transfer Learning based FSL methods . . . 52

4.1 Scene segmentation tasks. . . . 57
4.2 Popular NLP paradigms . . . 60
4.3 Typical training objectives for transformer-based LMs . . . 62
4.4 Replacing the fully connected layers . . . 67
4.5 A Fully Convolutional Network (FCN) . . . 68
4.6 Upsampling Layers . . . 69
4.7 Atrous Convolutions and Spatial Pyramid Pooling. . . . 70
4.8 The Building blocks of the transformer architecture . . . 71

5.1 Cross-Consistency Training (CCT) . . . 80
5.2 The Cluster Assumption in Image Segmentation . . . 83
5.3 A training Iteration of CCT . . . 84
5.4 CCT with Weak-labels . . . 87
5.5 CCT on Multiple Domains . . . 88
5.6 Ablation Studies on CamVid with 20, 50 and 100 Labeled Images . . . 89
5.7 Ablation Study on PASCAL VOC . . . 90
5.8 Qualitative Results . . . 94

6.1 Autoregressive Segmentation . . . 98
6.2 Orderings . . . 101
6.3 Masked Convolutions . . . 105
6.4 Attention Masks . . . 106

6.5 Blind Spots . . . 106
6.6 Overclustering . . . 109
6.7 Qualitative Results . . . 112

7.1 Sensitivity Analysis . . . 119
7.2 Effect of TC on a Toy Dataset . . . 120
7.3 Mixing Augmentations . . . 121
7.4 TC and Domain Discriminators . . . 122
7.5 Image Segmentation Results . . . 125
7.6 Mixing Augmentations . . . 125
7.8 Segmentation Results . . . 126
7.7 Qualitative Analyses . . . 126

8.1 Spatial Contrastive Learning . . . 130
8.2 Analysis of the Learned Representations . . . 133
8.3 Spectral Analysis . . . 134
8.4 Attention-based Spatial Alignment . . . 136
8.5 Degree of Clustering . . . 138
8.6 Hyperparameters . . . 141
8.7 Sequential Distillation . . . 143
8.8 ProtoNet Ablations . . . 145
8.9 Nearest Neighbors Analysis . . . 146
8.10 Spectral Analysis with SCL . . . 147

9.1 The Attention Operation for BitFit and RelBitFit. . . . 156
9.2 The Construction of The Content Based Relative Biases $\mathbf{R}_{i-j}$. . . . 157
9.3 Learned Biases. . . . 162

List of Tables

1.1 Thesis Contributions . . . 12
1.2 Thesis Open Source Projects . . . 13

2.1 Notation Summary . . . 22
2.2 Learning Paradigms . . . 23

5.1 CCT and Traditional Consistency Methods . . . 90
5.2 CCT with Different Distance Measures . . . 91
5.3 CCT Results with Multi-scale Inference . . . 91
5.4 Comparison with The-state-of-the-art SSL Methods . . . 92
5.5 CCT Applied to CS+CVD . . . 93
5.6 CCT Applied to CS+SUN . . . 93

6.1 AC Ablations . . . 108
6.2 Comparing ARL and AC . . . 110
6.3 Unsupervised Image Segmentation . . . 110

7.1 TC Loss Ablations. . . . 124
7.2 Adversarial Loss Ablations . . . 124
7.3 Classification Results . . . 125

8.1 Loss Function . . . 140
8.2 Data Augmentation . . . 141
8.3 Aggregation Function . . . 142
8.4 Distillation . . . 142
8.5 Features Used at Test-Time . . . 143
8.6 State-of-the-art Comparison on ImageNet Derivatives . . . 143
8.7 State-of-the-art Comparison on CIFAR Derivatives . . . 144
8.8 State-of-the-art Comparison on Cross-Domain Few-Shot Benchmarks . . . 145
8.9 ProtoNet Experiments on *mini*-ImageNet . . . 145

9.1 GLUE Benchmark . . . 159
9.2 GLUE Evaluation Metrics. . . . 159
9.3 Number of Trainable Paramters. . . . 160
9.4 Parameter Reduction. . . . 160
9.5 Comparaison with Relevant Works on Low-label Settings. . . . 161
9.6 A Further Reduction of Labeled Training Data. . . . 161

CHAPTER 1

Introduction

 *

*All chapter covers were generated using DALL·E 2

1 Introduction

Constraints are not restraints

At every design and creation endeavor, true abundance, a state in which we have access to an unlimited or large amount of the desired resource, is only encountered in the theoretical realm, and it is often used to identify an unattainable performance upper bound of a given system we wish to converge to under ideal conditions. However, the empirical realm is filled with various limitations and constraints, rendering such idealistic systems impractical[1]. As a result, such restrictions must be incorporated into a given system's design to make it practically feasible. For human-made systems, concepts, or structures, these constraints might first appear as either self-imposed conditions or arise by virtue of the system itself and its physical limitations. They nevertheless can be used as an incentive to stifle creativity and design better overall systems.

The engineering discipline as a whole can also be seen from this lens; the creative application of scientific principles under a given set of constraints to design systems, structures, and machines for some desired application or use case. Similarly, when it comes to the field of computer science, and as Frederick Brooks points out in quot.1, we find that different types of restrictions and limitations dictate the design of many systems at different levels of the hardware and software stack: from the transistors that constitute the building block of computers that are constrained by physical limitations, to everyday consumer-facing applications where we encounter different types of constraints, be it economical, societal, or time-related limitations.

> *"Computer architecture, like other architecture, is the art of determining the needs of the user of a structure and then designing to meet those needs as effectively as possible within economic and technological constraints. Architecture must include engineering considerations, so that the design will be economical and feasible."* — Frederick Brooks[2]. 1

Following along such footsteps, this thesis, and the contributions therein, can be viewed as our attempt to build efficient yet effective learning methods under the constraint of data with limited labels. For each one of the settings and tasks we will tackle in this work, we will try to view this constraint not as a hindrance or a restraint, but as an opportunity to design better and enhanced systems.

[1] For instance, while the laws of physics deal largely with ideal conditions, the reality is messy. The laws that hold in a perfect vacuum or frictionless environments will not be useful for real-world applications that deal with settings full of friction and other suboptimal conditions.

[2] Frederick Brooks is a computer architect, software engineer, and computer scientist known for developing IBM's System/360.

AI, and the rise of DL

As a subfield of computer science, the goal of Artificial Intelligence (AI) is to develop machines and systems that exhibit human-like intelligence. However, this ambitious goal has proven to be highly challenging and remains far from being reached to this day. After the initial enthusiasm for AI, the field experienced two periods of skepticism, known as "AI winters," in the 1970s and 1990s. These periods were mainly due to the mismatch between the expectations and promises of AI and what the early methods were able to deliver in terms of performance and results. These setbacks showed that the discipline was still in its infancy, with many limitations in terms of the methods, paradigms, training data, and computational resources available for solving tasks that require human-like pattern recognition.

Today, after many advances under the umbrella of Deep Learning (DL), from the learning algorithms (*e.g.*, back-propagation Rumelhart et al. 1986 and ADAM Kingma et al. 2014a), the neural architectures (*e.g.*, CNNs LeCun et al. 1998 and Transformers Vaswani et al. 2017), specialized integrated circuits (*e.g.*, Graphical Processing Units cite Tensor Processing Units), to the availability of large scale datasets (*e.g.*, ImageNet Deng et al. 2009), many of the previous limitations were almost overcome. As a result, DL now dominates the discipline of AI with its unprecedented success across many applications (*e.g.*, image classification K. He et al. 2016, image segmentation L.-C. Chen et al. 2017a and natural language understanding Devlin et al. 2019).

The success of many deep learning (DL)-based applications is partly due to the adoption of the supervised learning (SL) paradigm. In SL, a model is trained to perform a desired task by presenting it with inputs and their corresponding labels or targets. The inputs are the data we wish to analyze, and the labels are the desired predictions or outcomes we want to infer. The model is then trained using a pool of labeled examples to learn a mapping from input patterns to the correct output values. This mapping is often a complex and non-linear function that deep neural networks can approximate well. The learned features reflect the important qualities of the inputs that are useful for making the correct predictions for the task at hand. If a model with sufficient capacity is trained on a large and diverse enough dataset, the SL paradigm often produces a well-performing model at inference time. This model can then be deployed for a given real-world application, provided that the data is similar to those seen during training. But can we improve upon the SL paradigm? And what are its main limitations that might guide us in designing better DL-based systems?

Label-efficient DL

In order to identify the main limitations of the popular SL paradigm, we can take inspiration from how humans learn. Throughout the history of AI in general, and DL in particular, humans often played a joint role of serving as both a challenging benchmark for currently solvable tasks and guiding the design and conception of novel AI systems. For instance, the learning procedure that consists of adapting the connectivity weights of the model, one iteration at a time over a subset of input-output pairs, draws strong inspiration from how the brain works. So what are the main differences between human-like learning and SL-based DL methods?

If we compare the goal of AI, which is to develop machines with human-like intelligence, to the current popular SL-based methods, a clear divergence emerges, as pointed out by François Chollet

in quot. 2. Humans are able to adapt to novel environments with limited or no supervision or feedback mechanism (Piaget 1976), using their prior knowledge and experience to perform well across a wide variety of tasks. However, SL-based methods are often limited to a narrow set of tasks that require a large amount of high-quality labeled data, and they fail to adapt as effectively and efficiently to changes in inference conditions or to new tasks. In most real-world scenarios, collecting and creating such large and high-quality labeled sets is infeasible, requiring significant effort and resources. This makes the SL framework inapplicable in many settings, in which the availability of labeled data is limited.

> *"The promise of the field of AI, spelled out explicitly at its inception in the 1950s and repeated countless times since, is to develop machines that possess intelligence comparable to that of humans. But AI has since been falling short of its ideal: although we are able to engineer systems that perform extremely well on specific tasks, they have still stark limitations, being brittle, data-hungry, unable to make sense of situations that deviate slightly from their training data or the assumptions of their creators."* — François Chollet[3]. 2

Given such limitations, one wonders if it is possible to design well-performing DL-based methods that are capable of showing good generalization in scenarios where only limited or no amount of labeled data is available. In this work, we focus on answering this question and set to develop novel learning methods that are label-efficient, all while being effective and performant.

The Aim of this thesis

To summarize, and based on the aforementioned limitations and practical motivations, our aim is to develop effective and flexible learning algorithms and methods across a variety of conditions, *under the constraint of limited labeled data*, for different modalities and tasks.

1.1 Concrete Motivations

To concretely motivate the need for effective and flexible deep learning methods under the constraint of limited labeled data, we present an illustrative application that sheds light on the limitations of the standard SL approach. This will help us define the goals and desired properties of the label-efficient methods we will discuss later.

An Illustrative Application

In the medical field, a large amount of clinical data and medical records are collected and stored for each patient. Traditionally, these documents were stored in a written form on paper, but in recent years, this data is often saved electronically as Electronic Health Records, or EHRs (Evans 2016). As illustrated in Fig. 1.1, these EHRs contain the patient's information, from administrative

[3] François Chollet is a software engineer and artificial intelligence researcher known for creating the Keras deep-learning library.

Figure 1.1: Electronic Health Records. An illustration of the different types of data and the various information that can be stored in EHRs.

and billing data to health-related information such as medical histories, progress notes, radiology images, diagnoses, and medications. Using such digital records makes the storage process easier and more efficient, the patient's data readily available, and most importantly, in our case, easily usable to train and deploy DL-based methods.

To begin, let's think about how a medical professional might use such documents. First, they would likely quickly review the document to ensure it is of the relevant one and to identify the different sections of the document. Next, they would move on to the analysis and understanding phase, where they would examine the relevant sections of the document separately. For example, they might look at the section on the patient's medical history, then review the patient's latest diagnosis and analyze any related data such as MRI or CT scans, blood test results, and doctor's notes. As they do this, they may identify any inconsistencies or biases in the data that should be overlooked or corrected. Finally, the professional can use the identified relevant information to construct a mental model of the patient's medical condition and determine how to use it for the specific situation at hand.

The Drawbacks of a Fully-Supervised Approach

The way a medical professional examines the document can be described as a global-local-global approach. They first analyze the document's structure and identify relevant sections, then delve into these sections in detail to understand them, and finally synthesize all of the extracted information to develop a holistic understanding of the document. How can we design a DL system with similar capabilities to assist medical professionals with the automatic analysis of EHRs of a given patient, thus helping streamline their work processes, lighten workloads, and reduce any potential errors? To design a system with similar functionalities (see Fig. 1.2) , it must :

1. Operate over different levels of abstractions, both globally at the document level and locally at the section and item level.
2. Accept input data of various modalities *i.e.*, multi-modal inputs, mainly visual and textual inputs.
3. Produce outputs at different levels of abstraction and over the different input modalities.

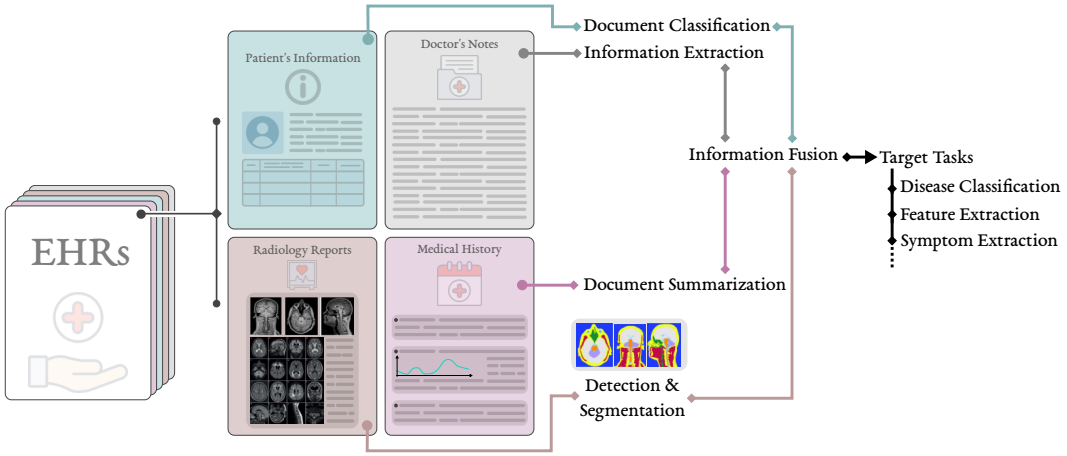


Figure 1.2: EHRs Analysis System. A sketch of a possible implementation of an EHRs analysis system. Note that this design is not intended to be realistic or practical, but only to highlight the need for a flexible DL-based method capable of processing various data types and modalities.

Based on these requirements, one potential implementation of this system could use a SL approach, with different sub-models for each level of abstraction and modality. Optional fusion models could be used to combine the outputs of these sub-models for predictions that require a cross-level, holistic view of the documents. Under this paradigm, we would need to construct input-label pairs in advance for all possible inputs the system might encounter, along with their corresponding desired targets. While conceptually straightforward and simple to implement, this approach has the following critical limitations:

- **Expensive labels.** To achieve acceptable performances, the training set must be large enough to accurately represent the underlying data distribution and include all possible input-output combinations. This requires creating a large dataset with high-quality labels, which can be time-consuming and resource-intensive. Additionally, while the annotation process for simple tasks like image classification of common objects with one or few labels per input can be straightforward, more complex tasks often require expert knowledge and careful attention to detail. In our example, for MRI scans, labeling at the pixel level requires precise annotations of various anatomical and pathological structures, which is much more expensive and labor-intensive than labeling at the region (*i.e.*, bounding boxes) or global level (*i.e.*, document type). This makes such fine-grained annotations notably more expensive and laborious (T.-Y. Lin et al. 2014), with a cost that can be up to 15 times higher than region-level labeling and 60 times higher than global-level labeling[4].
- **Erroneous and biased labels.** In addition to being expensive, the annotation task is often an ambiguous and subjective process. In it, the annotator's mental and physical state at the time of annotation, background, biases, and level of expertise are implicitly, and sometimes

[4] Additionally, in terms of annotation time, it takes up to 1h30min to finely annotate a single image of an urban scene of Cityscapes dataset (Cordts et al. 2016a) for instance, which can be used to train driverless car systems (Richter et al. 2016a).

explicitly, reflected in the produced labels. This can result in noisy and biased annotations that are hard to detect and filter out during the annotation process since such errors are often plausible yet incorrect (Paullada et al. 2021). In our example, the annotation of medical data is very susceptible to such errors, often requiring an independent annotation of the data by a group of experts or raters to mitigate such subjective biases and other possible errors such as the negligence of subtle symptoms (Joskowicz et al. 2019; S. Kumar et al. 2007; Schaekermann et al. 2019). This makes the annotation process even more expensive and cumbersome and might require the development of novel methods to take into account the inter-annotator variability.

- **Limited applicability.** The use of deep learning models in the SL paradigm is often limited by the need for a large and fully labeled dataset. This can be challenging in cases where the data is rare, the predictions are complex, or the annotation cost is high, such as in the fine-grained identification of rare plant species (Tan et al. 2019). In the case of medical diagnosis example, if a patient's health records indicate a rare disease that is not represented in the training data, the system may fail to detect this signal, leading to missed or incorrect diagnoses (Schaefer et al. 2020).

- **Limited scalability.** One of the simple and effective conventions in DL is that the bigger the scale, the better the results. A simple increase in the size of the training set, the model's capacity, and the computational resources will improve the system's performance (Hestness et al. 2017). However, this simple scaling law is not applicable in the context of SL, where the need for a fully labeled training set can make it impractical to increase the size of the dataset. In such cases, any scaling of the training set would require a corresponding increase in the annotation cost. This limitation is particularly relevant in the internet age, where vast amounts of data are readily available but often lack appropriate labels for SL-based DL methods. In our example, and similar to the rest of DL applications related to the medical domain, large quantities of data in the form of EHRs are readily available but with rare and sparse expert annotations.

- **Limited transferability.** To be well-performing and highly useful, a model must be able to adapt and generalize to a variety of examples from different tasks and domains. However, in a fully supervised setting, achieving this requires a large number of labels for each new scenario in order to adapt the model. This is impractical, as real-world data is complex and there are an infinite number of possible cases. Transfer Learning (Caruna 1993) tries to improve upon this by transferring knowledge from domains with a large amount of labeled data to those with less labeled data, but this approach can fail if the differences between the domains are significant and there is only a limited amount of labeled data available. In our example, in addition to the aforementioned rarity of good-quality labeled data, the domain gap problem is widespread in medical data (Guan et al. 2021) due to different equipment, measurement parameters, subject cohorts, etc. These factors thus limit the transferability of trained models from one domain to the other under the supervised paradigm.

- **Overly rigid learning objective.** In a supervised setting, defining an output label space that reflects the desired functionality is not always a simple and straightforward process. We may also wish to condition and change the outputs with respect to some external factors

(*e.g.*, open-set recognition Bendale et al. 2016) in which a model must not only distinguish between the training classes, but also identify novel classes not yet encountered. Moreover, assigning a single or few labels to a given input (*e.g.*, this is an image of a car) might be too restrictive as a training signal, pushing the model to focus only on the factors of the data that are related to the desired targets while disregarding the rest of the factors that might be as important. In our example, labeling the doctor's notes and the associated clinical tests with a single label corresponding to the identified disease might push the model to overlook some consequential aspects of the data, such as focusing on simple keywords in the input and disregarding the more complex relationships between the symptoms and the diagnosis. Additionally, some tasks, such as the association between the different sections of the medical document, cannot be easily represented using a pre-defined set of targets, and can change by use case and from one document to the other.

- **Biologically implausible.** As previously noted, AI aims to mimic the capabilities of humans who can learn new concepts and tasks efficiently from a limited set of examples and with minimal or no explicit supervision (Lake et al. 2017). This remarkable ability is often the result of humans' ability to repurpose and reutilize their experience and knowledge and use it as a seed to facilitate the learning process in novel settings. Yet, if we look back to the formulation of the SL paradigm, learning each new task using a distinct and specific dataset seems to be orthogonal to how humans learn. In our example, when a medical professional consults the patient's records, they draw from their expertise and years of education and experience to assist the patient, even if it is a rare case.

While SL suffers from all of the above limitations, it must be noted that these drawbacks are not associated exclusively with this paradigm. Many of such drawbacks still permeate the field of DL at large, with various active areas of research that seek to solve them. For instance, learning without any labels does not guarantee the removal of all biases (Steed et al. 2021), since some biases can still come from other sources unrelated to the annotators, such as the data collection process, *e.g.*, geographic biases (Shankar et al. 2017).

Learning under the constraint of limited labeled data

If we revisit the requirements of our automatic EHRs analysis system and based on the enumerated drawbacks of the standard supervised approach, adding the constraint of limited labeled data becomes a fundamental and necessary design choice. Although a constraint, training with limited supervision adds a significant amount of flexibility to our system, since now, instead of requiring labels at every level of abstraction and for all modalities, we can design and train our sub-models with the appropriate learning paradigm for each task based on the amount of supervision available. For instance, at the document level, where we may obtain labels easily, we might use a fully supervised approach. If labels are scarce at the local level, but there is plenty of unlabeled data available, a semi-supervised learning paradigm may be used. In the absence of labels, unsupervised learning may be the best option. If the data is scarce and the target classes are likely to change quickly, a few-shot learning approach that focuses on fast learning and rapid adaptation with only a few training examples may be appropriate.

Note that in this section, we used the EHRs analysis system as an example to motivate the need for label-efficient methods and to guide our selection of the tasks and the learning paradigms to be considered in this thesis. However, while it is worth exploring, designing such a system is not the focus of this thesis. Nonetheless, we hope that the contributions to be presented in this work can make the conception and implementation of similar systems more feasible and affordable in the future.

1.2 Objectives

As motivated in the previous section, this thesis studies the problem of learning effectively under the constraint of limited labeled data, for different modalities and tasks, and over various levels of abstraction[5]. Specifically, in settings where the amount of available labeled training examples is not sufficient to build a well-performing model under the supervised or the transfer learning (*i.e.*, when a pertained model is used as a starting point) paradigm, we set out to develop training methods that perform better than the supervised baseline. To this end, we start with the following simple hypothesis:

> *For a given task and under a given constraint of limited labeled training data, a new set of appropriate inductive biases[6] can be defined to prioritize solutions that better leverage the provided training data and better reflect the task objective, resulting in better generalization and performance.*

Such a hypothesis is perfectly in line with current popular DL methods, where several key inductive biases (*i.e.*, a set of preferences) over the space of all learnable functions are introduced to induce good generalization and acceptable performances (Goyal et al. 2020). The need for such preferences can be inferred from the no-free lunch theorem of machine learning (Baxter 2000; Wolpert et al. 1995). It states that for any given task with a finite set of training examples, many possible solutions exist that have equal generalization on the training set but differ on new and unseen examples. As such, given the complexities of the true underlying data distribution, which cannot be fully captured with a finite training set, some inductive biases are necessary to prioritize solutions with properties we deem appropriate for the task in question (*e.g.*, the use of CNNs LeCun et al. 1995 for applications where translation invariance is desirable).

From this perspective, the goal of this thesis can be reformulated as opting to adjust and adapt the standard inductive biases popularized under the SL paradigm to better match settings with limited or no labeled examples and the different tasks and modalities to be tackled. As such, in this work, we set the following the desiderata:

- **Effective and efficient learning.** Our first objective is to define better inductive biases to train models in settings where only a limited or no amount of direct supervision can be

[5] Here, we use "level of abstraction" to refer to the degree of granularity in which the inputs will be processed. For instance, an input might be processed at the image scale for classification or the pixel scale for segmentation.

[6] In this thesis, the set of inductive biases we will focus on will be limited mainly to the training objective and model architecture.

extracted from the training set to obtain better generalization and performance than the supervised baseline.

- **Considering multiple modalities.** As humans, our understanding of the world is inherently multi-modal (Edelman 1987; L. Smith et al. 2005), and our experiences are rarely limited to a single sensory stream. When we see an object, we often infer its name, texture, flavor, sounds, etc. Such a multi-modal understanding might be one of the pillars behind the efficiency humans demonstrate when learning new skills. Additionally, many real-world applications (*e.g.*, Section 1.1) also require processing different modalities. As such, in this thesis, our objective is to develop label-efficient learning methods for different modalities (*i.e.*, vision and text)[7].
- **Considering multiple levels of abstraction.** Across the many applications, DL methods have become the go-to approach, the set of chosen inductive biases (*e.g.*, model architecture and loss function) depends heavily on the task and its level of abstraction. For instance, in computer vision, classification at the image level requires a different model formulation than a classification at the pixel level. The new inductive biases to be introduced under the constraint of limited labeled data must also be conditioned on the task and its level of abstraction. Additionally, under the same modality, our objective is also to consider different tasks that operate over different levels of abstraction (*e.g.*, image classification and segmentation), making the set of the proposed methods more diverse and more readily applicable to systems that require such capabilities (*e.g.*, Section 1.1).
- **Conceptually simple and computationally efficient methods.** One of the main advantages of the SL approach is its simplicity, and with the appropriate hardware, its computational efficiency, making it the first choice for many real-world applications. Since our objective is to define better-performing methods suitable for real-world use cases, the proposed approaches must maintain these properties.

1.3 Scope

Tasks and Modalities

To fulfill the aforementioned high-level objectives defined in Section 1.2, we first need to narrow the scope of this thesis and focus on specific modalities and specific tasks within each modality. In terms of modalities, we consider both visual and textual tasks, given that vision and text are the two most popular modalities within the DL community, with many established methods and baselines that facilitate the evaluation of novel approaches, and with a broad number of real-world applications, making the impact of novel and effective methods more direct. For the specific tasks, we tackle the following ones:

- **Visual tasks.** For the visual domain, we consider two visual tasks: image classification and segmentation, thus covering different levels of abstraction, *i.e.*, both the image or instance level and the sub-instance or pixel level.

[7] Note that in this thesis we only consider a separate modality each time. We do not consider multi-modal settings where multiple modalities are used simultaneously.

- **Textual tasks.** For the textual domain, we consider the popular tasks within Natural language processing (NLP), which aims to map a given input text to outputs in the form of linguistic structures that encode its meaning (N. A. Smith 2011). For example, the sentiment analysis task consists of an N-way classification over N levels of polarization, or Natural language Inference (NLI) (MacCartney 2009) that consists of identifying the relationship between two given input sentences.

Learning paradigms

While reducing the application scope to cover the current popular tasks and modalities is relatively straightforward, defining the learning settings under the constraint of limited labeled data is more challenging, given the many variables and dimensions that can be considered. In this thesis, we consider the following angles of attack:

- **The type of data available.** The training data can contain either labeled examples only, unlabeled examples only, or both simultaneously.
- **The amount of data available.** In terms of the size of the labeled set, and by virtue of the constraint we impose, the amount of labeled data available must be limited compared to the standard supervised setting. However, the number of labeled examples can still have an outsized impact on the performance, even if limited, making it worth studying. Acquiring unlabeled data might also be an expensive process that impacts performance, but studying it is outside this thesis's scope.
- **The number of sub-tasks.** In many cases, only one sub-task is considered at a time. However, we may want to consider a setting with different sub-tasks for applications that require quick adaptation to novel and unseen sub-tasks at test time. For example, we can define multiple sub-classification tasks under the classification task, each consisting of classifying different breeds of a specific animal. By training the model to adapt quickly at test time, it can classify a new breed with only a few training samples and still have acceptable performance.
- **Distribution shift.** When evaluating a trained model, the standard procedure consists of a testing phase on examples that come from the same source (*i.e.*, the same underlying data distribution) as that of the training examples. In real-world applications, the trained model is often deployed on data with different characteristics (*e.g.*, different image backgrounds), and such a discrepancy, even small and unnoticeable, can significantly impact the accuracy of the predictions. As a result, this discrepancy must also be considered during the training phase to avoid any significant performance degradation at test time.

Based on these angles of attack, we consider the following learning paradigms (see Fig. 1.3):

- **Semi-supervised Learning (SSL).** In this setting, we have limited access to both labeled training examples and a larger number of unlabeled training examples. The training and testing sets share the same sub-task and underlying data distribution. The task of focus for this paradigm is image segmentation given its importance as a visual task and the limited SSL works tackling it.

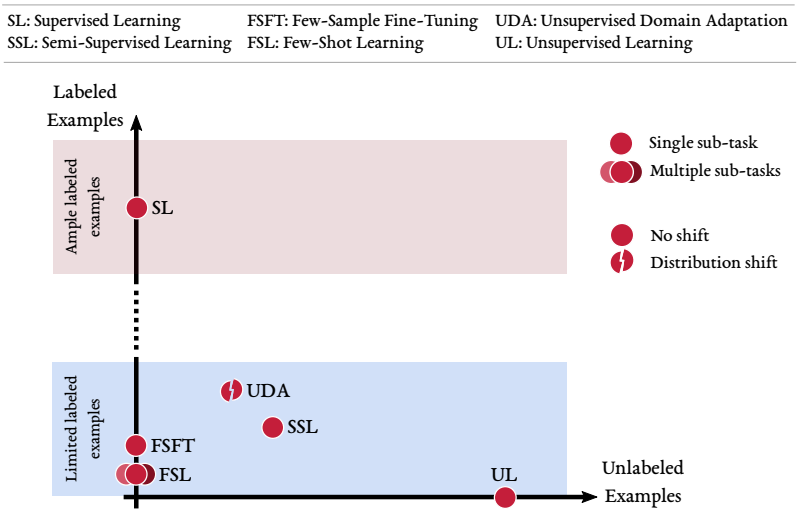


Figure 1.3: An illustrative map of the different learning paradigms. We show the different characteristics of each learning paradigm we consider in this thesis compared to the standard Supervised Learning (SL) paradigm. Note that the sole intent behind this figure is to provide a mental representation highlighting the main differences between the paradigms.

- **Unsupervised Learning (UL).** In this setting, we only have access to unlabeled training examples and no labeled examples. The training and testing share the same sub-task and the underlying data distribution. Similar to SSL, the task of focus for this paradigm is image segmentation.

- **Unsupervised Domain Adaptation (UDA).** In this setting, we consider the possible distributional shifts between labeled and unlabeled examples. Specifically, during the training phase, we are provided with both labeled and unlabeled sets, but each set is collected from a different domain (*i.e.*, source and target, respectively). The testing is then done on the same sub-task but only on examples from the target distribution. Note that in UDA, the amount of available labeled examples, while still limited, is generally larger than its SSL counterpart. The tasks of focus for this paradigm are image classification and segmentation given the flexibility of the approach and popularity of both tasks within the UDA community.

- **Few-Shot Learning (FSL).** In this setting, we consider many training and testing sub-tasks, in which we have access to a minimal amount of labeled training examples for each individual sub-task. The examples of both the training and testing sub-tasks share the same underlying data distribution. The task of focus for this paradigm is image classification given its popularity within the FSL community. In this case, it is worth noting that in the literature, the word *task* can have a dual meaning, referring both to the overall application (*e.g.*, image classification) and to the different sub-tasks (*e.g.*, the different sub-classification tasks). We use the term sub-task in this introductory chapter to avoid this confusion.

- **Few-Sample Fine-Tuning (FSFT).** In this setting, we consider a transfer learning scenario but with minimal labeled training examples available. The training and testing share the

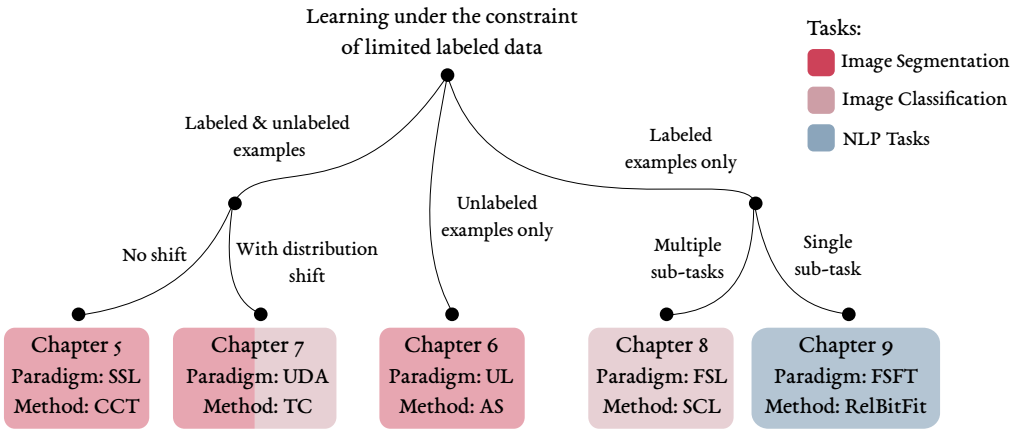


Figure 1.4: Thesis contributions and their corresponding chapters. We show the learning paradigms and the corresponding tasks tackled in each contribution to be presented in this thesis, together with their corresponding chapters.

Task	Modality	Learning Paradigm	Contribution	Venue
Segmentation	Vision	SSL	CCT (Ouali et al. 2020c)	CVPR'2020
Segmentation	Vision	UL	AS (Ouali et al. 2020b)	ECCV'2020
Segmentation & Classification	Vision	UDA	TC (Ouali et al. 2020d)	CaP'2020
Classification	Vision	FSL	SCL (Ouali et al. 2021)	ECML-PKDD'2021
NLP tasks	Language	FSFT	RelBitFit	To-be-published
		Survey articles		
An Overview of Deep Semi-Supervised Learning (Ouali et al. 2020a)				arXiv e-Print

Table 1.1: Thesis Contributions. The contributions proposed in our thesis, categorized by the task, the modality and the learning paradigm.

same sub-task and underlying data distribution. The tasks of focus for this paradigm are NLP tasks in order to tackle the textual modality.

Note that overall, the choice of the task to focus on for each paradigm is based on the availability of benchmarks to evaluate our methods on the amount of computational resources required to avoid excessive costs. In addition, the state-of-the-art at the time of the introduction of the proposed methods impacted the choice of tasks that might have been overlooked in the literature. For instance, when we set out to explore the semi-supervised paradigm, almost all of the methods were focused solely on image classification and, to a lesser extent, object detection, but only a handful of works tackled the task of image segmentation, which motivated the choice of this task in our work.

1.4 Outline and Contributions

Throughout the rest of this thesis, we will introduce the different proposed contributions to fulfill the objectives defined in Section 1.2 under the scope set in Section 1.3, *i.e.*, the identified learning paradigms and their corresponding tasks. These contributions are listed in Table 1.1, and each

Project	Description	Project URL
`pytorch-segmentation`	PyTorch Framework for Semantic Segmentation	pytorch-segmentation
`Notes`	Notes of various DL papers related to this thesis	ml-paper-notes
`SSL-papers`	Up to date SSL Papers	awesome-semi-supervised-learning
`CCT`	Official Implementation of CCT (Ouali et al. 2020c)	CCT
`SCL`	Official Implementation of SCL (Ouali et al. 2021)	SCL

Table 1.2: Thesis Open Source Projects. The open-sourced projects produced during the time of this thesis.

contribution will be detailed in its respective chapter as illustrated in Fig. 1.4. Additionally, the works presented in this thesis resulted in various open-source implementations and DL-related projects that gained relative success within the community. Refer to Table 1.2 for a brief overview.

Specifically, the rest of this thesis is divided into two parts: Part I provides the necessary background information, while Part II presents the main contributions of this work. In terms of content, each chapter will cover the following topics:

- **In Chapters 2 to 4**, we start by laying out the foundations we will build upon in the rest of the thesis. We first define the general DL-based learning framework used to solve a given problem. This is be followed by a presentation of the different learning paradigms that consider the learning problem under the constraint of limited labeled data. Finally, we conclude by introducing the relevant tasks, model architectures, and datasets to be used in the contributions parts of this work.
- **In Chapter 5**, we tackle the image segmentation task in the SSL setting. We start with an analysis of the smoothness of the learned features to examine the cluster assumption (*i.e.*, the inputs form clusters that correspond to the output classes) at the pixel level. Then, based on the observed behavior, we introduce Cross-Consistency Training (CCT), a novel semi-supervised learning method tailored for the image segmentation task. With extensive experiments, we demonstrate that CCT achieves state-of-the-art results in several benchmarks. We published this contribution in (Ouali et al. 2020c).
- **In Chapter 6**, we further reduce the level of supervision and explore the fully unsupervised setting for the image segmentation task. Based on recent advancements in UL and autoregressive generative modeling, we propose a novel view generation method rooted at the pixel level. It is designed specifically for image segmentation upon which we base our autoregressive segmentation method. The proposed can be applied for both clustering and representation learning objectives in the context of image segmentation. Through comprehensive experimentation, we demonstrate that our approach outperforms existing unsupervised methods for image segmentation. We published this contribution in (Ouali et al. 2020b).
- **In Chapter 7**, and as a natural extension to the SSL paradigm, we consider the possible distributional shift between the labeled and unlabeled sets, *i.e.*, the UDA setting. By investigating the robustness of standard UDA methods under the prism of the cluster assumption, we show that the cluster assumption is violated on the unlabeled set despite

being maintained in the labeled set. This indicates a lack of robustness on the latter. To address this problem, we enforce the cluster assumption on the unlabeled set while aligning the features between the two sets in a per-class manner. The proposed approach results in a notable improvement in both image classification and segmentation benchmarks. We published this contribution in (Ouali et al. 2020d).

- **In Chapter 8**, we abandon the single and shared task setting (*i.e.*, a single sub-task), and consider the FSL paradigm that optimizes for fast adaptability over many sub-tasks. To promote such fast adaptability, we examine a new training objective to learn more general and transferable features using contrastive learning (*i.e.*, the direct optimization of the features so that semantically similar inputs are mapped close by in the representation space and vice-versa) as a data-dependent regularizer during the training phase. Specifically, we present a novel attention-based spatial contrastive objective to learn locally discriminative and class-agnostic features. With extensive experiments, we show that the proposed method outperforms state-of-the-art approaches, confirming the importance of learning good and transferable embeddings for few-shot learning. We published this contribution in (Ouali et al. 2021).
- **In Chapter 9**, we shift from the visual domain to the textual domain and consider the FSFT setting. Given that large pre-trained language models have become the starting block for many NLP tasks, we position ourselves in the transfer learning phase, and set to find more label-efficient ways to adapt such models to the desired downstream tasks in a label-constrained environment. We build on the recently introduced work of parameter-efficient fine-tuning, and present a novel bias fine-tuning method that is both efficient and performant. It consists of a reformulation of the attention operation by injecting new relative-biases to increase the model's capacity during the fine-tuning phase. We show its effectiveness with multiple experiments on various NLP downstream tasks. This chapter is based on yet-to-be-published work.
- Finally, **in Chapter 10**, we conclude by discussing open challenges in designing algorithms requiring a limited amount of labels. We also examine possible future works to conceive more common algorithms that can be applied to a broader range of tasks and across diverse applications.

Page intentionally left blank

PART ONE
Foundations

Part I

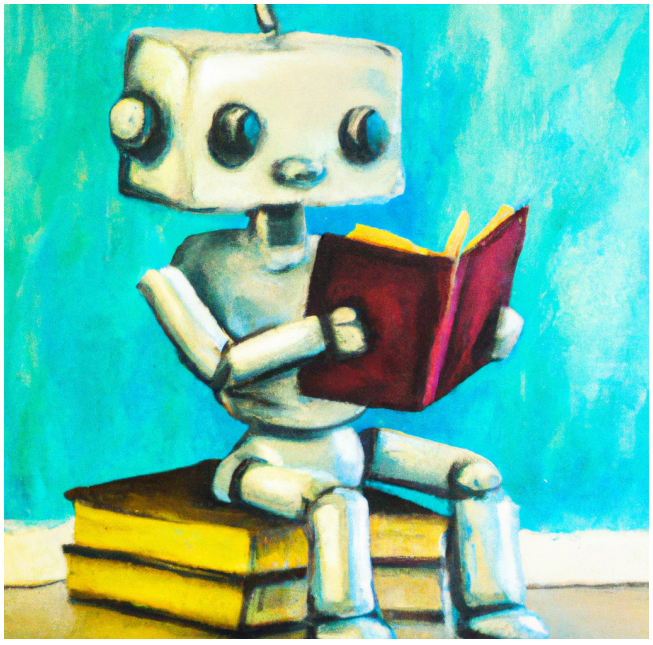
Foundations

Machine learning, in general, and deep learning, in particular, consist of equipping machines with a narrow form of intelligence tailored for a specific yet complex application using some form of past experience (*i.e.*, training data in our case). The essence of deep learning lies in the construction of many-layered neural networks, resulting in end-to-end systems where all the operations at each layer are learned jointly from training data. So what techniques, models, problem formulations, and applications make deep learning so popular and successful? In the first part of this thesis (*i.e.*, Chapters 2 to 4), we present the necessary deep learning-related background knowledge upon which Part II of this thesis is built (*i.e.*, the contributions presented in Chapters 5 to 9).

Note that for brevity and conciseness, the presented background is strictly limited to information related to the contributions of this thesis and is intended only to refresh the reader's memory rather than providing a comprehensive introduction. For a detailed overview, we refer the reader to the appropriate resources such as (Goodfellow et al. 2016) and (A. Zhang et al. 2021).

CHAPTER 2

A Learning Machine



2 A Learning Machine

First, we start by presenting the standard building blocks and design procedure followed when solving a given task of interest using a deep learning (DL) based system. For many practical problems, the desired objective is transformed into a prediction task to fit the DL framework, requiring a computer or, more specifically, a model to consume some input data we wish to process and produce the desired output (*e.g.*, taking as input a set of pixel values and predicting the category of the object they depict). To this end, a model is designed as a mapping from the inputs to the desired outputs, and since it is infeasible to specify the desired mapping manually given the task difficulty and the complexity of real-world data, the model is constructed as a many-layered network with learnable parameters to be learned from the data directly. In DL, the standard structure the model takes is a deep neural network consisting of multiple layers. In its simplest form, each layer involves a linear matrix-vector product followed by a non-linearity function (*i.e.*, a multi-layer perceptron Pal et al. 1992). Then, a learning procedure is conducted to update the model's parameters and obtain the desired mapping. It consists of returning an updated version of the parameters so that the behavior aligns with the desired task using some form of training data provided as input by the user. The ideal behavior is often specified using a loss or an objective function, which is used to compute a given performance measure. The desired and ideal behavior of the model can then be inferred by updating its parameters to minimize this loss over the training data. Finally, after training, an evaluation setup is often conducted to probe the model and verify that its behavior is consistent with the desired one. This step generally consists of testing the trained model on samples unseen in their exact form during the training phase, producing an evaluation measure that provides a proxy of the eventual performances we can expect from the model during deployment. If the results are satisfactory, the model can be applied to solve the desired application. Otherwise, the model design, the learning procedure, and the training data must be revised to improve performance. Next, we define this learning problem more formally and detail its different instantiations that consider the label-efficient learning problem.

2.1 Problem statement and terminology

In this section, and while the statistical learning theory (Bousquet et al. 2003; James et al. 2013; Vapnik 1999) is vast and detailed, we take a more practical and narrow view of the learning problem. Generally, the problem of learning and inference is that of knowledge extraction, prediction generation, decision-making, or model construction based on a set of data. This learning and inference process consists broadly of the following steps (Bousquet et al. 2003):

- Observing a given phenomenon of interest.
- Constructing a model of that phenomenon.
- Making predictions and decisions based on this model.

In this thesis, we are interested in a machine learning, or specifically, a DL-based learning framework that tries to automate parts of this process (*i.e.*, learning and inference). Under this framework, and in a practical setting, the observations are provided as training data consisting of many data points that encode some knowledge about a task of interest. First, we define a learnable mapping from the input instances to the desired outputs as our inference function. Then, from a collection of possible models, the learning procedure selects the model that best fits the provided training data. This model is then evaluated on some previously unseen data to probe its degree of generalization.

2.1.1 The Standard Framework: Supervised Learning

Formally, we consider an input space $\mathcal{X}$ and output space $\mathcal{Y}$, and we assume that an input instance or observation $\mathbf{x}$ belongs to this input space $\mathcal{X}$, *i.e.*, $\mathbf{x} \in \mathcal{X}$, and that an output or desired target y belongs to the output space $\mathcal{Y}$, *i.e.*, $y \in \mathcal{Y}$. We assume there exists a fixed unknown joint data distribution $p(\mathbf{x}, y)$ according to which the data are identically and independently distributed (iid). We also assume that this distribution factorizes as $p(\mathbf{x}, y) = p(\mathbf{x})p(y \mid \mathbf{x})$, with the conditional distribution $p(y \mid \mathbf{x})$ describing the relation between the inputs and targets, and the marginal distribution $p(\mathbf{x})$ modeling the uncertainty in sampling the input instances.

The goal of a learning procedure is to estimate the best input-output mapping to solve a given task of interest. This is done via an inference function[1] $f_\theta : \mathcal{X} \to \mathcal{Y}$ that maps the inputs $\mathbf{x}$ to the desired outputs y and consists of learnable parameters $\theta \in \Theta$, with Θ as the parameter space defined by our choice of the neural architecture (*e.g.*, CNNs LeCun et al. 1995 or transformers Vaswani et al. 2017).

In its simplest form, the task we wish to solve is described by a set of labeled training data points (*i.e.*, a supervised learning setting) and a task-specific loss function. The learning procedure consists of finding the optimal parameters so that the produced inference function is capable of generating the correct predictions over unseen examples, Formally, a given task $\mathcal{T}$ is often specified by its corresponding dataset $\mathcal{D}$ and task-specific loss function $\mathcal{L} : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$. The dataset $\mathcal{D} = \{\mathcal{D}^{\text{tr}}, \mathcal{D}^{\text{test}}\}$ consists of a training dataset $\mathcal{D}^{\text{tr}} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$ containing N input-target pairs sampled iid from the joint probability distribution $p(\mathbf{x}, y)$, and a test dataset $\mathcal{D}^{\text{test}} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N'}$ containing N' unseen examples sampled iid from the same distribution[2]. The loss function $\mathcal{L}$ is a point-wise measure of the prediction error $\mathcal{L}(y, f_\theta(\mathbf{x}))$ that incurs when the inference function predicts $\hat{y}$ instead of y, with $\hat{y} = f_\theta(\mathbf{x})$. It is worth noting that in many cases, the loss function $\mathcal{L}$ we care about (*e.g.*, classification error) is an intractable loss that cannot be optimized efficiently, so we typically optimize a tractable surrogate loss function (*e.g.*, cross-entropy loss) instead, that acts as a proxy for the original loss function (Goodfellow et al. 2016).

The learning problem consists of finding the optimal the parameters θ^* so that the behavior of the inference function $f_\theta(\mathbf{x})$ minimizes the population risk $\mathcal{R}[f_\theta] = \mathbb{E}_{(\mathbf{x},y)\sim p}[\mathcal{L}(f_\theta(\mathbf{x}), y)]$. However, since we only have access to the distribution $p(\mathbf{x}, y)$ via our training dataset $\mathcal{D}^{\text{tr}}$, the

[1] In this thesis, we also refer to it as the predictive function, the representation function, or simply, the model

[2] In this thesis, we regard the validation set as either a testing set if the testing set is not explicitly provided or otherwise as an extension of the training set used for hyperparameter tuning.

learning procedure consists of finding the optimal parameters θ^* by minimizing the empirical risk $\mathcal{R}_{\text{emp}}[f_\theta]$ as follows (Vapnik 1999):

$$\begin{aligned} \theta^* &= \arg\min_\theta \mathcal{R}_{\text{emp}}[f_\theta] \\ \text{with } \mathcal{R}_{\text{emp}}[f_\theta] &= \frac{1}{N}\sum_{i=1}^{N} \mathcal{L}(y_i, f_\theta(\mathbf{x}_i)) \end{aligned} \tag{2.1}$$

By solving the optimization problem of Eq. (2.1), we hope that the learned parameters θ^* also minimize the average test error over unseen test samples $\mathcal{D}^{\text{test}}$ so that the inference function can be reliably used to solve the task of interest.

2.1.2 A More Flexible Framework

In this thesis, we are interested in a more general learning problem than the supervised setting, *i.e.*, learning under the constraint of limited labeled data. Thus, we introduce a more flexible learning framework that we will use to express the different learning paradigms to be considered in this work. Precisely, we introduce the following notions and extensions:

- **Multiple Domains.** The first extension consists of considering multiple domains or environments. Each domain D is defined by a dataset $\mathcal{D}_D^{(\cdot)}$ containing iid examples from its corresponding probability distribution $p_D(\mathbf{x}, y)$. Specifically, we consider either a single domain, falling back to the standard setting, or two domains, the source domain D_s and the target domain D_t, with their corresponding probability distributions p_s and p_t, respectively.
- **Multiple training subsets.** We consider that a given training dataset $\mathcal{D}^{\text{tr}} = \{\mathcal{D}_l^{\text{tr}}, \mathcal{D}_u^{\text{tr}}\}$ consists of two subsets, a labeled subset $\mathcal{D}_l^{\text{tr}} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N_L}$ consisting of N_L input-label pairs and an unlabeled subset $\mathcal{D}_u^{\text{tr}} = \{\mathbf{x}_i\}_{i=1}^{N_U}$ consisting of N_U input instances only without any labels, with $N = N_L + N_U$. As such, a dataset $\mathcal{D} = \{\mathcal{D}_l^{\text{tr}}, \mathcal{D}_u^{\text{tr}}, \mathcal{D}^{\text{test}}\}$ of a given task $\mathcal{T}$ consists of three subsets: a labeled training set $\mathcal{D}_l^{\text{tr}}$, an unlabeled training set $\mathcal{D}_u^{\text{tr}}$, and a test set $\mathcal{D}^{\text{test}}$. Each of these three subsets can be from a different domain D, *i.e.*, sampled from a different probability distribution p_D.
- **Multiple Tasks.** We extend the learning objective to many tasks sampled from a task distribution $p(\mathcal{T})$. The training stage then consists of training our inference function to learn to adapt to this task distribution. In this case, the tasks of interest are divided into training tasks $\mathcal{I} = \{\mathcal{D}_t\}_{t=1}^{T}$ and testing tasks $\mathcal{S} = \{\mathcal{D}_q\}_{q=1}^{Q}$, and the dataset $\mathcal{D}_{(\cdot)}$ of each task consists of a labeled training and testing subsets (*i.e.*, no unlabeled training set). All the subsets come from the same domain, *i.e.*, single domain setting.

Based on such an extension, the learning problem becomes more flexible and can take into account different paradigms that deal with label-efficient learning. For an overview of the terminology and notation introduced in this section, refer to Table 2.1

[3] We will use f to refer to f_θ, and both notations will be used interchangeably in this thesis.

Symbol	Notation	Description
$\mathcal{T}$	Task	The task we want to solve via an inference function. It is described with its corresponding dataset $\mathcal{D}$.
f_θ[3]	Inference function (model)	A learnable mapping from the inputs $\mathbf{x} \in \mathcal{X}$ to the desired outputs $y \in \mathcal{Y}$, with learnable parameters θ.
$\mathcal{L}$	Loss function	A point-wise measure of the prediction error that incurs when the inference function produces incorrect outputs. The choice of the loss function will often depend on the task and paradigm considered.
$\mathcal{D}$	Dataset	A dataset corresponding to a given task and consisting of two subsets: - The training set: $\mathcal{D}^{\text{tr}}$. - The test set: $\mathcal{D}^{\text{test}}$.
$\mathcal{D}^{\text{tr}}$	Training set	A training set of a given dataset consisting of two subsets: - The labeled training set: $\mathcal{D}_l^{\text{tr}}$. - The unlabeled training set: $\mathcal{D}_u^{\text{tr}}$.
D	Domain	A domain (or an environment) defined by a dataset $\mathcal{D}_D^{(.)}$ containing iid examples from its corresp. distribution p_D.
$p(\mathcal{T})$	Task distribution	In a setting with multiple tasks, we consider a task distribution $p(\mathcal{T})$ from which the training $\mathcal{I} = \{\mathcal{D}_t\}_{t=1}^{T}$ and testing tasks $\mathcal{S} = \{\mathcal{D}_q\}_{q=1}^{Q}$ are sampled.

Table 2.1: Notation Summary. A summary of the different notations and terminology used in this thesis.

2.2 Scenarios and Variations

In this section, we consider various paradigms that deal with the learning problem under the constraint of limited labeled data, which are special cases of the flexible learning framework introduced in Section 2.1.2. Refer to Table 2.2 for a summary of these paradigms and to Fig. 2.1 for a visual description of them.

The Supervised Setting. In the classical supervised setting, we are interested in a single task $\mathcal{T}$ with its corresponding dataset $\mathcal{D} = \{\mathcal{D}^{\text{tr}}, \mathcal{D}^{\text{test}}\}$ from domain D, and we are provided with a sufficient number of labeled training examples $\mathcal{D}^{\text{tr}} = \mathcal{D}_l^{\text{tr}}$ specific to this task and domain. We then define the neural architecture of our model f_θ with randomly initialized parameters θ, and then train it on the labeled training set. This is the standard *Supervised Learning* paradigm.

Knowledge Transfer. In this case, we have access to two tasks, a source $\mathcal{T}_s$ task and a target $\mathcal{T}_t$ task, both with their corresponding labeled training sets. The source training samples are from the source domain D_s, and the target training samples are from the target domain D_t. The objective is to learn a prediction function that performs well on the target's test set, *i.e.*, solve the target task $\mathcal{T}_t$. If the two tasks are sufficiently similar and have a small enough target-source domain gap, we can use the source training set to better initialize some or all of the model's parameters θ. Then we proceed to the *Fine-Tuning* stage, where we train and evaluate our model following the classic

Learning Paradigm	Task(s)	Domain(s)	Training Data (Domain)	Test Data (Domain)
Supervised Learning	$\mathcal{T}$	D	$\mathcal{D}_l^{\text{tr}}(D)$[4]	$\mathcal{D}^{\text{test}}(D)$
Semi-Supervised Learning	$\mathcal{T}$	D	$\mathcal{D}_l^{\text{tr}}(D)$ and $\mathcal{D}_u^{\text{tr}}(D)$	$\mathcal{D}^{\text{test}}(D)$
Unsupervised Learning	$\mathcal{T}$	D	$\mathcal{D}_u^{\text{tr}}(D)$	$\mathcal{D}^{\text{test}}(D)$
Unsupervised Domain Adaptation	$\mathcal{T}$	D_s, D_t	$\mathcal{D}_l^{\text{tr}}(D_s)$ and $\mathcal{D}_u^{\text{tr}}(D_t)$	$\mathcal{D}^{\text{test}}(D_t)$
Few-Shot Learning	$\mathcal{T} \sim p(\mathcal{T})$	D	$\{\mathcal{D}_t\}_{t=1}^{T}$ with $\mathcal{D}_t(D)$	$\{\mathcal{D}_q\}_{q=1}^{Q}$ with $\mathcal{D}_q(D)$
Transfer Learning	$\mathcal{T}_s, \mathcal{T}_t$	D_s, D_t	$\mathcal{D}_l^{\text{tr}}(D_s)$ and $\mathcal{D}_l^{\text{tr}}(D_t)$	$\mathcal{D}^{\text{test}}(D_t)$

Table 2.2: Learning Paradigms. A summary of the different learning paradigms we consider in this work.

supervised scenario on the target domain. This process, which we refer to as *Transfer Learning*, results in a sequential knowledge transfer from the source domain to the target domain, which can help with convergence speed and reduce to some extent, the amount of labeled training data required. In recent years, initializing the model's parameters from some source domain has become a standard practice given the availability and ease of use of pre-trained models on various large-scale datasets. Most DL methods consider it as the natural starting point regardless of the task or setting being considered. Note that in cases with few labeled target data, knowledge transfer coupled with the standard supervised fine-tuning method becomes insufficient. This results in overfitting of the target samples and poor generalization, thus requiring a different solution to such a problem. In this work, we refer to this scenario as *Few-Sample Fine-Tuning* (FSFT).

Abundant Data but Few Labels. Disregarding a minority and rare type of data where the collection process is costly (*e.g.*, in the medical field), unlabeled data is available in large quantities for most applications. However, this is not the case for labeled data, for which the annotation process quickly becomes expensive or even infeasible, in addition to being highly prone to errors and biases. As such, many areas of DL focus on developing label-efficient methods to overcome such hurdles while maintaining a good degree of generalization. In a setting similar to supervised learning with a single task and domain, if the provided training set only contains unlabeled examples, *i.e.*, $\mathcal{D}^{\text{tr}} = \mathcal{D}_u^{\text{tr}}$, then we are in a fully *Unsupervised Learning* setting. Otherwise, if a small portion of the training set is labeled while the rest is unlabeled, *i.e.*, $\mathcal{D}^{\text{tr}} = \{\mathcal{D}_l^{\text{tr}}, \mathcal{D}_u^{\text{tr}}\}$, we are in a *Semi-Supervised Learning* setting.

With a Distribution Shift. When deploying a trained model for a given application on real-world data, the data encountered will likely have some dissimilarity compared to the training data, resulting in noticeable performance degradation. As such, many DL settings consider a setup with similar conditions to tackle this problem. For a given task $\mathcal{T}$, we consider two domains, a source domain D_s and a target domain D_t. The target domain will simulate real-world data via a distribution shift with respect to the source data, *i.e.*, the marginal probability distributions of the source and target domains are different $p_s(\mathbf{x}) \neq p_t(\mathbf{x})$. Given this setup, many possible variations can be constructed: Do we have access to both target and source data available during training, or

[4] Here the notation $\mathcal{D}^{(\cdot)}(D)$ is used to convey that the examples of the set $\mathcal{D}^{(\cdot)}$ originate from domain D, *i.e.*, they are sampled iid from the probability distribution p_D of domain D.

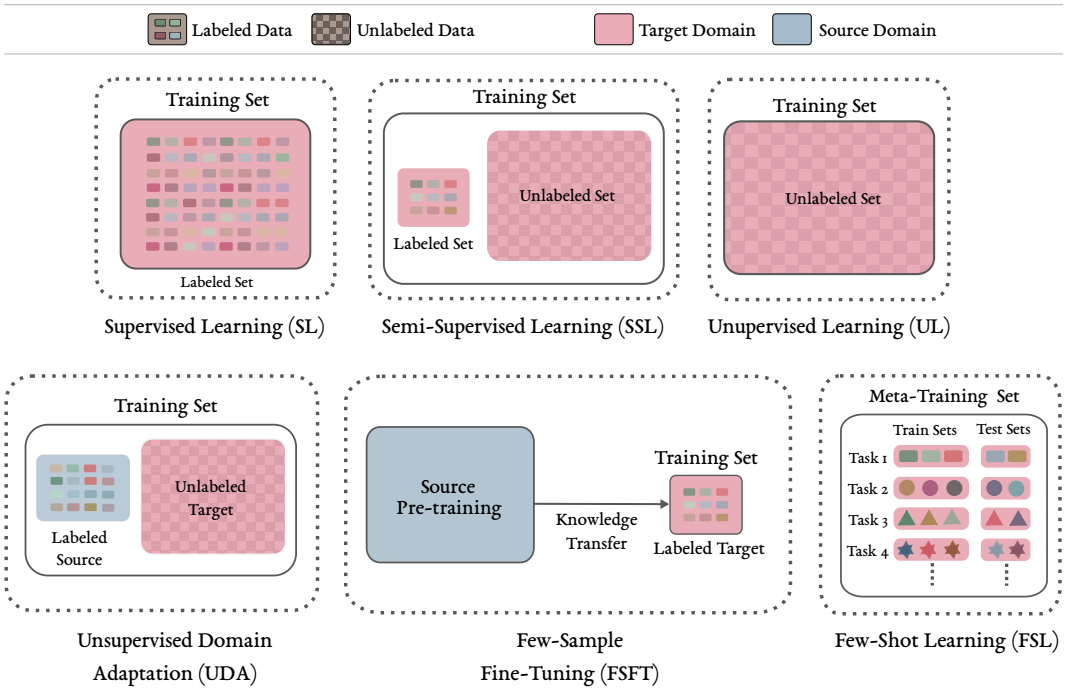


Figure 2.1: Learning paradigms. We show the structure of the provided training sets for each of the learning paradigms we tackle in this thesis.

only the target data and a model trained on source data? Do we have access to labeled target data, or are the labels available only for the source data? These variations and others not mentioned fall under the *Domain Adaptation* setting. In this thesis, we consider the *Unsupervised Domain Adaptation* setup, in which we have access to labeled examples $\mathcal{D}_l^{\text{tr}}$ from the source domain but only unlabeled training examples $\mathcal{D}_u^{\text{tr}}$ from the target domain. The model is evaluated solely on data from the target domain at test time.

Meta-Learning. In all the cases above, we considered a single task. The learning task consisted of finding the optimal parameters of the model at training time, fixing them, and deploying the model at test time. But what if our objective is to quickly solve new tasks that will be encountered at test time? In this case, rather than being fed a single example and providing the corresponding prediction in return, during testing, we will encounter a new task, with its training and testing sets. The objective is to quickly adapt our model to this task using the provided training set and then returning the predictions over the test set. In this case, we go from a learning problem to a *meta-learning* problem, where instead of a training stage on a single task $\mathcal{T}$ using its training set, we will adapt our model to a distribution of tasks $p(\mathcal{T})$ during a meta-training set. This set consists of many training tasks $\mathcal{I} = \{\mathcal{D}_t\}_{t=1}^{T}$, and each task has its corresponding dataset $\mathcal{D}_t$ with its labeled training $\mathcal{D}_t^{\text{tr}}$ and testing $\mathcal{D}_i^{\text{test}}$ sets[5]. The overall objective becomes training a model that can quickly adapt to new tasks. This adaptability is often evaluated during a meta-testing stage

[5] In a meta-learning setting, the training and testing sets of each task are also called the support and query sets, respectively.

with a set of novel tasks $\mathcal{Q} = \{\mathcal{D}_q\}_{q=1}^{Q}$ unseen during meta-training. In this thesis, we consider the *Few-Shot Learning* scenario, where the training set of each task consists of only a handful of data points (*e.g.*, 1 to 5 examples per class for each task), which is often referred to as a K-shot learning setting where the model is adapted to a novel task using only K training samples per class. Additionally, in its standard form, all the tasks share the same domain and fall under the same underlying general classification task. As such, to avoid confusion, we also refer to them as sub-tasks to differentiate them from the main classification task in cases where confusion may arise, as mentioned in Section 1.3.

2.2.1 Generalization

Under the previously highlighted settings, the learning problem becomes more elaborate and involves many possible variations, but regardless of the paradigm and setting to be considered, the goal remains the same: generalization. For all these settings, the goal of the learning process is not limited to producing an inference function that models the training data well, but also consists of the ability to *generalize* well to data unseen during training. However, it would be unreasonable to seek an unconstrained generalization to unseen and unspecified examples from a finite training data sample since any form of generalization cannot arise in a vacuum, but only as a by-product of many prior assumptions and helpful information explicitly chosen for the task(s), the domain(s), and the setting at hand. As a result, the problem of generalization becomes a selection and design process to specify the optimal and appropriate priors that will maximize the test performances on the task(s) and domain(s) being considered.

In DL, the set of priors, or *inductive biases*, consists broadly of the neural architecture used to construct the model f_θ (*i.e.*, the parameter space Θ), the objective function $\mathcal{L}$, and the learning procedure. All of them are chosen based on the setting being tackled. Similarly, in this thesis, our contribution process will generally consist of the following two steps:

- We consider one of the paradigms defined in Section 2.2 that deals with the problem of label-efficient learning.
- We introduce the appropriate set of priors (*i.e.*, mainly the loss function and the model's architecture) that better reflects the setting tackled and produces better generalization than previous works.

Next, we briefly introduce the standard steps to be followed in the prior selection and training stages.

2.3 In Practice

To summarize, let us present the standard steps one might consider when implementing a DL system to solve a given task under one of the previously defined flexible settings. When it comes to the design and implementation process, the most crucial step is the careful selection of the appropriate priors that best match the setting and task(s). Overall, this process consists of the following steps (see Fig. 2.2):

- Defining the neural architecture of the prediction function f_θ, *i.e.*, the parameter space Θ of all possible inference functions.

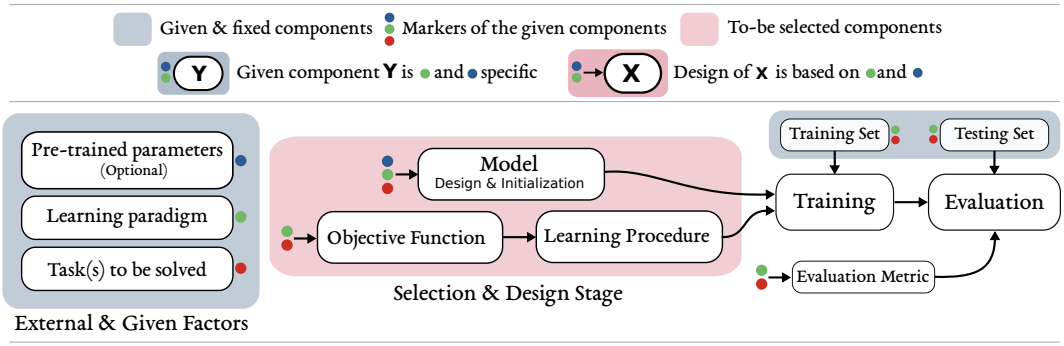


Figure 2.2: DL Pipeline. We show the main steps one must follow to implement a DL-based solution for a given application of interest. In this thesis, we will focus mostly on the design stage of the objective function and the model's architecture to define revised inductive biases that better reflect the learning paradigm and the task we are interested in.

- Finding a good model initialization via transfer learning if such a possibility is available.
- Specifying the loss function $\mathcal{L}$, which is a scalar-valued function that measures the mismatch between the desired output and the model's prediction.
- Selecting the learning procedure to optimize the model's parameters θ.
- Choosing the appropriate task-specific metric for evaluation.
- Designing the overall pipeline, from the data manipulation and pre-processing step to the training, evaluation, post-processing, and even the system deployment.

After these choices are made, the next step is to train the model, *i.e.*, finding the optimal parameters θ^* that minimizes the loss $\mathcal{L}$ over the training set, which is then followed by an evaluation step, *i.e.*, computing the task-specific metric(s) over the unseen examples of the testing set to get an indication of the generalization capabilities of the model. While all the above steps are important, in this thesis, we mostly focus on the prior selection related steps, *i.e.*, model architecture, objective function, and learning procedure.

While the testing stage is often simple to implement and follows the same procedure as prior works, the training stage is the most important step since it involves the task of finding the optimal parameters that will then determine the degree of generalization the model will exhibit. It involves solving an optimization problem in Eq. (2.1), *i.e.*, , $\theta^* = \arg\min_\theta \mathcal{R}_{\text{emp}}[f_\theta]$. In DL, and since such a problem is often solved using first-order methods[6], the neural architecture of choice must be constrained to only using differentiable functions to be able to compute the gradients $\nabla_\theta \mathcal{L}$ with backpropagation. These gradients are then used to obtain slightly better parameters θ by adding a small negative amount of them to achieve a slightly lower loss. This method, which consists of alternating between computing the gradients with backpropagation and updating the parameters following the direction of the negative gradients is what constitutes the backbone of the learning algorithms in DL, *i.e.*, the gradient descent algorithm. Practically, and since the datasets

[6]Methods that iteratively update the model's parameters in order to minimize a given objective function using only the first derivative of the objective function with respect to the learnable parameters.

used in DL are often very large, computing the gradients over all examples is infeasible. As a result, we iterate over the whole dataset, computing them over only a limited number of examples (*i.e.*, a mini-batch $\mathcal{B}$) each time, *i.e.*, the Stochastic Gradient Descent (SGD) algorithm. In addition to SGD, there also exist other neural network optimizers (*e.g.*, SGD with momentum Sutskever et al. 2013, ADAM Kingma et al. 2014a, and SAM Foret et al. 2021) that differ in the manner in which the gradients are computed to promote additional properties, such as faster convergence, better robustness or more stable training.

To summarize, the learning procedure in DL consists overall of the following steps, which are repeated until convergence:

1. Sample a mini-batch $\mathcal{B} \sim \mathcal{D}^{\text{tr}}$ of $N_{\mathcal{B}}$ training datapoints.
2. Forward each input data point $\mathbf{x}_i \in \mathcal{B}$ through the prediction function to get the prediction $f_\theta(\mathbf{x}_i)$.
3. Compute the average loss over the mini-batch between the predictions and the desired outputs, *i.e.*, $\frac{1}{N_{\mathcal{B}}} \sum_{i=1}^{N_{\mathcal{B}}} \mathcal{L}(f(y_i, f_\theta(\mathbf{x}_i))$.
4. Compute the gradients of each parameter of the model with respect to the loss using backpropagation.
5. Apply a gradient step to update the parameters with a neural network optimizer such as SGD.

Note that the above steps only describe a standard learning procedure. While the process is very similar in a meta-learning setting, some additional steps are needed. For instance, the first step consists of sampling a mini-batch of *tasks*, then sampling a mini-batch of *examples* for each one of the sampled tasks. Additionally, some meta-learning methods such as MAML (C. Finn et al. 2017) are based on second-order methods, where the model's parameters are meta-learned so that the model is easily adaptable at test time.

Key Takeaways

- DL consists broadly of designing inference functions based on deep neural networks trained in an end-to-end manner to solve a given task of interest.
- When designing a DL-based system, the aim is to equip it with a good degree of generalization that is sufficient for the system to be deployed on real-world data while maintaining acceptable performances.
- The design stage of a given DL system often consists of selecting the appropriate set of inductive biases (*e.g.*, model architecture, learning objective, and learning procedure) that are appropriate to the setting and task(s) we are tackling so that the desired degree of generalization can be attained.
- After the design stage, the DL model is then trained on the provided training data under the pre-defined inductive priors and then evaluated on held-out test data to provide a proxy measure of the eventual performance during its deployment.

Usage

In this thesis, we will follow this same DL pipeline and set to introduce better and more appropriate inductive biases for various settings we consider under the constraint of limited labeled data and for various tasks.

Conclusion

In this first chapter, we presented the reader with the standard formulation of a learning problem, the different learning paradigms we consider in this thesis, and the standard DL-based pipeline to be followed to solve a given task.
Next, we introduce these learning paradigms in more detail in Chapter 3. It is then followed in Chapter 4 by an introduction of the tasks we consider, with their definitions, the models used to solve them, and the data used to train and evaluate the proposed methods.

Page intentionally left blank

CHAPTER 3

Learning Paradigms

3 Learning Paradigms

This second chapter presents the different DL approaches that leverage the scenarios introduced in Chapter 2, *i.e.*, the type of training data available, the domains the data belong to, and the task(s) to be solved.

Specifically, we detail the popular methods for the different learning paradigms that tackle the learning problem under limited labeled data constraints. These methods set to define the correct priors, mainly the objective function and the learning procedure that best matche the setting at hand. Generally, for each paradigm, we try to formulate the learning problem as a prediction task and train a model to produce predictions that match a given target set, even if the actual underlying targets are unavailable or inaccessible. This process enable us to extract a training signal even if the labels are unavailable.

3.1 Supervised Learning

Supervised Learning (SL) remains to this day, the most widely used method to solve many prediction problems for various applications that require complex mappings from the observations to the desired targets. While currently very popular, this paradigm has existed for decades and was used for different commercial applications. However, such applications were limited to straightforward and elementary problems.

SL is a learning paradigm in which we assume that a sufficient and representative amount of labeled data consisting of input-target pairs (*i.e.*, the input $\mathbf{x}$ and its corresponding desired output or target y) is provided to us in advance. This data enables us to learn an optimal inference function to solve the problem of interest. Common SL problems consist of classification tasks where we learn a mapping from the inputs to classes or category IDs and regression tasks where the outputs are real-valued. In this work, we are mainly interested in classification tasks.

To solve a classification task over C classes or categories we are interested in using SL, we first design our model f as a deep neural network so that its output $\hat{y} = f(\mathbf{x})$ is a vector of C values per a given input $\mathbf{x}$. In this case, $\hat{y}$ is interpreted as probabilities over the C classes, *i.e.*, $\hat{y} \in [0, 1]^C$ and $\hat{y}_c$ as the probability of predicting class $c \in \{1, \ldots, C\}$. The standard choice to cast the outputs into probabilities is either a softmax layer (*i.e.*, $\mathrm{softmax}(\hat{y}_c) = \frac{e^{\hat{y}_c}}{\sum_{j=1}^{C} e^{\hat{y}_j}}$) for C-way single-label classification, or a sigmoid activation function (*i.e.*, $\mathrm{sigmoid}(\hat{y}_c) = \frac{1}{1+e^{-\hat{y}_c}}$) for binary or a C-way multi-label classification. Then, to train our model to assign the correct classes to each input data point, we need to optimize its weights so that it assigns 1 to the true class(es) and 0 to all the rest. This is generally done using the Cross-Entropy (CE) loss as the objective function. The CE loss measures the difference between the output $\hat{y}$, which is interpreted as probabilities over the C desired classes, and the corresponding ground-truth label y, which is the true distribution that has all probability mass on the correct class(es). With $\hat{y}_c$ and y_c are the output probabilities

approximated using the model's output and ground-truth label for a given class $c \in \{1, \ldots, C\}$, respectively, the CE loss is defined as follows:

$$\mathcal{L}_{\text{CE}}(y, \hat{y}) = -\sum_{c=1}^{C} y_c \log(\hat{y}_c) \tag{3.1}$$

Starting from randomly initialized parameters θ, and by minimizing the average CE loss over a batch of examples during the training stage, and by following the standard SGD-based training procedure (*e.g.*, Section 2.3), we obtain the optimal parameters so that the model produces the correct classifications over the unseen examples. Note that optimizing Eq. (3.1) is equivalent to maximizing the likelihood of data where one seeks to find the parameters of the model that maximize the probability of the data under such a configuration (Goodfellow et al. 2016), making the CE loss a theoretically sound choice in such a setting.

3.1.1 A Related Setting: Transfer Learning

Transfer Learning (TL) (Pan et al. 2009) aims to improve the learning of an inference function on the target domain D_t to better solve the target task $\mathcal{T}_T$ using knowledge from the source, even if the source and target domains or tasks differ. Under such a definition, the two possible supervised variations of TL are:

- **Inductive TL**, in which the source and target tasks are different, thus requiring labeled data in the target domain to learn the inference function. If we have access to source data, both tasks can be simultaneously learned using multi-task learning (Y. Zhang et al. 2018). Otherwise, if the source data is unavailable, we must find an alternative method to transfer knowledge from the source. A possible choice is initializing some or all of the model's parameters before the SL-based training (*i.e.*, fine-tuning) on the target domain.

- **Transductive TL**, in which the tasks are the same, but the two domains are different. Labeled target data is not required, while source-labeled data is necessary to train the model. This scenario also requires an adaptation step to consider the source-target domain gap. This scenario is often investigated in detail under the Domain Adaptation setting.

In this thesis, the usage of TL will always refer to the inductive case of TL. Instead of randomly initializing the model's parameters, we often transfer some knowledge from a related source task by initializing a relatively large portion of model's parameters from a model trained to solve the source task on the source domain[1] While still requiring ample target-labeled data, this initialization scheme often helps in reducing the amount of labeled data needed to train the model, resulting in faster convergence and more stable training.

[1] In image segmentation, for instance, the model consists of two submodules: an encoder or a feature extractor and a decoder. The encoder is often initialized from a pre-trained classification model trained on a given source classification task, such as ImageNet, while the decoder is often initialized randomly and trained from scratch.

3.1.2 A Related Setting: Few-Sample Fine-Tuning

If the source and target tasks and domains are sufficiently similar, TL can be a simple and effective method to overcome the scarcity of labeled data on target by transferring knowledge from a related source task with readily available labels. In recent years, with the introduction of large-scale standardized datasets (*e.g.*, ImageNet Deng et al. 2009, Kinetics Carreira et al. 2017, COCO T.-Y. Lin et al. 2014, or WMT16 Bojar et al. 2016) and the availability of pre-trained models on such datasets in open source access[2], the TL procedure has become very popular and almost synonymous with the standard SL approach.

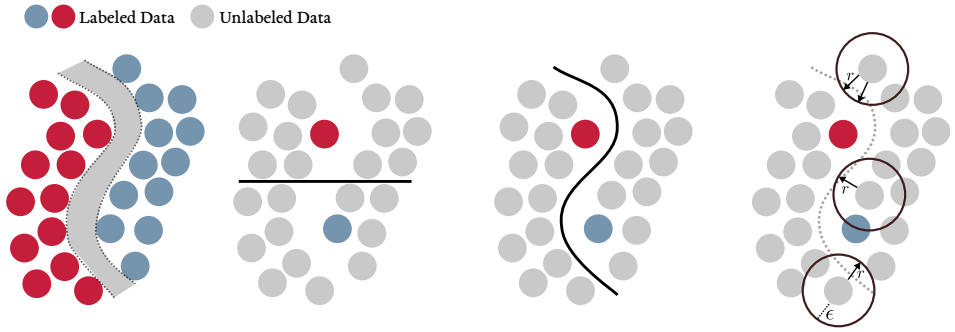
However, if the target and source tasks are sufficiently different (*e.g.*, image classification and image segmentation) or the domain gap between the source and target is considerable (*e.g.*, from synthetic to real images), the TL scheme[3] fails and can even lead to performance degradation. This phenomenon is known as negative transfer (Z. Wang et al. 2019). In this case, we fall back to the standard SL setting, requiring significant supervision to obtain a good model. Additionally, even in cases where TL might be helpful, if the amount of labeled data is minimal, the classic fine-tuning optimization process can be very unstable and sensitive to the different hyperparameters, often leading to degenerate models (Dodge et al. 2020), and making it insufficient to obtain good performances. In this thesis, we refer to this scenario as a Few-Sample Fine-Tuning (FSFT) setting, and set to develop a stable and well-performing fine-tuning method that better leverages the knowledge transferred from the source but with minimal target supervision.

Key Takeaways

- In SL, we are provided with a sufficient amount of labeled training data that consists of input-label pairs corresponding to the task we want to solve.
- The model is often trained with a CE loss with this large labeled training data and is then evaluated on unseen test data from the same distribution.
- In cases where a pre-trained model on a related source task and from a relatively similar source domain is available, we can follow the TL paradigm and transfer the knowledge from such a source to the desired target. This is done by initializing part or all of our model's parameters from a pre-trained version. It often results in faster convergence, more stable training during fine-tuning, and a reduction in the amount of labeled data required.
- Under the TL paradigm, and in cases where the source and target tasks and domains are sufficiently different, and the labeled data target is minimal, the standard fine-tuning method becomes insufficient and requires adjustments to adapt it to such a setting, which we refer to as FSFT.

[2] For example: `https://pytorch.org/serve/model_zoo.html`

[3] More specifically, the inductive TL scheme in which: i) the source and target tasks are different, *e.g.*, language modeling and text classification and ii) we have sequential access to labeled data in both the source and target domains.



(a) Low density region (b) Supervised Decision Boundary (c) Semi-supervised Decision Boundary (d) Adversarial Perturbation

Figure 3.1: Effect of SSL on the learned decision boundary. An illustration of the different learned decision boundaries. The red and blue circles represent labeled data points of different classes. The gray circles represent unlabeled data points. (a) The two classes are separated by a low-density region illustrated in gray, providing additional clearance when placing the decision boundary while maintaining optimal performance. (b) The learned decision boundary if only labeled examples are used. (c) The learned decision boundary when unlabeled data are leveraged using a semi-supervised technique. (d) An example of an adversarial perturbation over the unlabeled data points. From all the ϵ spheres of possible transformations to be applied to the unlabeled data, the adversarial perturbation r is chosen to alter the example class by crossing the decision boundary, *i.e.*, in the direction of the decision boundary.

Usage

In this thesis, our objective is to introduce methods that can produce well-performing models similar to those that could be obtained using SL but under the constraint of limited labeled data. Additionally, we will often use the SL or TL approaches as starting points for developing new methods or as baselines when evaluating them. For instance, in Chapter 5, we will use SL as our baseline in an SSL paradigm. In Chapter 8, we will cast the few-shot classification problem as a TL one and set to develop a better pre-training objective.

3.2 Semi-supervised Learning

In Semi-supervised Learning (SSL), we are provided with a dataset containing labeled and unlabeled examples to solve a task of interest. The portion of labeled examples is usually small compared to the unlabeled examples (*e.g.*, 1% to 10% of the total number of examples). The goal is to extract an additional and useful learning signal from the unlabeled examples to obtain a better generalization. This setting's training and testing data come from the same domain, *i.e.*, a single domain setting.

More formally, given a training dataset $\mathcal{D}^{\text{tr}}$, divided into a labeled $\mathcal{D}_l^{\text{tr}}$ and an unlabeled $\mathcal{D}_u^{\text{tr}}$ subsets, the objective is to leverage the unlabeled set to train a better-performing model than the one obtained using only the labeled portion with the SL baseline, and hopefully, get closer to the optimal performance that could be obtained if the entirety of the training dataset was labeled. In

terms of the learning objective, the goal of SSL is to leverage the unlabeled data $\mathcal{D}_u^{\text{tr}}$ to produce a better prediction function f than what could have been obtained using the labeled data $\mathcal{D}_l^{\text{tr}}$ only. For instance, $\mathcal{D}_u^{\text{tr}}$ might provide us with additional information about the structure of the data distribution $p(\mathbf{x})$ to better estimate the decision boundary between the different classes. As shown in Fig. 3.1 (a-c), when the data points with distinct labels are separated with a low-density region, leveraging unlabeled data with an SSL approach can provide us with additional information about the shape of the decision boundary between two classes, thus reducing the ambiguity present in the supervised case in which only the labeled examples are used.

3.2.1 Main Assumptions.

Before introducing the popular SSL approaches, the first question we need to answer is under what assumptions or conditions can we apply SSL algorithms? SSL algorithms are only effective if some assumptions about the data structure hold for both the unlabeled and labeled sets. Otherwise, without such assumptions, it would not be possible to extract any meaningful training signal from the unlabeled examples and learn a better model that generalizes well. The main assumptions in SSL are as follows:

- **The Smoothness Assumption.** If two input data points $\mathbf{x}$ and $\mathbf{x}'$ that reside in a high-density region are close, so should their corresponding outputs y and y' (Chapelle et al. 2009). It means that if two inputs are near each other in a high-density region of the input space, then their corresponding outputs cannot be arbitrarily different from each other.
- **The Cluster Assumption.** If two points belong to the same cluster, they are likely to be of the same class (Chapelle et al. 2009). It can be viewed as a special case of the smoothness assumption, where we suppose that input data points form clusters, and each cluster corresponds to one of the desired output classes.
- **The Manifold Assumption.** The high-dimensional data lies roughly on a low-dimensional manifold (Chapelle et al. 2009). If our input data lies on some lower-dimensional manifold within the high dimensional space, we can try to discover such low dimensional representation using the unlabeled data, then use the labeled data to solve a simplified learning task in the lower dimensional space.

3.2.2 Dominant Approaches.

Under one of these assumptions, many SSL methods and approaches have been introduced over the years to leverage the unlabeled examples and obtain better performances compared to the supervised baseline. These algorithms can be broadly divided into the following categories:

- **Consistency Training or Consistency Regularization.** Such methods (Laine et al. 2016; Miyato et al. 2018; Verma et al. 2022; Xie et al. 2020) are based on the conjecture that if a realistic perturbation[4] is applied to an unlabeled data point, the model's prediction on this

[4] Realistic perturbations are the set of perturbations that can occur naturally during the data collection, processing and transmission steps, *e.g.*, different lighting conditions, various orientations, and viewing angles, or a small amount of Gaussian noise.

perturbed version should not change significantly compared to the clean one. The model can then be trained to have consistent predictions on unlabeled and perturbed examples.

- **Pseudo-Labeling.** Such methods (D.-H. Lee 2013; Pham et al. 2021; Shi et al. 2018; Sohn et al. 2020) leverage a trained model on the labeled set and use it as a labeling function to produce additional training examples by labeling instances of the unlabeled set. If these pseudo-labeled examples satisfy a given set of heuristics (*e.g.*, threshold based selection), they can be added to the existing labeled training set, and the model can then be fine-tuned using this newly expanded set.
- **Generative Models.** Such methods (Kingma et al. 2014b; A. Kumar et al. 2017; Odena 2016) are based on the assumption that if a generative model is capable of generating realistic images from the data distribution $p(\mathbf{x})$ learned based on unlabeled data, then it must also learn useful and transferable features that can be leveraged to solve a discriminative task using the labeled examples.
- **Graph-Based SSL.** Such methods (Iscen et al. 2019; B. Jiang et al. 2019; Subramanya et al. 2014; Z. Yang et al. 2016; Zhu 2005) consider the labeled and unlabeled data points as nodes of a graph. They then focus on introducing methods to propagate the labels from the labeled nodes to the unlabeled ones on the constructed graph. For instance, label propagation can be based on the similarity between two nodes $\mathbf{x}$ and $\mathbf{x}'$, which representations the edge weight between them, to assign labels to the unlabeled nodes.

In this thesis, we focus on Consistency Training and Pseudo-Labeling based methods since they use similar architectures and training procedures as the supervised case, with only minor adjustments to leverage the unlabeled data $\mathcal{D}_u^{\text{tr}}$. Broadly, while sharing a common framework, such methods differ in their loss formulation by using an additional unsupervised auxiliary loss to be computed over the unlabeled examples. In this case, the total loss $\mathcal{L}$ consists of such an unsupervised loss $\mathcal{L}_u$, in addition to the traditional supervised loss $\mathcal{L}_s$ computed only over the labeled set $\mathcal{D}_l^{\text{tr}}$. The standard CE loss is often chosen as the supervised loss as formulated in Eq. (3.1). The total training loss $\mathcal{L}$ is then defined as a simple summation of the two losses, with an unsupervised loss weighting $\omega_u \in \mathbb{R}^+$ to control the contribution of $\mathcal{L}_u$. As such, the total loss to be minimized can be computed as follows:

$$\mathcal{L} = \mathcal{L}_{\text{s}} + \omega_u \mathcal{L}_u = \frac{1}{|\mathcal{D}_l^{\text{tr}}|} \sum_{\mathbf{x}_i, y_i \in \mathcal{D}_l^{\text{tr}}} \mathcal{L}_{\text{CE}}(y_i, f(\mathbf{x}_i)) + \omega_u \mathcal{L}_u \tag{3.2}$$

3.2.3 Consistency Training

Consistency Training based-SSL methods favor inference functions that give consistent predictions over similar unlabeled data points. Rather than minimizing the classification cost at the zero-dimensional data points of the input space, the model is trained to minimize the cost on a manifold around each data point. It thus pushes the decision boundaries away from the unlabeled data points and smoothing the manifold on which the data reside (Zhu 2005). Concretely, in its simplest form, given an unlabeled data point $\mathbf{x}_i \in \mathcal{D}_u^{\text{tr}}$ and a slightly perturbed version of it $\tilde{\mathbf{x}}_i$, the unsupervised loss measures the distance $\mathbf{d}$ between the two corresponding outputs produced

by a given inference function f, *i.e.*, $\mathbf{d}(f(\mathbf{x}_i), f(\tilde{\mathbf{x}}_i))$. Therefore, given two outputs $f(\mathbf{x}_i)$ and $f(\tilde{\mathbf{x}}_i)$ in the form of a probability distribution over the C classes, we consider $f(\mathbf{x}_i)$ as the target and force $f(\tilde{\mathbf{x}}_i)$ to be similar to it. In this case, the loss $\mathcal{L}_u$ to be minimized is computed as follows:

$$\mathcal{L}_u = \frac{1}{|\mathcal{D}_u^{\text{tr}}|} \sum_{\mathbf{x}_i \in \mathcal{D}_u^{\text{tr}}} \mathbf{d}(f(\mathbf{x}_i), f(\tilde{\mathbf{x}}_i)) \tag{3.3}$$

In this case, the distance measure $\mathbf{d}(.,.)$ is often chosen as either a Mean Squared Error (MSE) or a Kullback-Leibler (KL) divergence, defined as follows:

$$\begin{aligned} \mathbf{d}_{\text{MSE}}(y, y') &= \frac{1}{C} \sum_{c=1}^{C} (y_c - y'_c)^2 \\ \mathbf{d}_{\text{KL}}(y, y') &= \sum_{c=1}^{C} y_c \log \frac{y_c}{y'_c} \end{aligned} \tag{3.4}$$

with y and $\hat{y}$ as any two output probability distributions over the C desired classes, and y_c and $\hat{y}_c$ as the probability of predicting class $c \in \{1, \ldots, C\}$. Note that the loss in Eq. (3.3) can also take different forms based on the same concept. For instance, the perturbed output can be obtained using perturbed features instead of a perturbed input. Additionally, instead of using the clean output as the target, the perturbed output can be considered as the target, *i.e.*, $f(\tilde{\mathbf{x}})$ as the target and force the clean prediction $f(\mathbf{x})$ to be similar to it.

Furthermore, under this framework, the various consistency training-based SSL methods differ along two axes: i) the choice of the perturbation method applied to the unlabeled examples and ii) how the target for a given unlabeled input is obtained. In the following, we list some possible choices for both of these two variables:

- Perturbation method. In terms of the choice of perturbations, Ladder Network (Rasmus et al. 2015) injects Gaussian noise into the intermediate features instead of the inputs, while Virtual Adversarial Training (VAT) (Miyato et al. 2018) tries to make the predictions invariant to small transformations by choosing additive perturbations r in the adversarial direction of a given unlabeled input, *i.e.*, so that the predicted class changes as illustrated in Fig. 3.1 (d).

- Target generation. For the generation of the targets, the proposed methods try to improve over the the naive approach of using the model's output as a target, *i.e.*, $f(\mathbf{x}_i)$, to avoid noisy targets that change from one iteration to the other. Temporal Ensembling (Laine et al. 2016), for example, proposed to maintain an Exponential Moving Average (EMA) of the targets for more stable training. Specifically, for each unlabeled training sample $\mathbf{x}_i$, we maintain a EMA of its prediction noted y_i^{ema}, then, at a given training iteration, y_i^{ema} is updated using the model's current prediction $f(\mathbf{x}_i)$ as $y_i^{\text{ema}} \leftarrow \alpha y_i^{\text{ema}} + (1 - \alpha) f(\mathbf{x}_i)$ which is then used as the target, with α as a momentum term that controls how far the current targets reach past target values. On the other hand, Mean Teacher (Tarvainen et al. 2017) maintains an EMA of the model's weights instead of the outputs. Specifically, with θ as the model's weights at the latest training iteration, the EMA version of its weights is

computed as $\theta^{\text{ema}} \leftarrow \alpha\theta^{\text{ema}} + (1-\alpha)\theta$. The target for a given unlabeled data point can then be obtained using this EMA version of the model (*i.e.*, $f_{\theta^{\text{ema}}}(\mathbf{x})$) and used to compute the unsupervised loss.

3.2.4 Pseudo-Labeling

Pseudo-Labeling (D.-H. Lee 2013; Shi et al. 2018), and related methods such as Self-Training (McClosky et al. 2006; Riloff 1996), are the class of SSL algorithms that produce pseudo-labels (*i.e.*, automatically generated ground truths) for the unlabeled data points. These pseudo-labels can then be used to train a new version of the prediction function over both the original labeled set and the unlabeled examples with newly generated pseudo-labels following the standard CE-based supervised training. These pseudo-labels are often generated using the prediction function itself, either online where we pick the class with the maximum probability at each training iteration (D.-H. Lee 2013) and use it as a pseudo-label simultaneously, or sequentially where we first train the model on the labeled set, use it to generate the pseudo labels, and then re-train the model with both the labeled set and the newly labeled portion of the unlabeled set. They can also be generated with more elaborate methods such as neighborhood graphs (Iscen et al. 2019) or with models trained on different data views as in Multi-View learning (A. Kumar et al. 2011). However, while these pseudo-labels provide some additional and useful training information, the performance might degrade if the produced labels are very noisy and do not reflect the ground truth. As a result, many methods (Rizve et al. 2021; Shi et al. 2018) try to add a filtering step to only use the confident and clean predictions as pseudo-labels. Precisely, in its simplest and most popular form, pseudo-labeling consists of using the inference function f itself (*i.e.*, often initially pre-trained on the small labeled set $\mathcal{D}_l^{\text{tr}}$) to assign pseudo-labels to the unlabeled data points $\mathbf{x}_i \in \mathcal{D}_u^{\text{tr}}$ provided that the model's predictions pass some filtering step. Specifically, given a prediction $f(\mathbf{x}_i)$ corresponding to an unlabeled data point $\mathbf{x}_i$ in the form of a probability distribution over the C classes we are interested, the pair $\{\mathbf{x}_i, \text{argmax}(f(\mathbf{x}_i))\}$[5] is considered a newly labeled data point if some heuristic is met, *e.g.*, the probability assigned to the most likely class is higher than a predetermined threshold τ. The unsupervised loss can then be computed using the standard CE loss over the unlabeled examples that satisfy the chosen heuristic:

$$\mathcal{L}_u = \frac{1}{|\mathcal{D}_u^{\text{tr}}|} \sum_{\mathbf{x}_i \in \mathcal{D}_u^{\text{tr}}} \mathbb{1}_{\max(f(\mathbf{x}_i)) > \tau} \, \mathcal{L}_{\text{CE}}(\text{argmax}(f(\mathbf{x}_i)), f(\mathbf{x}_i)) \tag{3.5}$$

with $\mathbb{1}_{\text{cond}} \in \{0, 1\}$ as an indicator function evaluating to 1 iff cond is satisfied, and where the application of the $\text{argmax}(.)$ operation over the output probabilities results in a valid one-hot probability distribution where all of its mass is assigned to the class with the maximum probability. Note that pseudo-labeling with its use of hard labels (*i.e.*, one-hot vectors) is similar to that of entropy minimization (Grandvalet et al. 2005). In both cases, the network is forced to output low-entropy (*i.e.*, confident) predictions. While simple and often effective, the main downside of

[5] Here, we abuse the argmax notation, and follow previous works by considering that $\text{argmax}(\hat{y})$ results in a one-hot vector, with 1 set to the class $c_{\max}$ with the maximal assigned probability, *i.e.*, $c_{\max} = \underset{1 \leq c \leq C}{\text{argmax}}(\hat{y}_c)$ with $\hat{y}_c$ as the probability assigned to class $c \in \{1, \ldots, C\}$.

such methods is that the model is unable to correct its own mistakes, and any biased and wrong classifications can be quickly amplified, resulting in confident but erroneous pseudo-labels.

3.2.5 A Related Setting: Weakly-Supervised Learning

In Weakly-Supervised Learning (WSL), we are provided with weaker and coarse annotations instead of a ground-truth labeled training set with the exact fine annotations we desire. These annotations could come from crowd workers, be the output of heuristic rules, or be the result of distant supervision (Mintz et al. 2009). The objective, in this case, is to extract a training signal useful for the actual desired task with fine predictions. For instance, for the task of image segmentation (*i.e.*, assigning a class to each pixel of the input image), weakly-supervised segmentation methods consist of leveraging a weaker level of supervision (*e.g.*, image-level annotations) to train the segmentation network instead of the expensive per pixel (*i.e.*, pixel-wise) annotations. Such weak labels can come in different forms, such as bounding boxes (J. Dai et al. 2015b), scribbles (D. Lin et al. 2016), points (Bearman et al. 2016) or image-level labels (Papandreou et al. 2015a). Bounding boxes are the conventional representation of object positions and their classes used for object detection. Scribbles refer to earmarking each type of semantics as a mark. Points imply the object's central location in the image. Image-level labels are simply the category IDs of the objects in the input image. Among all these different levels of supervision, image-level supervision remains the weakest one, in which the labels can be obtained more easily and efficiently than the rest. Using image-level supervision, related methods generally involve two consecutive steps:

- Generation. The weakly labeled examples are first used to generate sparse and often imprecise pixel-level labels, which are then refined to correct the wrongly labeled regions, resulting in cleaner and more precise fine-grained pseudo-pixel-level labels.
- Training. The generated and refined pseudo labels can then be used to train a segmentation network following the standard CE-based SL setting.

Note that the weakly labeled set $\mathcal{D}_w^{\text{tr}}$ can also be accompanied with a smaller finely annotated set $\mathcal{D}_l^{\text{tr}}$, *i.e.*, $\mathcal{D}^{\text{tr}} = \mathcal{D}_l^{\text{tr}} \cup \mathcal{D}_w^{\text{tr}}$. It thus makes such a version of WSL closely related to the setting of SSL where the unlabeled set $\mathcal{D}_w^{\text{tr}}$ is replaced with a weakly labeled set $\mathcal{D}_w^{\text{tr}}$. In this setup, we can combine SSL and WSL approaches to extract a better training signal from the weakly labeled set by generating fine pseudo-labels in addition to using an SSL approach using the inputs only without their weak labels. In Section 5.4.4, we investigate this scenario and show that it leads to better results.

Key Takeaways

- In SSL, the training set we are provided with consists of a small labeled set and a larger unlabeled set, both from the same distribution.
- The goal of SSL is to leverage the available unlabeled data to obtain better-performing models than what could have been obtained using only the labeled set.
- The two lines of work relevant to us under the SSL setting are consistency training and pseudo-labeling methods.

- In consistency training, the model is trained on a given unlabeled data by forcing it to have consistent predictions, with and without some perturbation. The decision boundary is thus forced to lie in low-density regions of the input space.
- In pseudo-labeling, the model itself is used to generate pseudo-labels on the unlabeled data. The newly generated pseudo-labeled data points are then added to the training set to train the model further.
- In WSL, we are provided with a weakly labeled set, *i.e.*, the provided labels do not correspond to the desired targets, and the objective is to leverage this weak set to train a model capable of solving the desired task.

Usage

In Chapter 5, we consider the setting of SSL and develop a new approach for image segmentation. Specifically, we introduce Cross-Consistency Training (CCT), a consistency-based SSL approach tailored for the image segmentation task. Additionally, we use a form of pseudo-labeling to leverage weakly-labeled data (*i.e.*, image-level labels) in cases where they are available to improve the performance of the proposed CCT approach. In Chapter 7, we propose integrating consistency training into a UDA framework and show that this leads to better performances.

3.3 Unsupervised Learning

Unsupervised Learning (UL) can be generally defined as the set of methods attempting to extract useful information from the data without any explicit supervised signal. In this work, we differentiate between two lines of UL work:

- Generative UL. It aims to extract useful information from the underlying data distribution.
- Discriminative UL with two possible objectives:
  - Clustering: directly predicting class assignments that correspond to some semantically meaningful aspects of the data.
  - Representation Learning: learning useful and general-purpose representations that can be used on a downstream task of interest.

Alternatively, these two types of UL can also be categorized based on their learning objective. In unsupervised generative methods, we want to learn a maximum likelihood estimation of the true data distribution. In unsupervised discriminative methods, we want to learn a one-to-one mapping for solving some unsupervised prediction task, which can be defined based on either a clustering or a representation learning objective.

3.3.1 Generative UL

In deep generative UL, *a.k.a*. deep generative modeling, we are primarily interested in a parametric approximation of the data distribution, which summarizes all the information about a dataset $\mathcal{D}$ with a finite set of parameters. For instance, given access to a dataset $\mathcal{D}$ drawn from some underlying and unknown distribution p_{data}, the objective is to learn a generative model with parameters θ within the space of possible models, such that the model p_θ[6] provides a good approximation of the data distribution p_{data}. Formally, with p_{data} as the data distribution accessed through the empirical data $\mathcal{D}$ and $\mathbf{d}$ as a distance measure between two probability distributions, the training objective is defined as follows:

$$\min_{\theta} \mathbf{d}(p_{\text{data}}, p_\theta) \tag{3.6}$$

During inference and with a learned joint distribution over the entire data, generative models can be used for either i) density estimation, *i.e*., given an example $\mathbf{x}$, we can approximate its likelihood of being drawn from the same data distribution p_{data} as the training samples with $p_\theta(\mathbf{x})$, a measure that can be useful in detecting outliers and other data anomalies. ii) sampling, *i.e*., we can generate novel examples from the model distribution $\mathbf{x}_{\text{new}} \sim p_\theta$, or iii) feature extraction, while not their primary usage of such models, we can also use them as feature extractors to extract useful representations of a given example $\mathbf{x}$.

In this thesis, we take inspiration from the structure of a specific and popular type of generative models called autoregressive models (X. Chen et al. 2018; Salimans et al. 2017; Van den Oord et al. 2016) that are applied to the image domain. With an autoregressive model, the joint probability distribution of high-dimensional data, or more specifically images, is decomposed as a product of conditionals over its sub-components (*i.e*., the pixels) for tractable likelihood computation. Concretely, the joint probability $p(\mathbf{x})$ of an image $\mathbf{x}$ is factorized as a product of conditionals over its n pixel x_i as follows:

$$p(\mathbf{x}) = \prod_{i=1}^{n} p(x_i | x_1, \ldots, x_{i-1}) \tag{3.7}$$

The model $p(x_i|x_1, \ldots, x_{i-1})$ is then trained to predict the current pixel x_i based on the past values $x_{\leq i-1}$ in a raster scan fashion (*i.e*., left-to-right and top-down) using masked convolutions (Van den Oord et al. 2016) as illustrated in Fig. 3.2. After being trained, the model can either be used for likelihood estimation by computing the product of conditions, or for generation by producing new images one pixel at a time, starting from pixel x_0 up to x_n, and at each time step, the current pixel x_i being generated is conditioned on the previously generated pixels.

In terms of specific implementations of these autoregressive models, there is a large body of work (X. Chen et al. 2018; Salimans et al. 2017; Van den Oord et al. 2016) proposing various modeling methods in the domain of natural images. The standard PixelCNN model (A. v. d. Oord et al. 2016; Van den Oord et al. 2016) specifies the conditional distribution of a sub-pixel, or color channel of a pixel, as a full 256-way softmax. While PixelCNN++ (Salimans et al. 2017) models the conditional

[6] For a more accurate notation, in generative modeling, we often denote the learnable function as p_θ instead of f_θ, since the model p_θ can be viewed as a probability density function or a probability mass function on the input space.

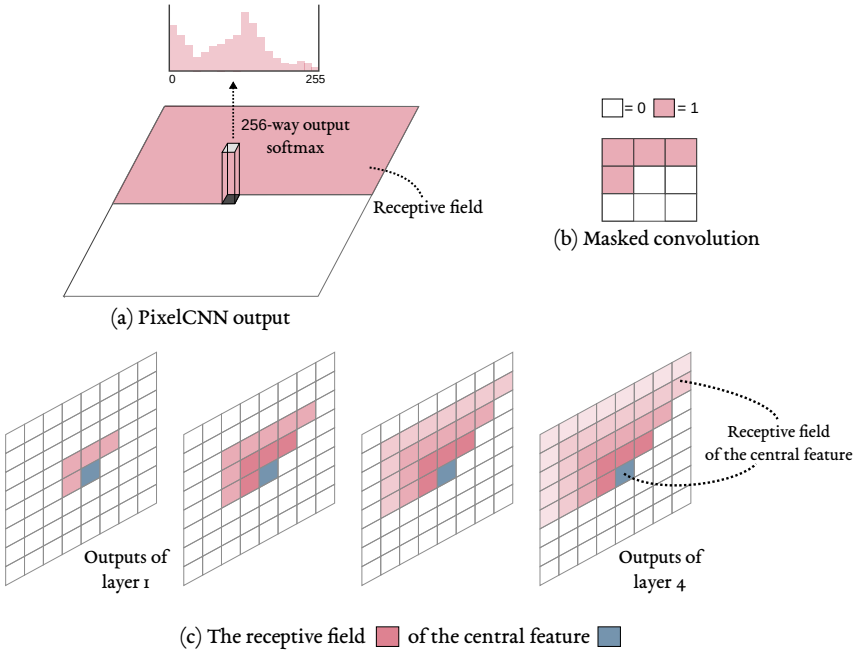


Figure 3.2: An example of an autoregressive model. An illustration of some components of the autoregressive model PixelCNN. (a) The outputs of PixelCNN at a given spatial location. The model predicts the color information of the input pixel at a given location based solely on the pixels to the left and above its position, illustrated in light red. In this case, the output layer is a 256-way softmax assigning a probability over all possible 8-bit colors for each channel separately. (b) An example of 3×3 kernel of a masked convolution operation. The masking maintains the correct dependencies between the inputs and the outputs. (c) By applying a series of convolutions with the causal masking shown in (b), we obtain the desired receptive field where each feature only depends on the input pixels at positions above it and to the left of it.

distribution as a mixture of logistics. In both cases, causal convolutions are used to process the initial image $\mathbf{x}$ in an autoregressive manner. In contrast, Image (Parmar et al. 2018) and Sparse (Child et al. 2019) transformers use self-attention (Vaswani et al. 2017) over the input pixels. While PixelSNAIL (X. Chen et al. 2018) combines both attention and masked convolutions for a larger receptive field and more efficient likelihood estimation.

3.3.2 Discriminative UL

Clustering

Clustering aims to group semantically similar data into the same cluster based on some similarity measure. Although effective when applied to simple problems, conventional methods usually perform poorly on high-dimensional data due to the inefficiency of the similarity measures used in these methods, which often suffer from high computational complexity on large datasets. To overcome this, DL models can be used to transform the high-dimensional input data into a lower dimensional space, in which the representations are more clustering-friendly. For instance, Deep Clustering Network (P. Huang et al. 2014) combines an auto-encoder with the k-means algorithm.

First, the auto-encoder is pre-trained with a reconstruction objective, followed by joint optimization of both the reconstruction and the k-means-based clustering loss. Based on a similar framework, Deep Embedding Network (DEN) (B. Yang et al. 2017) proposes adding a locality-preserving constraint to preserve the original data's local structure.

In recent years, a growing line of work based on Invariant Information Clustering (IIC) (Ji et al. 2019) proposes end-to-end clustering methods based on Mutual Information (MI) maximization, in which the cluster assignments are directly learned without an intermediate dimensionality reduction step. Specifically, for a given input image $x \in \mathcal{X}$, the goal of IIC is to learn a an inference function f_θ that preserves what is in common between two different views or augmentations of the input $\mathbf{x}$ while discarding instance-specific details. In terms of the optimization objective, this can be achieved by maximizing the MI I between the model's predictions $f_\theta(\mathbf{x}_1)$ and $f_\theta(\mathbf{x}_2)$ as follows:

$$\max_\theta I(f_\theta(\mathbf{x}_1); f_\theta(\mathbf{x}_2)) \tag{3.8}$$

where the two predictions $f_\theta(\mathbf{x}_1)$ and $f_\theta(\mathbf{x}_2)$ are in in the form of a probability distribution over the K desired clusters and with $\mathbf{x}_1$ and $\mathbf{x}_2$ as two augmented versions of the input $\mathbf{x}$. In Chapter 6, we will present this MI maximization-based learning objective in more detail and propose an adaptation of it for the task of unsupervised image segmentation.

Representation Learning.

In representation learning (Bengio et al. 2013), the main goal is to train a model, or more accurately a feature extractor, that maps the raw input data into a feature space that only captures the useful concepts and patterns in a compact representation. These learned features should contain sufficient knowledge to solve a wide variety of downstream tasks while disregarding the rest of the irrelevant information. From the view of generalization, its goal is to learn general-purpose features that generalize well to several diverse and a priori unknown downstream tasks.

In this context, DL-based representation learning methods often consist of reformulating the task of learning useful features as a discriminative task. It consists of solving a pretext or a surrogate task that requires some complex understanding of the inputs. As a result of surrogate solving this task, we hope that the learned features will be helpful in solving the desired downstream tasks. This way, starting from a fully unsupervised setting and by defining some prediction task where the labels are generated from the data itself, we fall back to a setting similar to that of SL. As a result, we often refer to this setting as *Self-Supervised Learning*. One of the main benefits of such a learning objective is the possibility of reusing popular and effective SL methods and techniques and applying them to raw data without any human annotations.

In recent years, self-supervised representation learning has become a popular research topic within the DL community following the downstream performances it demonstrated. This resulted in the introduction and investigation of a wide variety of surrogate tasks to determine the ones that give rise to the most valuable features downstream. We propose the following non-exhaustive categorization of these methods:

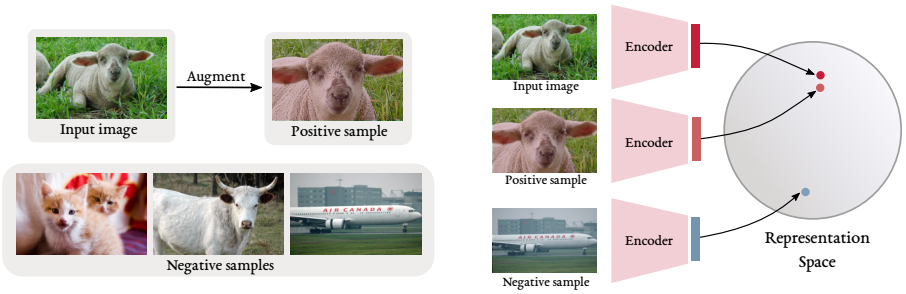
- **Input Completion.** Such tasks consist of masking or removing some portions or aspects of the inputs and tasking the model with predicting them. For instance, for the task of image colorization (Larsson et al. 2016; R. Zhang et al. 2016), the RGB images are first transformed

into Lab color space. The model then takes the lightness channel L as input and is trained to predict the color channels ab. Other examples are Masked Language Modeling (MLM) used in BERT (Devlin et al. 2019), where the model is tasked with predicting the masked input tokens, and image inpainting (Pathak et al. 2016), where the model must reconstruct the masked patch in the input image.

- **Predicting Transformations.** Another popular set of self-supervised tasks is the prediction of the transformation applied to the input. Such transformations can be simple geometric transformations such as 4 type of rotation resulting in a 4-way classification problem (*i.e.*, 4 possible rotations: $\{0°, 90°, 180°, 270°\}$), in which the model must predict the type of rotations applied to the input image (Gidaris et al. 2018c). Similarly, other tasks consist of predicting the relative position of one patch with respect to the others within the input image (Doersch et al. 2015), the type of permutation applied to the input patches (Noroozi et al. 2016), or the camera transformation between a pair of images (Agrawal et al. 2015).
- **Invariant Features.** Motivated by the fact that different inputs depicting the same objects or referring to similar concepts should have similar representations, these methods try to learn features invariant to superfluous aspects of the input while highlighting common and semantically rich patterns that are meaningful downstream. In an unsupervised setting, a simple and popular way to generate such superfluous aspects is the use of data augmentations (*e.g.*, rotation, translation, color jittering, cropping, or more complex combinations of these elementary transformations, such as AutoAugment (Cubuk et al. 2019a) and RandAugment (Cubuk et al. 2019b)). Instead of using them as a training regularizer, they are used to generate two or more augmented versions of a single input. The model is then tasked with producing similar representations of these inputs. This objective can be formulated as an N-way classification problem (Dosovitskiy et al. 2014), with N as the size of the data, where each original input represents its own class. The model can then be trained to assign the correct class to each input's augmented version. Another set of approaches (Bachman et al. 2019; K. He et al. 2020; Hjelm et al. 2018a; A. v. d. Oord et al. 2018a; Tian et al. 2019, 2020b), known as contrastive methods, drops the N-way classifier and optimizes the features directly, pulling features of similar inputs (*i.e.*, different augmentations of a given input) closer in the feature space while pushing away the dissimilar examples (*i.e.*, any two distinct examples). In this thesis, such methods are of interest to us and will be detailed below.

Contrastive Representation Learning.

Although many of the previously mentioned self-supervised representation methods demonstrated promising results downstream, contrastive representation learning methods remain the most competitive and best-performing methods for learning useful representations. As illustrated in Fig. 3.3, the objective in contrastive representation learning is to learn a function that maps semantically similar data points (*i.e.*, positive pairs) close together in the embedding space while pushing apart dissimilar points (*i.e.*, negative pairs). One of the major design choices in contrastive learning is how to select positive and negative pairs. The standard approach for generating positive pairs without human annotations is creating multiple views of each example. For instance, these views



(a) Sampling of postives and negatives

(b) Contrastive Learning

Figure 3.3: Contrastive Learning. A schematic view of visual contrastive representation learning. (a) First, we need to select a positive pair and a set of negative pairs before encoding them and computing the loss. Given an input image, its corresponding positive pair is a transformed version of itself in the fully unsupervised case, while the negatives are inputs sampled randomly from the dataset. (b) The contrastive learning objective can then be computed over the encoded features, pushing the model to produce similar representations for the positive pairs while simultaneously encoding the negatives differently.

can be generated by splitting the image into luminance and chrominance (Tian et al. 2019), applying different random crops and data augmentations (T. Chen et al. 2020; Z. Wu et al. 2018b), or using different patches within a single image (A. v. d. Oord et al. 2018a). Negative pairs, on the other hand, can be generated by randomly sampling images and patches from the rest of the dataset.

Formally, we consider a single input mini-batch $\mathcal{B} = \{\mathbf{x}, \mathbf{x}^+, \mathbf{x}_1^-, \mathbf{x}_2^-, \ldots\}$, consisting of an unlabeled input example $\mathbf{x}$ in addition to a $K+1$ corresponding inputs with a single positive sample $\mathbf{x}^+$ which is an augmented version of $\mathbf{x}$, and K negatives $\mathbf{x}_i^-, \forall i \in \{1, \ldots, K\}$ which are randomly sampled instances from the training dataset. Using an inference function f, which we refer to as our encoder since it encodes each example into a code or a feature vector, we obtain the representations of each example, *i.e.*, $f(\mathbf{x}) = \mathbf{z}$ for the original input. Similarly, we obtain the $(K+1)$ corresponding representations $\{\mathbf{z}^+, \mathbf{z}_1^-, \mathbf{z}_2^-, \ldots\}$ for the rest of the mini-batch. The contrastive objective then consists of maximizing the similarity between the input example's representation $\mathbf{z}$ and its positive $\mathbf{z}^+$, while minimizing its similarity with all of the other negatives $\mathbf{z}_i^-, \forall i \in \{1, \ldots, K\}$. Using cosine as our similarity measure (*i.e.*, we often consider that the representations are L_2 normalized, thus only requiring the computation of the dot product to measure the cosine similarity), the unsupervised contrastive loss $\mathcal{L}_u$ can be computed as follows:

$$\mathcal{L}_u = -\log \frac{\exp(\mathbf{z} \cdot \mathbf{z}^+ / \tau)}{\exp(\mathbf{z} \cdot \mathbf{z}^+ / \tau) + \sum_{i=1}^{K} \exp(\mathbf{z} \cdot \mathbf{z}_i^- / \tau)} \tag{3.9}$$

with τ as a temperature hyperparameter (Z. Wu et al. 2018b) that controls the relative importance of the distances between pairs of points. For instance, at low temperatures, the loss will be dominated by the small distances, while the distances between widely separated representations will be deemed almost irrelevant. Intuitively, this loss is the log loss of a $(K+1)$-way softmax-based classifier that tries to classify $\mathbf{x}$ as $\mathbf{x}^+$.

After training using the objective in Eq. (3.9), the model f learns rich and useful features that can then be transferred to downstream tasks, such as image segmentation, in which the model

can be fine-tuned to the desired performances using a reduced amount of labels. Note that for simplicity, the loss in Eq. (3.9) represents the loss computed for a single input $\mathbf{x}$. In practice, for a mini-batch of N examples, each input within it is augmented to generate its corresponding positive sample, resulting in a mini-batch of $2N$ examples. The total loss is then computed over all N examples, and each time, the $2N - 2$ remaining examples are considered as negatives.

Key Takeaways

- In UL, only unlabeled data is available, and the objective is to extract useful information from the data without any explicit supervision generated by human labor.
- In UL, we differentiate between generative UL, *i.e.*, extracting useful information from the underlying data distribution, and discriminative UL, *i.e.*, solving some intermediate tasks to leverage the provided unlabeled data.
- For generative UL, we focus on autoregressive models that try to model the data distribution with a tractable likelihood computation, in which a given data point is factorized as a product of conditionals over its sub-components (*e.g.*, an image and its pixels, respectively).
- For discriminative UL, we differentiate between a clustering objective, *i.e.*, predicting class assignments directly, and a representation learning objective, *i.e.*, learning useful and transferable features.
- For representation learning, we focus on contrastive methods that try to directly optimize the learned features so that two views of the same input have similar representations while also being distinct from the rest of the data samples.

Usage

In Chapter 6, we consider the UL paradigm and introduce a discriminative UL method for the task of image segmentation. Specifically, we will leverage the structure of autoregressive models and use them as view generators for both a clustering objective (*i.e.*, autoregressive clustering) and representation learning (*i.e.*, autoregressive representation learning). Additionally, in Chapter 8, we will use a novel contrastive learning-based objective for the task of few-shot classification (*i.e.*, to be introduced in the next section) to learn more general-purpose and transferable features.

3.4 Unsupervised Domain Adaptation

Despite the empirical success of SL, a critical limitation is often overlooked; the possible distributional mismatch between the labeled training set $\mathcal{D}^{\text{tr}}$ and the testing set $\mathcal{D}^{\text{test}}$. In the case of SSL, we can have the labeled $\mathcal{D}_l^{\text{tr}}$ and unlabeled $\mathcal{D}_u^{\text{tr}}$ sets drawn from different distributions. These two scenarios are often encountered in real-world scenarios and applications, making it a case worth studying. A popular framework that deals with such a scenario is Domain Adaptation (DA). As

Figure 3.4: An example of a UDA benchmark. In a UDA setting, we are often provided with a benchmark consisting of many domains containing the same classes but sampled from a relatively different data distribution. In this figure, we show examples from Office-Home dataset (Venkateswara et al. 2017) consisting of four domains: Art, Clip Art, Product, and Real World, with each one containing images depicting 45 everyday objects. When evaluating on such a benchmark, we often consider all possible pairs of source-target domains, *i.e.*, 6 possible pairs. We keep the source labels each time, disregard the target labels, and then report the accuracy obtained on the target examples at the end of training.

opposed to SL, in which the model is tested on examples drawn from the same data distribution as that of the labeled training set, domain adaptation takes into consideration this distributional mismatch and adapts the model to a different test distribution than the training distribution. Note that while the two distributions are different, they remain related and share some common factors, otherwise, the adaptation task will be infeasible since any form of learning on the training distribution would not be helpful at test time.

In DA, we consider two domains, a source domain D_s and a target domain D_t with different marginal probability distributions $p_s \neq p_t$. The goal is then to learn a well-performing model on the target since we postulate that the new examples to be encountered at test time will be more similar to those of the target domain. Under this definition, various subsets of DA deal with different variations of this problem. For instance, in the supervised case and at training time, we have access to labels in both the target and source domains. For the unsupervised version (*i.e.*, Unsupervised Domain Adaptation or UDA), which will be the focus of this thesis, we only have a labeled source domain but an unlabeled target domain. For an example of a UDA benchmark, see Fig. 3.4.

Considering this unsupervised variation (*i.e.*, UDA) of the DA problem, we would like to draw the reader's attention to its similarity with the (SSL) setting. In both cases, we have access to a training set in the form of labeled and unlabeled subsets. In UDA, the two sets are drawn from distinct data distributions, and the labeled set size is generally more significant than in the SSL case. Such a similarity might point us to the possibility of leveraging SSL methods and integrating them into the UDA framework to exploit the unlabeled examples and obtain better target generalization. We will explore this case in detail as part of one of our contributions in Chapter 6.

Now, given a labeled source and an unlabeled target under the UDA setting, the obvious question to be raised is how we can adapt the model to the target domain using the provided unlabeled

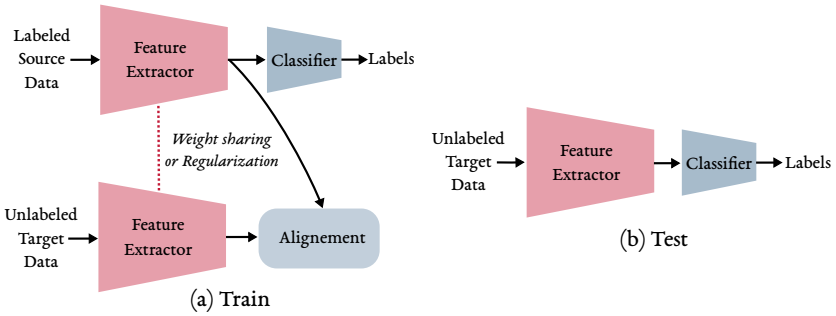


Figure 3.5: Domain-Invariant Representations. The general framework of domain-invariant representations based UDA methods. (a) During training, the whole model (*i.e.*, the feature extractor and the classifier) is trained on the source-labeled data with the standard CE loss, while the feature extractor is trained with an additional alignment loss to enforce an invariance of features across the two domains. (b) At inference, we fall back to the standard usage in which the model is used to classify the unseen examples from the target domain.

target sample, while also leveraging the source labels to learn a well-performing classifier. Largely, UDA methods solve this problem with two main types of approaches (Ruder 2019):

- **Representation methods.** Such methods (Ganin et al. 2015; Gretton et al. 2007; M. Long et al. 2015) try to cast both source and target data into a shared low-dimensional feature space containing the common factors and aspects of the two domains, so that a classifier trained on the source features will be directly usable on the target features, *i.e.*, learning *Domain-Invariant Representations*.
- **Weighting and Selection methods.** Such methods try to correct for the source-target distributional discrepancy by either re-weighting (J. Huang et al. 2006; J. Jiang et al. 2007; Sugiyama et al. 2008; Tsuboi et al. 2009) the source samples so that this discrepancy is minimized, or by applying a binary weighting (Plank et al. 2011; Remus 2012) where the possible harmful examples are completely ignored.

In this thesis, we focus on domain-invariant representation-based methods and detail the most popular ones in the following.

Domain-Invariant Representations

Methods that fall under this category try to solve the source-target discrepancy by enforcing an invariance of their representations. A feature representation can be considered domain-invariant if the features follow the same distribution regardless of whether the input comes from the source or target domain (H. Zhao et al. 2019). In this case, if a classifier can be trained to do well on source using domain invariant features, then such a classifier may generalize well to the target domain. However, such methods assume that a possible shared representation space exists and that the marginal label distributions are relatively similar in both domains. Otherwise, the adaptation fails.

As illustrated in Fig. 3.5, domain-invariant-based methods can be described using a shared framework. Precisely, the model consists of two components, a feature extractor (*e.g.*, a standard deep neural network) and a shallow classifier (*e.g.*, a linear classification layer). At training time,

in addition to training on the labeled source data following the standard CE-based SL approach, an alignment loss is also used to impose the invariance of the representations over the feature extractor. Then, at inference, the model can be used directly to infer the labels of the data coming from the target distribution. Under this framework, the popular UDA methods are:

- **Divergence.** A possible approach for aligning the source and target distributions is minimizing a divergence that measures the distance between them at the feature level. Examples of such divergence measures are Maximum Mean Discrepancy (MMD) (Gretton et al. 2007), correlation alignment (B. Sun et al. 2015) and the Wasserstein distance (Bhushan Damodaran et al. 2018).
- **Adversarial.** Another popular approach relevant to our work is adversarial alignment (Ganin et al. 2015). It consists of enforcing a source-target distributional alignment using a domain classifier in an adversarial setting. The domain classifier takes as input the output representations of the feature extractor and predicts whether these representations come from the source or the target domain. In this case, and similar to the Generative Adversarial Network (GAN) framework (Goodfellow et al. 2020), the domain classifier plays the role of the discriminator that tries to differentiate between the two distributions. During training, the domain classifier is trained to correctly classify the domain of the features, the feature extractor is trained to fool the domain classifier, *i.e.*, generating similar features for both domains, thus making the classifier incapable of distinguishing between them. With a sufficiently low adversarial loss, we can expect that the representations produced by the feature extractor are aligned across the two domains. Then, the domain classifier can be disregarded at test time, and the source classifier can be applied to the produced target features.

  Formally, we consider a model $f = g \circ h$ consisting of a feature extractor $h : \mathcal{X} \to \mathcal{Z}$ that maps the inputs into a feature space $\mathcal{Z}$, a classifier $g : \mathcal{Z} \to \mathcal{Y}$ that maps the features into the label space $\mathcal{Y}$, and a binary domain classifier $d : \mathcal{Z} \to [0, 1]$ that differentiates between the source and target features. We defining the source and target domains by their specific joint distributions of inputs $\mathbf{x}$ and labels y, noted $p_s(\mathbf{x}, y)$ and $p_t(\mathbf{x}, y)$, respectively. The typical domain adversarial training procedure consists of a min-max objective as formulated as follows:

$$\begin{aligned}\max_{d} \min_{h,g} (\mathcal{L}_s - \lambda \mathcal{L}_{\text{adv}}) = \mathbb{E}_{(\mathbf{x},y)\sim p_s} \mathcal{L}_{\text{CE}}(g(h(\mathbf{x}), y)) \\ + \lambda \mathbb{E}_{\mathbf{x}\sim p_s} \log d(h(\mathbf{x})) + \lambda \mathbb{E}_{\mathbf{x}\sim p_t} \log(1 - d(h(\mathbf{x})))\end{aligned} \tag{3.10}$$

  with $\lambda \in \mathbb{R}^+$ as a weighting hyperparameter that controls the contribution of the adversarial loss. Note that objective in Eq. (3.10) can be optimized by either: i) alternating between training the model f and the domain classifier d following the GAN framework, *i.e.*, training the model f to minimize the objective, then training the domain classifier d to maximize it, or ii) by negating the gradients of the domain classifier with a gradient reversal layer (Ganin et al. 2015) when performing backpropagation to update the feature extractor's weights.

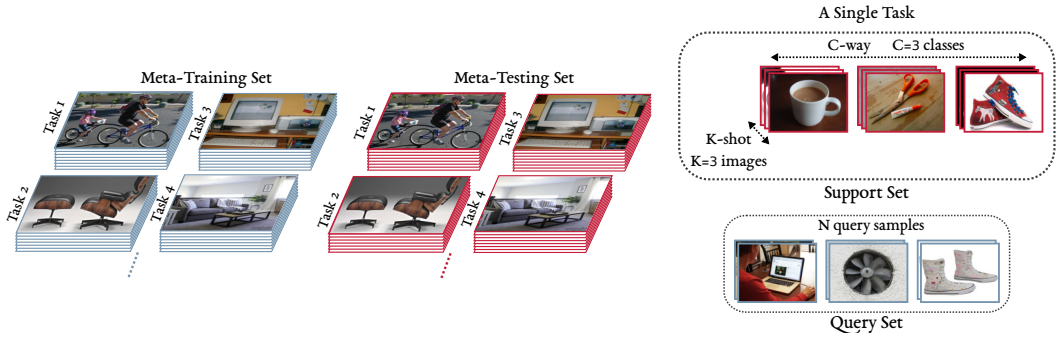


Figure 3.6: The structure of data in FSL. In FSL, we are provided with a meta-training and a meta-testing set, each containing a set of tasks. Each task consists of its training or support set containing $K \times C$ training examples, with K examples per each class of the C classes used to adapt the model to solve this task. In addition, we have a test or query set containing N test samples that are used to evaluate the adapted model on this task.

Key Takeaways

- In UDA, the training set consists of two sets coming from two distinct domains, a labeled set from a source domain and an unlabeled set from the target domain.
- In UDA, the objective is to leverage the labeled source to learn a well-performing model while simultaneously leveraging the unlabeled target so that the source-target distributional mismatch is also corrected, given that the end goal is to deploy the model on target test data.
- In this work, we focus on UDA methods based on learning domain-invariant representations. Such methods solve the source-target discrepancy by enforcing an invariance of their representations on some low-dimension feature space.

Usage

In Chapter 7, we consider the UDA paradigm and set to develop a better domain invariance-based method by taking inspiration from SSL methods. Specifically, we enforce a form of consistency regularization on the target domain and show that the proposed approach offers a notable performance gain when coupled with the appropriate representation alignment method.

3.5 Few-shot Learning

In all the learning paradigms we detailed above, we were always interested in solving a singular task, defined beforehand and a priori known. However, another important setting went unexplored, a case where the data as a whole is scarce, with only a few labeled training examples (*e.g.*, 1 or 5 examples per class) available. The objective at test time is to adapt our model quickly to a

new and unseen task at training time. In such a scenario, referred to as Few-shot Learning (FSL), the conventional DL paradigm of training the model to solve a novel task by optimizing an appropriate loss on the training set and then deploying the model on the test set becomes infeasible given the rarity of the data. In this case, the training goal becomes a meta or a high-level objective of equipping the model with the capability to quickly adapt to a new task using a few labeled training data points and limited training iterations. But how can we set up a training procedure to quickly equip the model to generalize to new and unseen tasks? And how can we then conduct the testing procedure to validate that the model demonstrates some degree of fast adaptability?

In FSL, the learning problem becomes a meta-learning or a learning-to-learn problem (C. B. Finn 2018). In previous settings, the training set consisted of a dataset $\mathcal{D}^{\text{tr}}$ sampled from a training data distribution p_{data} which is then used to train the model f to solve a singular and fixed task $\mathcal{T}$ using a proper loss function $\mathcal{L}$. In FSL, our model needs to be trained for fast adaptability over many tasks, so we consider a distribution over tasks $p(\mathcal{T})$ that we want our model to learn to adapt to and our training and testing set become meta-training and meta-testing sets of tasks sampled from $p(\mathcal{T})$. The meta-training set $\mathcal{I} = \{\mathcal{D}_t\}_{t=1}^{T}$ consists of several training tasks and each task consists of its own training set (*i.e.*, support set) $\mathcal{D}_t^{\text{tr}}$ and its testing set (*i.e.*, query set) $\mathcal{D}_t^{\text{test}}$. The meta-training stage then consists of training the model to effectively leverage the small task-specific training set $\mathcal{D}_t^{\text{tr}}$ so that its predictions on the corresponding test set $\mathcal{D}_t^{\text{test}}$ are correct. Then, for evaluation, the meta-testing step is conducted, where we are provided with a meta-testing set $\mathcal{S} = \{\mathcal{D}_q\}_{q=1}^{Q}$ consisting of new and unseen testing tasks which are also sampled from $p(\mathcal{T})$, and the model's performance is measured on the test set of each one of these tasks.

It is also worth noting that the taxonomy used in FSL can change depending on the modality and problem at hand. In computer vision, and specifically for classification problems, the FSL problem of interest is often defined by the number of classes and the number of labeled training samples per class that are provided in the training set $\mathcal{D}_t^{\text{tr}}$ for each mate-training task. For instance, a C-way K-shot classification problem, each training set $\mathcal{D}_t^{\text{tr}}$ contains $K \times C$ samples with C as the number of classes and K as the number of training examples per class as depicted in Fig. 3.6. In NLP, the FSL problem is often defined solely by the number of examples or demonstrations that are fed into the model for adaptation, *e.g.*, GPT 3 (Brown et al. 2020).

Regarding the learning procedure in FSL that equips the learner with a fast adaptation ability to novel tasks, the popular methods propose the following approaches:

- **Transfer Learning.** Rather than designing a learning algorithm specific to the FSL setting, transfer learning, or *learning-to-fine-tune* based methods, transform the FSL problem into a standard supervised TL problem, which is then solved following the pre-training, then fine-tuning pipeline. First, to cast the FSL training problem into an SL one, all of the meta-training tasks are merged into a singular task by combining their training set into a common one, *i.e.*, $\mathcal{D}^{\text{tr}} = \cup\{\mathcal{D}_1^{\text{tr}}, \ldots, \mathcal{D}_T^{\text{tr}}\}$. The meta-training stage then consists of pre-training our model, which consists of a feature extractor and a shallow classifier, on this merged training set following the standard CE-based SL training procedure. Then, at the meta-testing stage, the shallow classifier is removed. For a given unseen testing task, a newly initialized classifier is trained on top of the pre-trained feature extractor using this unseen task training set, and the adapted model is evaluated on its test set.
  Note that at test time, the pre-trained feature extractor can either be fine-tuned together

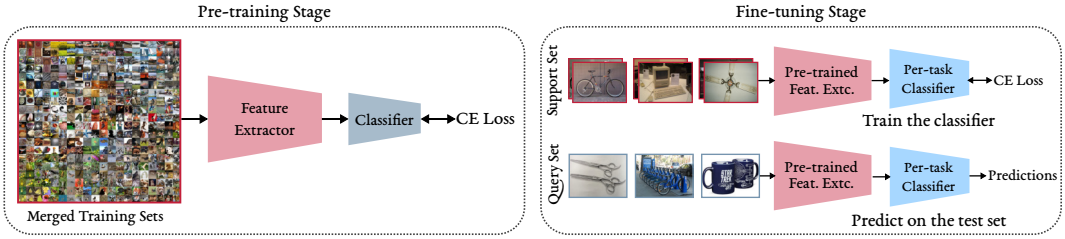


Figure 3.7: Transfer Learning based FSL methods. TL-based FSL methods cast the few-shot classification problem as a standard pre-train, then fine-tune problem. First, the training sets of all meta-training tasks are merged into a single training set is then used to pre-train the feature extractor following the standard CE-based SL approach. At test time and for a given novel test task, a newly initialized classifier is trained on top of the pre-trained feature extractor using the task's support set and is then used to produce the predictions over its query set.

with the classifier (Afrasiyabi et al. 2020; Dhillon et al. 2020), or fixed and used as a frozen feature extractor (W.-Y. Chen et al. 2019; Tian et al. 2020a). Additionally, the shallow classifier can take different forms, either a simple linear (*i.e.*, dense) layer (Tian et al. 2020a), or more elaborate implementations such as a cosine classifier (Gidaris et al. 2018a; Qi et al. 2018). We note that in this work, our contribution within the FSL framework presented in Chapter 8 will follow this line of work and try to improve upon it.

- **Meta-learning.** These approaches design learning procedures tailored specifically for the FSL problem. While there is not a clear definition of what makes an FSL approach a meta-learning one (Vinyals et al. 2016), if the predictions on the test (*i.e.*, query) set of a given task is explicitly conditioned on its corresponding training (*i.e.*, support) set, then the proposed method can be considered as a meta-learning approach. Given this definition, the various meta-learning based FSL methods can be distinguished via the strategies they propose to condition the query predictions on their corresponding support set (W.-Y. Chen et al. 2019). Specifically, we differentiate between the following two categories of meta-learning based FSL methods:
  - Metric learning based or *learning-to-compare* methods. With methods such as MatchingNet (Vinyals et al. 2016) and ProtoNet (Snell et al. 2017), the prediction on a given query sample is based on comparing the distance between its features and the features of each support sample. RelationNet (F. Sung et al. 2018) proposes a similar method but uses a relation module for sample comparison instead of a distance measure.
  - Optimization based or *learning-to-learn* methods. Methods such as MAML (C. Finn et al. 2017) explore optimizing the model for good initialization of its parameters that facilitate fast test-time adaptation. Thus, when a new task is encountered, its support set can be efficiently used to adapt these parameters to this new task with only a few gradient updates before using it to make a prediction on its query set.

Note that in this thesis, we limit ourselves to the standard setting of FSL, *i.e.*, inductive FSL. While the transductive FSL setting is a popular and growing line of work (Boudiaf et al. 2020; S. X. Hu et al. 2020; Y. Liu et al. 2018; Y. Liu et al. 2020; Qiao et al. 2019) in which we have access

to the unlabeled test (*i.e.*, query) sets at training time that can then be used to get substantial improvements in performances over inductive inference, in this work, we do not consider this setting, and any reference to FSL will always refer to the inductive setting.

Key Takeaways

- In FSL, the objective is to equip a model with the capability of fast adaptability to novel and previously unseen tasks with very few labeled training samples.
- In FSL, we are provided with a set of tasks divided into a meta-training set of seen tasks and a meta-testing set of unseen tasks. To solve each task, we are provided with its own task-specific training and testing sets. The first is used to adapt the model to this task, while the latter is used to evaluate the adapted model.
- The meta-training set is used to train the model and equip it with the required knowledge so that it can quickly adapt to the meta-testing task using only the task-specific training sets.
- In this work, we focus on the simple transfer learning based baseline for few-show classification. It transforms the meta-learning setting into a standard TL setup by merging the training set of all the meta-training sets. The model is then pre-trained on it, and at meta-testing, a per-task shallow classifier is trained on top of the pre-trained model to solve a given novel testing task.

Usage

In Chapter 8, we consider the FSL paradigm and set to introduce a novel few-shot classification method based on the transfer learning framework. Specifically, we will introduce a novel contrastive learning based pre-training objective to learn more transferable and general-purpose representations, thus making the model better at test-time adaptation on novel and unseen tasks. In Chapter 9, we consider a closely similar setup where we want to adapt a pre-trained model to a new setting using a limited labeled set, *i.e.*, FSFT, and similar to Chapter 8, we will rely heavily on transfer learning during the adaption.

Conclusion

In this second foundations' chapter (*i.e.*, Chapter 3), we introduced the learning paradigms that fall under the problem of learning under the constraint of limited labeled data.
Next, in the last foundations' chapter (*i.e.*, Chapter 4), we will introduce the tasks we consider, both their definitions in Section 4.1 and the models used to solve them in Section 4.2 Finally, the data used to train and evaluate our proposed methods is presented in Section 4.3.

CHAPTER 4

Tasks, Models, and Data

4 Tasks, Models, and Data

One of the benefits of the Deep Learning (DL) framework, as detailed in Chapter 2, is its universality. The overall scheme of defining the appropriate set of inductive biases, training the model, evaluating it, and deploying it remains the same regardless of the target application. However, the application or task we wish to solve has a great impact on the details of each step of this overall scheme and on the associated design choices. These choices range from the corresponding neural architecture, the loss function, the data used, the training procedure and the selection of the suitable evaluation metric to test the generalization capabilities of the model correctly. In this last foundations chapter, we present the details of the tasks, models, and data of interest to us, which are used in the contributions part of this work. First, we start in Section 4.1.1 by introducing some of the popular design choices for the different tasks we will address in the upcoming chapters, starting with their definition, their corresponding training pipelines and the metrics used for evaluation. The different neural architectures and benchmarks used for these tasks be later introduced in Section 4.2 and Section 4.3, respectively.

4.1 Tasks

4.1.1 Image Classification

The task of image classification consists of recognizing the objects and instances depicted in a given image by assigning it one or several category IDs from a set of predefined classes (Rawat et al. 2017). Despite its conceptual simplicity, it is considered as the backbone of computer vision upon which other popular tasks, such as localization, detection, and segmentation, are built. Additionally, given how natural and intuitive visual perception feels, it can be easy to overlook the complexity of visual data, making such a classification task daunting. Two images with different low-level statistics (*e.g.*, brightness and color values) can represent the same object (*e.g.*, various dog breeds). Conversely, with similar low-level statistics, the depicted objects can be different (*e.g.*, a dog and a cat). Such a semantic gap (Smeulders et al. 2000) highlights the difficulty of building a computer vision algorithm capable of producing the correct classifications.

The traditional methods attempted to solve this task with a dual-stage approach. First, by extracting a set of handcrafted features using feature descriptors such Scale-Invariant Feature Transform (SIFT) (Lowe 2004), Speeded Up Robust Features (SURF) (Bay et al. 2006) or Oriented FAST and Rotated BRIEF (ORB) (Rublee et al. 2011). The extracted features are then used to train a shallow classifier *e.g.*, a Support Vector Machine (SVM) classifier. However, while being relatively successful, such methods depend heavily on the extracted features' quality and the feature descriptor's design, therefore requiring a considerable amount of domain-specific feature engineering to obtain acceptable performances. Fortunately, with the resurgence of DL (Goodfellow et al. 2016) and its related techniques and architectures such as Convolutional Neural Networks

(CNNs) (LeCun et al. 1998), we have witnessed rapid progress in image recognition (K. He et al. 2016). It makes the task of image classification under normal conditions and with ample labeled examples (*i.e.*, SL setting) to be almost considered as a solved problem. The pipeline to follow to solve the classification task using a DL-based method is detailed next.

Pipeline

Compared to the traditional dual-stage approaches, the pipeline of DL-based methods used for solving image classification type tasks is relatively straight forward, and consists of the following steps:

- **Preprocessing.** The preprocessing step for image classification, which is also closely followed in many other vision tasks, generally consists of fetching the input images from storage, applying some data augmentation to each image (*e.g.*, resizing all images to the same size, cropping, and a random horizontal flip), followed by a normalization step for faster convergence. Then, the images are batched to construct a single mini-batching for parallel processing and are placed in the GPU for the next training step.
- **Training.** Overall, this step follows the standard DL training procedure. The input mini-batch is passed through the model for a single training iteration, where intermediate activations and the final outputs are computed in the forward pass. Then, the loss is computed using the predictions and the ground truths. The gradients of the model's weights with respect to the loss are evaluated during the backward pass, and the weights are finally updated using the selected neural network optimizer.
- **Postprocessing.** For image classification, and since the outputs of the model are already in the correct form (*i.e.*, category IDs can be obtained by an argmax operation over the output probabilities), a post-processing step is often unnecessary. However, an ensembling step can be conducted to obtain better results at test time. The original test image is first augmented into many images using crops centered at different locations and horizontal flips. Then, the final prediction is computed as either the average or the max over all the predictions obtained using the augmented images.

Metrics

The most natural way to evaluate models trained to solve some image classification problem is Accuracy (Acc) defined as follows:

$$\text{Acc} = \frac{\sum_{i=1}^{N} \mathbb{1}_{f(x_i)=y_i}}{N} \tag{4.1}$$

where $f(\mathbf{x}_i)$ represents the model's output for the i-th image $\mathbf{x}_i$, y_i represents the ground truth label for the i-th image, N represents the total number of images, and with the function $\mathbb{1}_{\text{cond}} \in \{0, 1\}$ evaluating to 1 iff cond is satisfied. Note Acc can also be referred to it as top 1 accuracy since we count a prediction as correct iff the class with the maximum probability corresponds to the correct class. In the context of large labeling spaces, we might also report the top k accuracy, where we count a prediction as correct if the correct class belongs to top k predicted classes ordered

(a) Input Image (b) Semantic Segmentation (c) Instance Segmentation (d) Panoptic Segmentation

Figure 4.1: Scene segmentation tasks. For a given input image (a), we show the desired output for (b) *image or semantic segmentation:* pixel-level classification, (c) *instance segmentation:* object-level segmentation and classification, and (c) *panoptic segmentation:* per-object and per-pixel segmentation and classification. In this work, we are interested in the task of image segmentation. Image source: (Kirillov et al. 2019)

by their probability in descending order. For instance, in the ImageNet challenge (Russakovsky et al. 2015), top 5 accuracy is often reported together with the top 1 measure. Additionally, for some particular cases, such as classification with imbalanced data (J. M. Johnson et al. 2019), other metrics that focus more on per-class performances, such as sensitivity-specificity or precision-recall metrics can be used. In this thesis, we generally report the top 1 accuracy when conducting image classification experiments. Note that for few-shot image classification, and since we compute the accuracy over each test set of each unseen meta-testing task separately, the reported accuracy is computed as the mean over all the accuracies obtained over all the meta-testing tasks.

4.1.2 Image Segmentation

Image segmentation, *a.k.a.*, semantic segmentation or pixel-wise classification, was always considered one of the big challenges in the long history of computer vision. It consists of automatically extracting information from images by producing dense per-pixel class assignments. It thus provides a more fine-grained understanding of the inputs and leads to a better scene understanding. The task of image segmentation jointly solves localization (*i.e.*, detecting all visible objects), classification (*i.e.*, specifying the category they belong to), and segmentation (*i.e.*, their position at the pixel level). Specifically, given an input image, an image segmentation algorithm assigns a class label to each input pixel from a set of predefined classes. The classes can be associated with specific object types in a supervised case or refer to some semantic meaning in an unsupervised setting. For general usage, a blank or a background class can be added to the label set to detect novel classes that might appear during the inference phase.

Specifically, in scene segmentation tasks, we differentiate between the following three possible variations:

- **Image segmentation.** The standard task of image segmentation consists of recognizing homogeneous regions of the image, *i.e.*, amorphous areas of similar texture or material (*e.g.*, grass, sky, road), or countable objects (*e.g.*, animals, tools, people), but without differentiating between the individual instances. As a result, if an image contains two objects of the same class, the pixels within both objects are assigned the same label.
- **Instance segmentation.** On the other hand, instance segmentation recognizes countable objects, with each instance detected separately. Each object instance is first detected independently using an object detector, and then localized within the bounding box by a follow-

ing per-instance segmentation step. In scene segmentation literature, image segmentation studies *stuff* (*i.e.*, regions of similar texture) based recognition, while instance segmentation studies *things* (*i.e.*, countable objects) based recognition.

- **Panoptic segmentation.** To reconcile between recognizing both stuff and things for a unified vision system, panoptic segmentation (Kirillov et al. 2019) leverages both image segmentation for localizing and segmenting amorphous and uncountable regions, and instance segmentation for detecting and segmenting countable objects. It thus creating a more comprehensive framework.

Fig. 4.1 for an example of the desired outputs for each scene segmentation task. In this work, we limit the scope of applications to image segmentation for simplicity and efficiency. However, the proposed methods are flexible and can easily be extended to another scene segmentation tasks and various pixel-wise prediction tasks.

Traditional image segmentation algorithms (Coleman et al. 1979; Fan et al. 2001; Shih et al. 2005), such as thresholding, clustering, and region growing, are based on handcrafted low-level features (*e.g.*, contours, edges, and blobs) to locate object boundaries in images. For instance, in the simplest case, satellite image segmentation can often be successfully performed by clustering pixels based on their wavelengths and their spatial position within the image. However, recent DL-based methods have made many of the older methods obsolete. DL-based methods are considered state-of-the-art and became the go-to techniques for image segmentation, achieving top benchmark performance across a wide range of well-known datasets and benchmarks. The pipeline to follow to solve the segmentation task using a DL-based method is detailed next.

Pipeline

To solve a segmentation task, the traditional image segmentation pipeline consists of receiving some raw pixels, applying a preprocessing step like scaling and feature extraction, which is then followed by the final training step. This training step often consists of training a shallow classifier on patches of the input image using raw pixels and the previously extracted features. However, with the introduction of end-to-end DL methods, and similar to image classification, this pipeline has become more straightforward, and consists broadly of the following steps:

- **Preprocessing.** The preprocessing step is similar to that of image classification, with data augmentation, normalization, and then the construction of a mini-batch of examples for the next training iteration. Compared to image classification, additional attention must be paid to the type of data augmentations used. For instance, applying geometric transformations (*e.g.*, rotation or translation) will result in a change of the input coordinate space, and in this case, the targets should also be transformed correspondingly to keep the correct per-pixel matching.
- **Training.** This step is the same as other DL systems, consisting of a forward pass to produce the predictions, followed by loss calculation, a backward pass to compute the gradients and an update of the model's parameters.
- **Postprocessing.** In image segmentation, an optional postprocessing step might be beneficial to refine the segmentation maps. It can consist of an inference step based on an image

pyramid of the input. The model first generates segmentation maps over different scales of the input image, that are then resized to the same original size and ensembled into a final prediction. Another possibility is the usage of a dense Conditional Random Field (CRF) (Krähenbühl et al. 2011) explicitly designed for the refinement step.

Metrics

For image segmentation, defining the correct metrics can be challenging mainly because we need to measure two values simultaneously: classification and localization. Classification measures the pixel-wise class labels, while localization measures the set of correctly classified pixels that enclose a given object. Different metrics are used to measure either or both of these values. In this thesis, we use the mean Intersection over Union (mIoU) for semi-supervised methods and pixel-wise Accuracy (Acc) for unsupervised methods.

Let $C \in \mathbb{N}$ be the number of classes we wish to recognize, $n_{ij} \in \mathbb{N}$ be the number of pixels predicted by the model to belong to class i but labeled as class j, and $t_i = \sum_{j=1}^{C} n_{ij}$ as the total number of pixels predicted as class i, both computed over all the images in the dataset. The following is a brief explanation of these two measures:

- **Pixel-wise Accuracy (Acc).** This measure, *a.k.a.*, global accuracy (Badrinarayanan et al. 2017) or per-pixel rate (Thoma 2016), is similar to the image classification accuracy, but with the correct predictions counted at the pixel level instead of the image level:

$$\text{Acc} = \frac{\sum_{i=1}^{C} n_{ii}}{\sum_{i=1}^{C} t_i} \tag{4.2}$$

  However, while the pixel-wise accuracy measures the classification performance, it does not take into account the localization aspect of the prediction. For example, if our dataset contains many images with large regions of the same class, *e.g.*, sky, the system might superficially learn to always assign the same class to this specific region of the image, resulting in overall higher accuracy, despite the model's defect.

- **Mean Intersection over Union (mIoU).** To overcome the limitations of the pixel-wise accuracy measure, mIoU can be used as a measure of both classification and localization. By leveraging per-class IoU used for comparing the similarity and diversity between the predictions and ground truths, the mIoU is computed as the class-averaged IoU:

$$\text{mIoU} = \frac{1}{C} \sum_{i=1}^{C} \frac{n_{ii}}{\sum_{j=1}^{C} (n_{ij} + n_{ji}) - n_{ii}} \tag{4.3}$$

4.1.3 NLP tasks

The field of Natural Language Processing (NLP) aims to equip computers with the ability to understand human languages. It is a collection of computational methods conceived to analyze and process languages through different NLP tasks automatically. Each task generally consists of a mapping from the input text to a different linguistic form that encodes or extracts some meaning

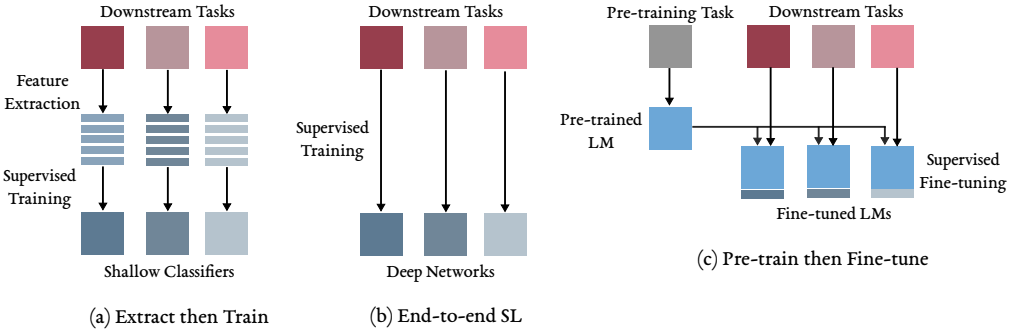


Figure 4.2: Popular NLP paradigms. We show the popular paradigms that can be followed to solve a given NLP task. (a) Extract and train consists of manual feature extraction and then per-task shallow classifier training. (b) End-to-end SL-based training with task-specific deep neural networks. (c) Pre-train then fine-tune pipeline, in which a large LM is first pre-trained on a substantial amount of text in an unsupervised manner, and then fine-tuned for each task by adding task-specific layers on top of it.

from it (N. A. Smith 2011). However, extracting meaningful information from raw text that bears some degree of resemblance to the capabilities demonstrated by humans is extremely challenging, and requires a deep understanding of natural language (Chowdhary 2020).

To solve such tasks, and mirroring to some degree traditional image recognition approaches, early NLP methods (Lafferty et al. 2001; Och et al. 2004; Y. Zhang et al. 2011) relied heavily on feature engineering that requires domain knowledge of specialists and researchers to design feature extraction methods from raw data. The extracted features are then used as inputs to task-specific models trained to solve distinct NLP tasks with the corresponding labeled training data. The next wave of DL-based NLP methods (Bahdanau et al. 2014; Y. Chen 2015; J. Chung et al. 2014; Kalchbrenner et al. 2014) consisted of designing neural architectures (*i.e.*, some variation of Feed-forward, CNN, or RNN-based networks) with the appropriate inductive biases to solve a given NLP task under the standard SL paradigm with fully labeled and task-specific datasets. However, while demonstrating a noticeable performance boost, such methods suffer from limited applicability since they are designed with a single or few NLP tasks in mind. Compared to vision applications, where most downstream tasks start from an often pre-trained image classification backbone, these different task-specific NLP architectures and their pre-trained parameters are not easily transferable from one task to the other.

In recent years, and after the introduction of the transformer architecture (Vaswani et al. 2017) which is more appropriate for NLP-like tasks, almost all state-of-the-art NLP methods switched to the pre-train then fine-tune framework. It consists of building a transformer-based model, which is first pre-trained as a large Language Model (LM) in an unsupervised manner on a massive amount of textual data that can be scraped easily from the web (*e.g.*, The Pile dataset L. Gao et al. 2020 containing 825 GiB of diverse textual data). After pre-training, the LM will have acquired robust, and general-purpose features (Clark et al. 2019; Ettinger 2020; Rogers et al. 2020) that reflect an elaborate understanding of the language necessary to solve the various downstream NLP-tasks. Then, to solve a given task, few task-specific parameters can be added to the model if necessary (*e.g.*, a linear layer for classification), followed by a supervised fine-tuning step with the appropriate loss

function to adapt the model to the data and task of interest. For an illustration of these three paradigms, see Fig. 4.2.

Since the *pre-train* then *fine-tune* paradigm has demonstrated very good performances and the resulting large pre-trained LM are also available in open-source access[1], we adopt this paradigm in this thesis. Our work with NLP tasks assumes a pre-trained model, and we focus on the improvement of the fine-tuning step in the context of limited labeled data.

Pre-training Task

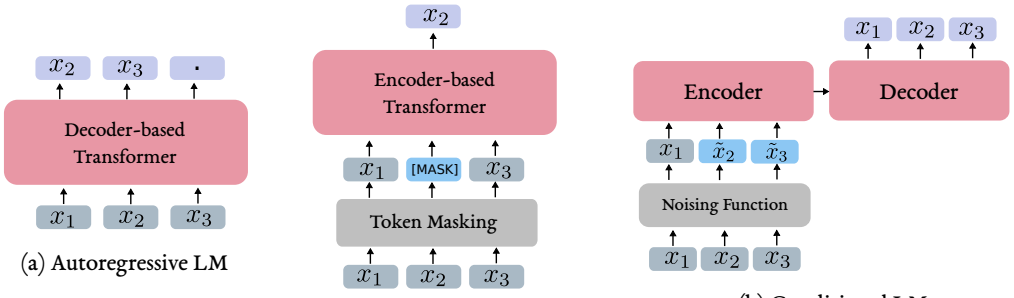
As discussed above, in the chosen paradigm, the pre-training step is the basis of all the downstream tasks. As such, the pre-training task must be carefully selected and designed to ensure that:

- The model acquires the general linguistic knowledge necessary for downstream during pre-training.
- The pre-training must be learned in an unsupervised manner to leverage the vast amount of available unlabeled raw text data.

The most popular task that satisfies these requirements is the task of language modeling (Qiu et al. 2020). Its objective is to learn a probability distribution over a given sequence of tokens (*i.e.*, the input text is represented as a sequence of elementary units we call tokens) corresponding to a language. *i.e.*, the probability assigned to a given input text $\mathbf{x}$ reflects its linguistic validity. For transformer based-models, this task is generally modeled and solved using one the following training objectives (P. Liu et al. 2021):

- **Autoregressive LM.** Such models (Fig. 4.3 (a)) directly optimize the LM objective, *i.e.*, the joint probability of the input sequence $p(\mathbf{x})$, by factorizing it as the product of the conditional probabilities of its tokens x_i using the chain rule, *i.e.*, $p(\mathbf{x}) = \prod_{i=1}^{n} p(x_i \mid x_1, \ldots, x_{i-1})$, making it a linguistically appropriate representation since human languages have a natural sequential order. Moreover, since most languages follow a left-to-right writing style, the tokens are often indexed similarly. and the autoregressive model is considered a Left-to-right LM. These models, and similar to image-based autoregressive models presented in Section 3.3.1, are trained to predict each input token based on the tokens ordered before it. Then, at test time, they can be used for both understanding and generation-based tasks. They are however principally used for generation tasks since the downstream task matches the pre-training objective of sequential token prediction. The most popular examples of such transformer-based models are GPT models (*i.e.*, GPT-1 Radford et al. 2018, GPT-2 Radford et al. 2019, and GPT-3 Brown et al. 2020).
- **Masked LM.** While autoregressive LMs are appropriate for language generation-based downstream tasks (*e.g.*, paraphrasing, sentence completion, machine translation, or text summarization), they are not optimal for other tasks that require more optimal, holistic, and bidirectional representations of the input sequence, such as classification. As such, Masked LMs (Fig. 4.3 (b)) popularized by BERT (Devlin et al. 2019), take a different approach. They define a denoising-based pre-training objective aiming to reconstruct the clean version $\mathbf{x}$

[1] For instance: https://huggingface.co/models



(a) Autoregressive LM (b) Masked LM (b) Conditional LM

Figure 4.3: Typical training objectives for transformer-based LMs. To pre-train a transformer-based LM, one of the following popular training objectives could be used. (a) Autoregressive LM consists of reconstructing the input token autoregressively, from left-to-right, where each output prediction depends only on the input tokens that came before it. (b) Masked LM consists of denoising the input, but only by predicting the clean version of the perturbed, *i.e.*, masked, input tokens. (c) Conditional LM reconstructs the whole clean input autoregressively, from left-to-right, but conditioned on the whole perturbed input sequence.

of the corrupted input sequence $\tilde{\mathbf{x}}$, *i.e.*, predicting $p(\mathbf{x}|\tilde{\mathbf{x}})$. Specifically, in Masked LMs (MLMs), a subset of the input tokens are first masked by replacing them with some special token (*e.g.*, `[MASK]`), resulting in a noisy input sequence. The model is then trained to only predict these masked tokens based on the whole input sequence (*i.e.*, bidirectional processing). After pre-training, the model can be fine-tuned to solve the task of interest over the clean inputs.

- **Conditional LM.** Although MLM can produce more optimal representations with their bidirectional processing, they cannot be directly used for generation tasks contrary to autoregressive models. Conditional LMs (Fig. 4.3 (c)) try to define a framework that solves the issues of both of these approaches while maintaining their benefits. They consist of two stages. First, an encoding step processes the input sequence bidirectionally similar to MLM. Then, a decoding step predicts the outputs autoregressively from left-to-right similar to Autoregressive LM, but this time conditioned on all of the encoded input. During pre-training, the model consumes the full corrupted input in the first step (*i.e.*, corruption methods can be either masking as in MLM, deletion of some tokens, random token replacement or permutation of a subset of tokens), and then is trained to predict, from left-to-right, the full clean input conditioned on the full corrupted input. These two steps can be either conducted by a shared model (*i.e.*, Prefix LM) as in UniLM 1-2 (Bao et al. 2020; P. Liu et al. 2021) and ERNIE-M (Ouyang et al. 2020), or by two separate models (*i.e.*, Encoder-Decoder LM) as in T5 (Raffel et al. 2020), BART (Lewis et al. 2019) and MASS (K. Song et al. 2019). Then, at the fine-tuning stage, the encoding and the decoding stages of the model can be adapted and used appropriately based on the nature of the downstream task.

Downstream Tasks

In NLP, the downstream tasks play an important dual role. They provide many realistic task formulations that mirror many language-based real-world applications, and they can be used to quantify and measure the general language learning abilities of pre-trained LM, which can help us choose and develop better NLP models and training objectives. However, given the many possible use cases of a language-based system, NLP tasks must cover a diverse set of scenarios, which has resulted in the development of many downstream tasks that explore different problems (Qiu et al. 2020). They range from standard tasks such as classification or generation to more unique ones that are particular to NLP, such as knowledge-probing tasks that try to quantify how much factual (Petroni et al. 2019) and linguistic (Ettinger 2020) knowledge is contained in the internal representation of a pre-trained LM. Other examples are reasoning tasks (B. Y. Lin et al. 2019; Wallace et al. 2019) that examine if these large pre-trained LM are capable of performing some form of complex reasoning and are not based mostly on memorizing the patterns encountered during pre-training (P. Liu et al. 2021; Niven et al. 2019).

The standard NLP downstream tasks that are commonly used to measure the effectiveness of the fine-tuning stage can be broadly categorized into the following:

- **Classification Tasks.** They consist of assigning a single or multiple categories IDs to the input sequence as a whole. They can be simple text classification tasks such as topic classification or sentiment analysis, where the output of the model directly reflects some properties of the input text that we are evaluating the model on. Furthermore, they can also consist of more elaborate tasks such as in Natural Language Inference (NLI)[2], which is used as an indirect way to probe the model's language understanding capabilities. In NLI, the model takes as input a contiguous sequence of text containing two sentences, a premise, and a hypothesis, and the model is trained to solve a 3-way multi-class classification problem predicting the relationship between these two sentences. The 3 possible relationships are entailment where the premise entails the hypothesis, contradiction where the premise contradicts the hypothesis, or neutral, where the premise is unrelated to the hypothesis.
- **Question Answering.** Another task that probes the language understanding capabilities of the model is the Question Answering (QA) task, which can also be referred to as a reading comprehension or a machine comprehension task. It predicts the answer to a given input question, often based on a context document. This task can be solved using different formulations: i) multi-choice QA that solves a multi-way classification, where the model chooses the correct answer from multiple choices provided as input. ii) Extractive QA that tasks the model with finding the span corresponding to the answer from the context document. Or iii) free-form QA that takes a more natural way of solving the task by generating the answer as an arbitrary textual string produced by the model.
- **Generation Tasks.** They consist of generating some forms of text that is coherent and human-readable. The generated text is often based on task-specific information provided as input to the model on which it will be conditioned on. Common generation based NLP tasks (Celikyilmaz et al. 2020) are text summarization (*e.g.*, summarization of single or multi

[2] Natural Language Inference (NLI) can also be referred to as Recognizing Textual Entailment (RTE) (MacCartney 2009)

documents, news, screen-plays, web-based text, and tables), machine translation, dialog response generation (*i.e.*, chatbot), paraphrasing, prompt-based text generation and data-to-text generation.

- **Extraction tasks.** They consist of finding and extracting a specific and predefined structured information from an unstructured input text. Examples of such tasks are relation extraction that predicts the semantic relationship between two entities in the input sentence, and Named Entity Recognition (NER) that consists of identifying entities and assigning each one its corresponding type from a set of predefined categories (*e.g.*, person, location, name).

In this work, we will limit ourselves mostly to classification-based tasks.

Pipeline

In NLP, the DL-based pipeline to solve a given task of interest is more involved and consists of preprocessing and postprocessing steps tailored for text-based inputs which are are not required in the previously discussed vision task. In NLP, the raw text inputs cannot be directly processed by our model of choice, so they must be converted into the appropriate representation at the preprocessing stage before the training step. Then, depending on the NLP task we are tackling, the outputs must also be converted into the desired format, such as a sequence of generated text. In the following, we will detail the essential steps conducted during a standard NLP DL-based training pipeline.

- **Preprocessing.** The objective of this step is to convert a batch of input sequences from raw text to a set of same-length sequences of unique integer IDs. Each ID represents the index of a given input token in a vocabulary of unique tokens found in the training corpus. As such, the first step that must be conducted once before the start of training is the construction of the tokenizer. To build it, all our training data, or corpus, is processed by a sub-word segmentation algorithm (*e.g.*, Byte Pair Encoding (BPE) (Y. Wu et al. 2016) or WordPiece (Sennrich et al. 2015)) to segment the input text into its elementary units, called tokens (*i.e.*, words, sub-words, and symbols). Then, the vocabulary can be constructed by only considering the unique tokens in our vocabulary as its entries [3].
  Now that the vocabulary is built, we can start the training process. At each iteration, we fetch the input sequences, tokenize them using a chosen sub-word segmentation algorithm, and then replace each token with its index in the vocabulary (*i.e.*, token IDs). Finally, all the input sequences are either padded to build a mini-batch of same length input sequences that can be fed into the model.
- **Training.** At a given training iteration, the batch of input sequences must first be converted into a sequence of learnable embeddings, where each token has its unique learnable representation. This can be done by simply passing the token IDs through an embedding layer, and the produced outputs are the learnable feature vectors (*i.e.*, word embedding)

[3] Note that the vocabulary construction step is done only during the pre-training stage. When we fine-tune on a given downstream task, the same vocabulary used for pre-training is also used down-stream, *i.e.*, reusing the same tokenizer.

corresponding to each input token ID. Then, we fall back to the standard DL-based training step, where this batch of embeddings is passed through the transformer-based model, resulting in a set of predictions. The latter are then used to compute the loss function, apply backpropagation, and update the model's weights and the learnable embeddings.

- **Postprocessing.** This step is often dependent on the target NLP task. For instance, for understanding tasks such as text classification, the process is straightforward and consists mostly of transforming the probability distribution over the classes into a category ID or a category name. However, this step can be critical for generation tasks since it directly affects the quality of the generated text. In such tasks, at each decoding step (*i.e.*, left-to-right generation, one token at a time), the probability distribution over the vocabulary must be converted into a token ID, and then the sequence of tokens into raw text. A simple approach consists of choosing in a greedy manner the token with the maximum probability at each step. Nevertheless, it often results in sub-optimal sequences. Indeed, while each token is chosen as the most likely one, the sequence as a whole might note be. An alternative to this simple approach is Beam search (Freitag et al. 2017) that keeps track of the most likely output sequences at each decoding step, and then chooses the one with the highest combined probability as the final prediction. Although it is memory intensive and computationally expensive, the quality, diversity, and coherence of the generated text are notably higher.

Metrics

The evaluation metric of a given NLP system depends on the downstream task we are tackling. For instance, for a multi-way classification task, Accuracy (Acc) is often chosen as the measure of choice. For binary classification, in addition to accuracy, the F_1 score is usually also reported to better estimate the overall quality of the model. This measure is defined as the harmonic mean of Precision (P) and Recall (R) and is computed as follows:

$$\mathrm{F}_1 = 2\frac{\mathrm{P}\cdot\mathrm{R}}{\mathrm{P}+\mathrm{R}}, \quad \text{with } \mathrm{P} = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FP}} \text{ and } \mathrm{R} = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}} \tag{4.4}$$

where TP, TN, FP, and FN are the number of True Positives, True Negatives, False Positives, and False Negatives, respectively. While the accuracy and F_1 score remain the most popular measures for binary classification, they can, however, give overly optimistic results if the evaluation dataset is imbalanced. In such a scenario, it is often preferred to report Matthews Correlation Coefficient (MCC) (Chicco et al. 2020) instead, since it gives good results only if they are also reflected in all the four measures (*i.e.*, TP, TN, FP, and FN).

For generation tasks, the evaluation becomes harder since a correct response can be formulated or expressed in multiple equally correct ways. For such tasks, the evaluation metric is often chosen as the classical BLEU score metric which has a transparent decision process, but it is of low quality. Another choice is BERTScore (T. Zhang et al. 2020) metric which is based on a pre-trained LM and offers a higher quality measure with a strong resemblance to the way humans judge the quality of the generated text, but remains as a black-box method, making it hard to explain, understand and improve upon. The exact definitions of these metrics are omitted since generation-type NLP tasks are not the focus of this thesis.

Key Takeaways

- For visual tasks, we focus on two tasks, image classification, and image segmentation. For image classification, the objective is to predict the correct class assignments for a given input image. The model's performance is often measured using the accuracy metric. For image segmentation, the objective is to predict the correct class assignments for each pixel in the input image. The model's performance is often measured using pixel-wise accuracy or mIoU.
- For textual tasks, we focus on various NLP downstream tasks that solve for a mapping from the provided input text to outputs in the form of linguistic structures that encode some form meaning and knowledge contained in it. To solve such downstream tasks, recent NLP methods often follow the pre-trained then fine-tune framework, in which a large model is first pre-trained on a language modeling task in an unsupervised manner and then fine-tuned on the downstream task of interest using a task-specific labeled set.

Usage

This thesis tackles one or two of these tasks for each learning paradigm we consider. Regarding the modalities, for the visual modality, image classification is tackled in Chapters 7 and 8, and image segmentation in Chapters 5 to 7. For the textual modality, Chapter 9 tackles various NLP downstream tasks. Specifically, the downstream NLP tasks we consider are mostly classification tasks, and depending on the task, we use one of the above metrics for evaluation (*i.e.*, either Acc, F_1, or MCC)[4].

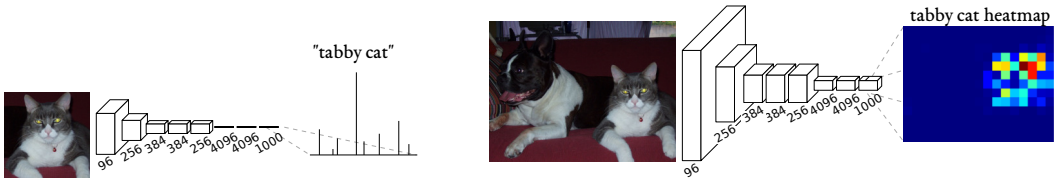
4.2 Models

This section presents the neural architectures and the set of design choices or inductive biases used to build our inference function f. Since the architecture is both modality-dependent and task-specific, we introduce the popular choices followed in the main applications we are interested in. First, we start with a brief overview of image classification networks which have become common for such tasks. Then, we present image segmentation networks that require more specialized designs and methods. Finally, we introduce the transformer architecture, which has become the backbone of most NLP tasks.

4.2.1 Image Classification

In this thesis, all of our neural architectures used to solve image classification tasks are CNN-based, and more specifically, ResNet (K. He et al. 2016) based. CNNs (Rawat et al. 2017) consist of a stack of modules or blocks, with each block containing a sequence of convolutional, normalization,

[4] Note that in Chapter 9, and in order to follow previous works, one of the downstream tasks if of regression type, and in this case, we will exceptionally report the Spearman Correlation as our regression measure.



(a) Classification Network (b) Segmentation Network (FCN)

Figure 4.4: Replacing the fully connected layers. The fully connected layers of a classification network (a) are turned into convolution layers (b), enabling the classification network to output a heatmap containing sparse information about the object's locations. With additional upsampling layers, the outputs can be turned into segmentation maps for an end-to-end image segmentation system. Image source: (J. Long et al. 2015).

non-linearities, and pooling layers. Within each block, the convolutional layer (Conv) serves as a features extractor with its learnable kernels. Its inputs are convolved by its kernel weights, resulting in updated feature maps in which each output neuron is only connected to a local region of the input volume by way of the layer's kernel. A normalization operation follows this, often conducted using Batch Normalization (BN) (Ioffe et al. 2015) that adjusts the mean and variance of the features to make the training more stable and less sensitive to changes in the parameter initialization and the learning rate. These normalized feature maps are then passed through a non-linearity layer (*e.g.*, ReLU (Nair et al. 2010) or GELU (Hendrycks et al. 2016)) to extract non-linear features. Finally, a pooling layer, usually maximum based, reduces the spatial dimension of the features maps, achieving spatial invariance and decreasing the computational requirements to be able to increase the depth of the feature maps. After a stack of such blocks, the final feature maps are usually flattened into a feature vector, and passed through a final fully connected layer to produce the classification scores.

ResNet (K. He et al. 2016) follows the same overall design, where each block consists of a series of $\{3 \times 3$ Conv - BN - ReLU$\}$, but with an additional residual connection that adds the input of the block to its output before passing the final feature maps to the next residual block. This residual design simplifies the learning procedure by requiring each block to only learn adjustments to its input rather than full input transformations. It also makes building very deep networks (*e.g.*, 152 ResNet model with 152 layers) feasible by allowing the errors to backpropagate directly to earlier layers, making the optimization procedure more stable.

Throughout this thesis, we use different ResNet-based variations, and which will usually be refereed to as R-N with N as the number of layers (*e.g.*, R-18, R-34, R-50), for both image classification tasks as our classification model, and for image segmentation tasks as our base model upon which the segmentation model is built.

4.2.2 Image Segmentation

Prior to the introduction of specialized neural architectures designed specifically for tasks that required dense predictions, *e.g.*, pixel-wise classification, the first iterations of DL-based segmentation networks (Ciresan et al. 2012; Ganin et al. 2014) consisted of converting the classification architectures, such as AlexNet (Krizhevsky et al. 2012) and VGG (Simonyan et al. 2014), to segmentation networks by swapping and fine-tuning the last fully connected layers for image segmenta-

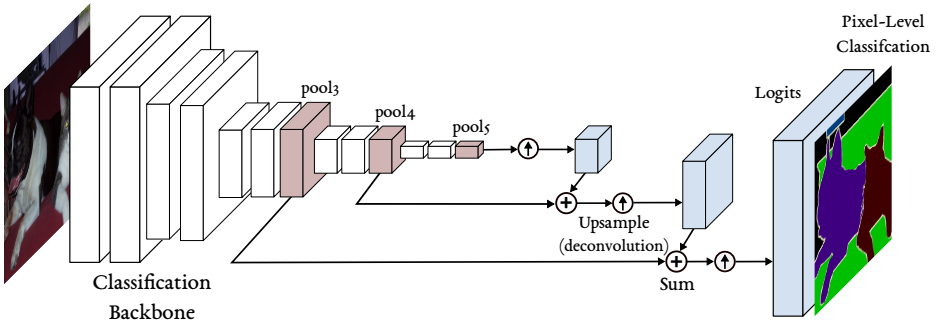


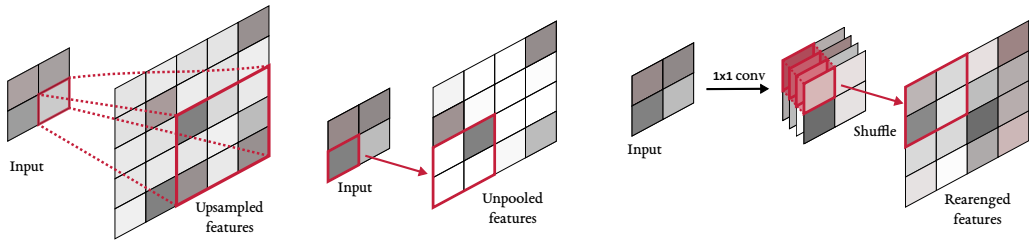
Figure 4.5: A Fully Convolutional Network (FCN). FCN consists of sequential upsampling intertwined with skip connections to learn to combine coarse and semantic information with fine and local details for more detailed and accurate segmentations. In this figure, we illustrate the FCN-8s version, where the output features are combined starting from pool3 of a VGG classification backbone.

tion. However, these attempts suffered from overfitting and were not sufficiently deep to create abstract and rich features for a coherent segmentation. Subsequent works proposed alternative solutions to fully connected layers such as recurrent architectures (Pinheiro et al. 2014) and hierarchical features (Farabet et al. 2012). Since these first attempts were based on the standard deep neural architectures that are capable of extracting abstract and rich local features, but cannot utilize the global context, the obtained segmentation maps were often of low quality, lacked global coherence, and required a heavy post-processing step to refine them.

To improve these architectures, CNN architecture tailored for image segmentation were proposed following their success in image classification. Fully Convolutional Network (FCN) (J. Long et al. 2015) proposed such an architecture by converting classification architectures into fully convolutional networks as illustrated in Fig. 4.4. Therefore, the resulting FCN is a classification network capable of processing images of arbitrary sizes and producing spatial feature maps with the correct corresponding sizes instead of classification scores. Then, these feature maps can be converted into pixel-wise classification scores using a classifier applied at each spatial location. Additionally, to capture both global and local context for joint localization and classification and produce more precise segmentations, an upsampling module can be added on top before the classification layer, upscaling and merging the feature maps from the final layers of the model with feature maps of earlier layers (see Fig. 4.5), resulting in more globally coherent results.

The introduction of FCNs has spurred a wave of novel segmentation approaches that proposed various techniques to better address the need for fine-grained localization of class labels at the pixel level while maintaining a global coherence at the object level. We describe in the following the set of popular methods and techniques that have demonstrated their effectiveness when they are integrated into a given segmentation architecture:

- **Encoder-Decoder Architecture.** Such models are one of the most popular architectures for image segmentation. As the name suggests, the model consists of two components: (a) an encoder where the spatial dimension of feature maps is gradually reduced, capturing long-range interactions and more global and semantic information, and (b) a decoder where object details and spatial dimensions are gradually recovered. The encoder is usu-

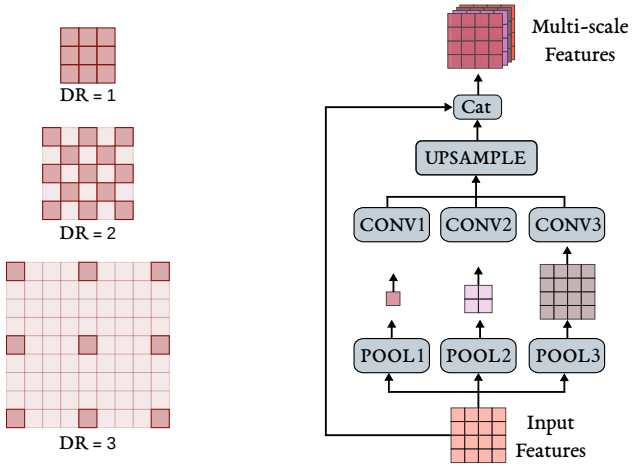


(a) Deconvolution (b) Unpooling (c) Pixel Shuffle

Figure 4.6: Upsampling Layers. Popular choices of upsampling layers in the decoder part of an encoder-decoder segmentation network. (a) Deconvolution, which consists of applying a transposed convolution. Such an operation is similar to the standard upsampling operation but with learned weights. (b) Unpooling, which applies the inverse operation of a pooling layer and consists of copying the input values to the same output positions used during the corresponding pooling layer in the encoder. (c) Pixel shuffle, which first applies a 1×1 Conv to adjust the depth with the desired upsampling rate, then the outputs at the same spatial location are rearranged spatially to get the correct upsampled output.

ally a standard pre-trained network on a classification task (*e.g.*, VGG, AlexNet, ResNet) to be fine-tuned for segmentation. The decoder, on the other hand, consists mostly of upsampling layers. Instances of such layers are shown in see Fig. 4.6. Specifically, they can be based on deconvolutions (Zeiler et al. 2011) (*i.e.*, transposed convolutions that can be seen as an upsampling operation with learned weights), unpooling (Badrinarayanan et al. 2017) (*i.e.*, the inverse of a pooling operation), or pixel shuffling (Shi et al. 2016) (*i.e.*, 1×1 Conv followed by a rearrangement of the features). For example, FCN (J. Long et al. 2015) employs deconvolutions to learn the upsampling of low-resolution features. SegNet (Badrinarayanan et al. 2017) stores the pooling indices when a pooling layer is used in the encoder and employs them in the decoder for upsampling together with convolutional layers to increase the density of the upsampled feature maps. U-Net (Ronneberger et al. 2015) is a symmetric encoder-decoder network with skip connections from encoder features to the corresponding decoder activations. After defining the decoder's architecture, it is then trained from scratch on top of the pre-trained backbone network (*i.e.*, the encoder). In this thesis, we often build our segmentation models following such an encoder-decoder structure.

- **Atrous convolution.** The long-range information captured with CNNs is often limited by the output's receptive field, which depends on the network depth and the convolution kernel size. As a result, the long-range information that can be captured is upper bounded by the available GPU memory and overall computational resources. To circumvent this, standard convolutions of the encoder can be replaced with atrous convolutions (L.-C. Chen et al. 2017b; Yu et al. 2015), *i.e.*, dilated convolutions. Specifically, atrous convolutions are characterized by a dilation rate, where a rate of r consists of uniformly enlarging the kernel of the convolution from size $k \times k$ to $k + 2(r-1) \times k + 2(r-1)$. This is done by zero padding with $r-1$ in between each pair of kernel weights. For an example, refer to Fig. 4.7 (a). With large dilation rates, we enlarge the model's field of view, enabling object encoding at multiple scales while also maintaining an acceptable efficiency. In this thesis, we often



(a) Atrous Convolutions (b) Spatial Pyramid Pooling Module

Figure 4.7: Atrous Convolutions and Spatial Pyramid Pooling. (a) shows examples of atrous convolutions (*a.k.a.*, dilated convolution) for a 3×3 kernel for three Dilation Rates (DR). In each example, we pad the original kernel with DR-1 zero weights between each pair. Note that for DR$= 1$, we fall back to the standard convolution. (b) shows an example of a Spatial Pyramid Pooling module used to generate multi-scale features in PSPNet (H. Zhao et al. 2017). The input features are first pooled at various rates to capture different scales, passed through a Conv layer, then upsampled to return to the original spatial input size. The final features are then obtained by concatenating all the upscaled and input features.

deploy dilated convolutions, but only in the last few blocks of our backbone, where the strided 3×3 Conv layers are replaced by their dilated counterparts, but without any stride (*i.e.*, stride $s = 1$).

- **Spatial pyramid pooling.** This method (Grauman et al. 2005; Lazebnik et al. 2006) presents an efficient alternative to the standard image pyramid (L.-C. Chen et al. 2016) approach (*i.e.*, applying the same model to multiple scales of the input image) by conducting the multi-scale processing at the feature level instead of the input level. The multi-scale processing consists of applying pooling operations with different kernel sizes or atrous convolutions with various dilation rates. It thus capturing contexts at several ranges and different scales. For instance, DeepLabv2 (L.-C. Chen et al. 2017a) uses this scheme with atrous spatial pyramid pooling, where parallel atrous convolutions with different dilation rates are used to capture multi-scale information. PSPNet (H. Zhao et al. 2017) performs spatial pooling at several grid scales and demonstrates outstanding performance on several image segmentation benchmarks (see Fig. 4.7 (b)). In this thesis, we follow PSPNet and use their proposed module for pyramid pooling.

In the upcoming contributions chapters, and depending on the use case and the learning paradigm, we employ various combinations and different variations of these techniques. The segmentation architecture and the techniques employed are detailed in the experimental section of the relevant chapters.

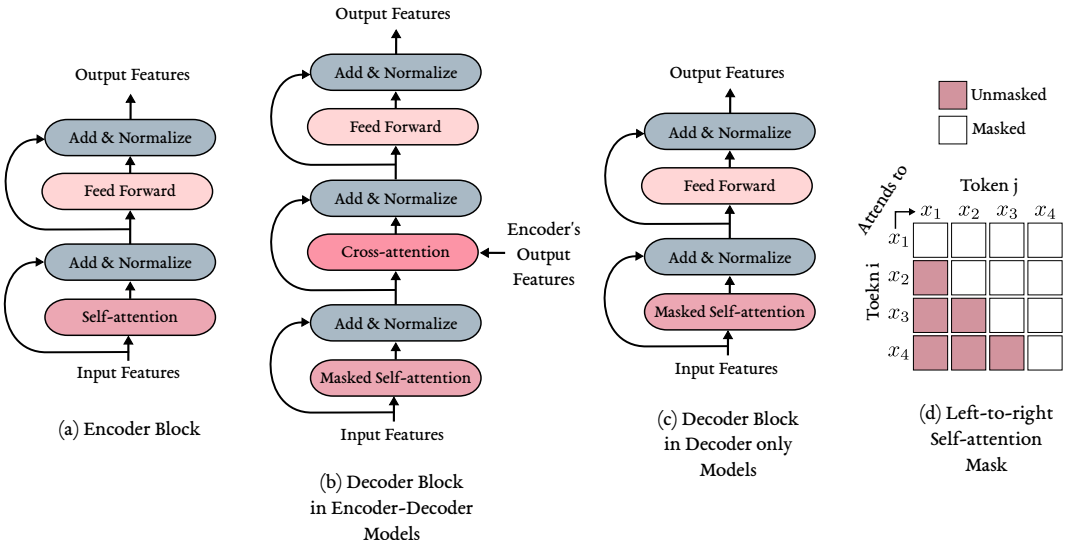


Figure 4.8: The Building blocks of the transformer architecture. We show the standard blocks used to build transformer-based models, in addition to the attention mask applied in a masked attention operation. (a) The encoder block used in either encoder-only models for bidirectional processing or decoder-encoder models for a conditional generation. (b) The decoder block used in decoder-encoder models in which the decoder features are updated based on the encoder's features. (c) The decoder block used in decoder-only models, which are often autoregressive LMs. (d) An example of the attention mask for an input sequence of 4 tokens to be applied to the attention weights during a masked self-attention operation in left-to-right autoregressive models. It forces each token x_i at position i to only attend to tokens to the left of it, *i.e.*, $x_{\leq i-1}$.

4.2.3 NLP

The transformer architecture (Vaswani et al. 2017), which was originally introduced for machine translation, has become the standard architecture in NLP under the pre-train then fine-tune framework (Section 4.1.3). It is a sequence-to-sequence model consisting of two main components, an encoder, and a decoder, with a similar number of blocks, and each block is composed of two or three modules. For the encoder, each block (Fig. 4.8 (a)) contains an attention module in which a self-attention operation is applied followed by layer normalization (LN) (Xu et al. 2019) and a residual connection, and a feed-forward module with a fully-connected layer, LN, and a residual connection as well. For the decoder, each block (Fig. 4.8 (b)) has an additional attention module with a cross-attention operation inserted after the first self-attention-based. With such a design, the model architecture is quite flexible and general-purpose. In each block, we gather global information from all the contexts conditioned on the local value using the attention operation, followed by an FC layer to merge the updated local information. This provides the model with the ability to learn structures and patterns that are data and task appropriate with a minimal amount of explicitly specified inductive biases (*e.g.*, self-attention can learn to perform operations similar to convolutions Cordonnier et al. 2020b).

In the transformer model, the self-attention operation used is a modified version of the original operation (Bahdanau et al. 2014). Specifically, Let $\mathbf{z} \in \mathbb{R}^{L \times d}$ be a set of d-dimensional features corresponding to an input sequence of length L. The self-attention operation starts by projecting the features into query, key and value vectors $\mathbf{q}_i, \mathbf{k}_i, \mathbf{v}_i \in \mathbb{R}^d$ at each input location $i \in [1, L]$ using linear projections $\mathbf{W}_q, \mathbf{W}_k, \mathbf{W}_v \in \mathbb{R}^{d \times d}$. The self-attention then consists of comparing the query $\mathbf{q}_i$ to every key across the whole input sequence to get per-location weights, which are then used as a weighted sum of all the values to obtain the updated features $\mathbf{z}_i$ at location i:

$$\mathbf{z}_i = \sum_j^N \mathbf{v}_j \frac{\exp(\mathbf{q}_i \mathbf{k}_j^\top)}{\sum_k^N \exp(\mathbf{q}_i \mathbf{k}_k^\top)} \tag{4.5}$$

Note that the equation above only presents the single-head formulation and omits the dot product scaling by $\frac{1}{\sqrt{d}}$. For self-attention with H heads, *i.e.*, multi-head self-attention, the input features are first divided into H features of dimension $\frac{d}{H}$, and the operation in Eq. (4.5) is applied to every one of them separately. The results are concatenated and then passed through a linear output layer to get the final output of the attention layer. For the cross-attention operation in the decoder, the procedure remains the same, but the set of queries are generated from the decoder's features, while the keys and values are generated from the features of the encoder (*i.e.*, the features produced by the last layer of the encoder). Additionally, the self-attention in the decoder consists of an additional masking operation (Fig. 4.8 (d)) so that a given prediction at position i depends only on the inputs at positions j to the left of it, *i.e.*, $j \leq i - 1$, resulting in a left-to-right type processing of the inputs.

Based on this original encoder-decoder architecture, transformer-based models can take different forms based on the NLP task we wish to solve. They can follow the original architecture for conditional sequence-to-sequence generation tasks (*e.g.*, T5 (Raffel et al. 2020), BART (Lewis et al. 2019)). They can be based on the encoder part only for bidirectional processing of the input sequence. Such models (*e.g.*, BERT (Devlin et al. 2019), RoBERTa (Y. Liu et al. 2019)) are mostly used for classification and sequence labeling tasks. They can also be based solely on the decoder, where in this case, the cross-attention module is removed (Fig. 4.8 (c)), resulting in a model convenient for language modeling (*e.g.*, GPT models (Brown et al. 2020; Radford et al. 2018, 2019)).

In this thesis, and as detailed in Section 4.1.3, we follow the popular pre-train then fine-tune the framework in NLP, starting from an already designed and pre-trained transformer-based LM (*i.e.*, RoBERTa), and set to develop better fine-tuning methods under the constraint of limited labeled data.

Key Takeaways

- For image classification, we focus on CNN-based and, more specifically, ResNet-based models. ResNet models are built as a series of residual blocks. Each block is built using convolutional blocks of {3 × 3 Conv - BN - ReLU} and residual connections for stable training and faster convergence.

- For image segmentation, we focus on encoder-decoder-based models. The encoder is often initialized using a pre-trained image classification model, with additional adjustments such as using atrous convolutions to obtain larger receptive fields or adding a spatial pyramid pooling module for multi-scale processing. For the decoder, it consists of either a simple bilinear upsampling and a linear classification layer, or elaborate designs with learnable upsampling, Conv layers, skip connections, and a final classification layer.
- For NLP tasks, we focus on transformer-based models, which are built using attention and feed-forward blocks, with residual connections and LN layers interleaved between them. We differentiate between three types of such models, encoder only for bidirectional processing, encoder-decoder for conditional generation, and decoder-only for language modeling.

Usage

In this thesis, we use the previously mentioned models as our starting point based on the task we tackle in each chapter. Then, we apply minor adjustments depending on the paradigms we consider and the method we propose.

4.3 Data

In DL, most methods are trained in an end-to-end manner, in which the useful features are determined by and learned from the data itself. Therefore, the training data becomes an intrinsic and important component of the system. Additionally, when testing the performances of novel approaches, the data must be selected carefully to mimic real-world settings so that the obtained performances can eventually be replicated during the system's deployment.

Broadly, popular datasets used in DL can be categorized based on either i) their use cases or ii) their data format and type. In terms of the use case, for visual tasks, the data can be generic for standard tasks, or more specialized such as images of urban settings, satellite imagery, or medical imaging. Similarity for language, we have either a general-purpose corpus or a more specialized text such as specific writings of a given author, medical transcripts, or legal documents. For the data type, for visual data, the inputs can be 2D RGB or gray-scale images, RGB-D images with depth information, 2D stereo images to replicate the human optical system, or 3D volumetric data such as X-ray scans. For language, the inputs can correspond to human languages, either single or multi-lingual, but can also be used to describe other types of information in language form such as tabular data, mathematical formulations, chemical symbols, or sequences of amino acids.

In this thesis, and to demonstrate the generality of the proposed algorithms, we apply them to various datasets with different use cases and data types based on the task, modality, and learning paradigm we consider in each given chapter. The details of the data used are presented in the experimental sections of each chapter.

As a precursor, we present the procedure we will follow to set up the training and testing data for each one of the learning paradigms we are interested in:

- **SSL** (Chapter 5). To imitate the SSL setting, we start from a fully labeled dataset often used in the SL setup, and only consider a portion (*e.g.*, 1% to 10%) of its training set as labeled while the rest is considered as unlabeled samples (*i.e.*, by disregarding their labels). For testing, the methods can be evaluated on the original test set of the dataset.
- **UL** (Chapter 6). In a fully unsupervised scenario, and since in this thesis we are interested in predicting the class assignments directly as opposed to representation learning, we train our model on simplified versions of the original datasets to create easier and more appropriate learning objectives compared to the supervised case. For example, for the task of image segmentation, we start from the challenging COCO-Stuff (Caesar et al. 2018) dataset, and simplify it by only considering a 15 coarse labels variant instead of the original 80 fine labels variant, and further reduced it to 52k examples from 164k by taking images with at least 75% stuff pixels. The number of classes can also be merged and reduced to simplify the learning objective further. To evaluate the methods using a labeled test set, first, we find the best one-to-one matching between the outputs and ground-truth classes, given that the model's class IDs do not correspond to the ground-truth ones. Then, with the optimal matching one-to-one already found, we can compute the chosen evaluation metrics as in the SL case and evaluate our method.
- **UDA** (Chapter 7). As the example in Fig. 3.4 shows, in a given UDA benchmark, the distributional shift between the two sets (*i.e.*, the labeled source data and the unlabeled target data) is often simulated by considering two training sets with compatible class labels but with different characteristics. For instance, the source can contain synthetic labeled images generated from computer games in which the annotation can be easily generated, while the target is composed of real images in which the model will eventually be deployed. We train on the labeled source and the unlabeled target and then report and obtained the result on the target set itself.
- **FSL** (Chapter 8). For the task of few-shot classification, datasets are organized based on their classes. They are divided into seen classes to be used in the meta-training set and unseen classes to be used for meta-testing (refer to Fig. 3.6 for an illustration). Then, the training and testing sets of each task can be constructed depending on the classification problem we choose to solve (*i.e.*, a C-way K-shot classification problem). For a given task, its support (*i.e.*, train) set contains $K \times C$ examples, with K examples per each one of the C classes, while its query (*i.e.*, test) is often constructed using a larger number of examples per each of the C classes to get more representative evaluation results in very low K-shot settings (*e.g.*, in our case, we will use 15 examples per class, resulting in $15 \times C$ test examples for each task). We train our model on the meta-training tasks and report the average of the accuracies obtained over each test set of the meta-testing tasks.
- **FSFT** (Chapter 9). In this setting, we follow the standard TL-based training and testing procedures in NLP. We start from a pre-trained model (*e.g.*, RoBERTa) and then set fine-tune it on the labeled training set of each one of the downstream tasks of a given NLP benchmark (*e.g.*, GLUE benchmark (A. Wang et al. 2018)). This is followed by an evaluation

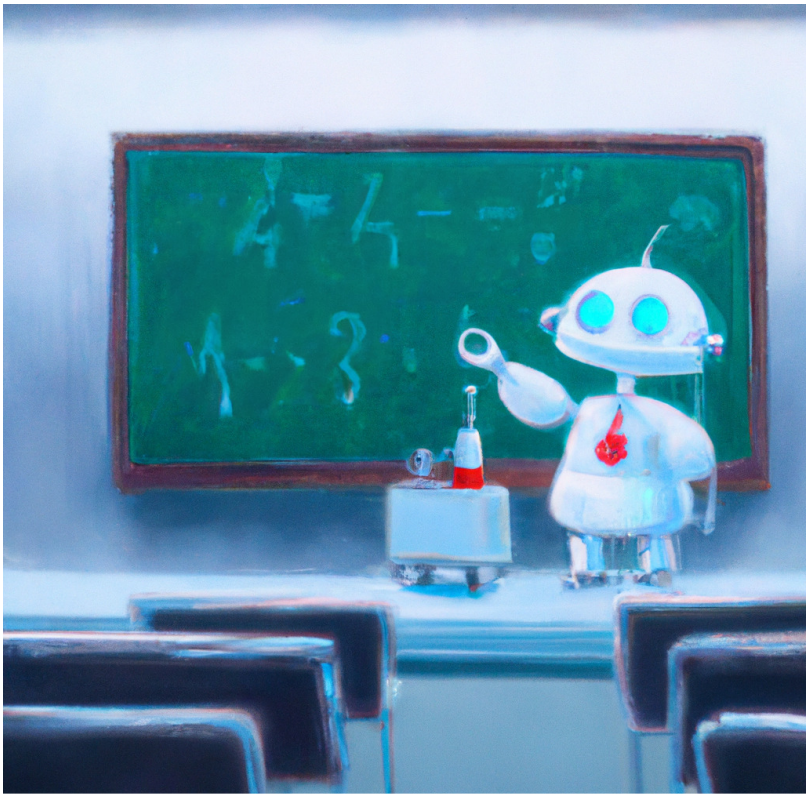
step on each one of the provided test sets. However, in our case, we constrain ourselves to cases where only a limited amount of labeled training data is available, which can be simulated by simply considering only a small portion of the original labeled training data as our training set.

Conclusion

This chapter concludes Part I of this thesis. In it, we presented the reader with the background knowledge required for the second part of this work. Specifically, in Chapter 2, we introduce the main ideas behind DL and its main building blocks. Then, in Chapter 3, we introduced the learning paradigms tackled in the contribution chapters. In this chapter, we established the tasks we consider, starting by their definitions in Section 4.1, the models used to solve them in Section 4.2 and finally, the data we use to train and evaluate the proposed methods in Section 4.3.

Next, in Part II of this thesis, Chapters 5 to 9 present the main contribution of this work.

PART TWO
Contributions



Part II

Contributions

After introducing the necessary background in Part I of this thesis, this second part presents the main contributions of this work. In each of one upcoming chapters, we tackle one of the presented learning paradigms that consider some variant of the label-efficient learning setup, and develop a novel method to solve the tasks previously described.

Specifically, the next five chapters consider the following paradigms and tasks:

- In Chapter 5, we tackle the semi-supervised learning paradigm and the image segmentation task.
- In Chapter 6, we tackle the unsupervised learning paradigm and the image segmentation task.
- In Chapter 7, we tackle the unsupervised domain adaptation paradigm and the tasks of image classification and segmentation.
- In Chapter 8, we tackle the few-shot learning paradigm and the image classification task.
- In Chapter 9, we tackle the few-sample fine-tuning paradigm and classification-based natural language processing tasks.

Each one of these subsequent chapters presented with the following structure:
Introduction → Related Work → Preliminaries → Method → Experimental Results → Conclusion.

CHAPTER 5

Cross-Consistency Training

5 Cross-Consistency Training

Chapter's Background

In this chapter, we consider:

- The Semi-Supervised Learning (SSL) paradigm (Section 3.2).
- The task of image segmentation (Sections 4.1.2 and 4.2.2).

We chose the SSL setting since it is a well-established setting in ML, and at the time of this contribution, it was gaining popularity in the DL research community. Additionally, it deals with the problem of label-efficient learning and is a flexible and practical framework. We choose the task of image segmentation motivated by two points. First, when we set to tackle the SSL setting, there were already a few established DL-based semi-supervised methods for image classification and object detection, while the SSL segmentation methods were mostly GAN-based and did not leverage the traditional SSL approaches, thus making it worth studying. Second, at the beginning of the thesis, the MICS aboratory was involved in many projects that focused on the image and document segmentation task with a limited amount of labeled data, and while not motivated by a specific application, it naturally oriented us to the task of image segmentation.

Chapter's Summary

In this chapter, we start with a preliminary analysis of cluster assumption at the pixel level for the image segmentation task. We show that the low-density regions are more apparent within the hidden representations than within the inputs for image segmentation. We thus propose Cross-Consistency Training (CCT), where an invariance of the predictions is enforced over different perturbations applied to the outputs of the encoder (*i.e.*, representation level). Concretely, a shown in Fig. 5.1, a shared encoder and a main decoder are trained in a supervised manner using the available labeled examples. To leverage the unlabeled examples, we enforce consistency between the main decoder predictions and those of the auxiliary decoders, taking as inputs different perturbed versions of the encoder's output, and consequently, improving the encoder's representations. The proposed method is simple and can easily be extended to use additional training signal, such as image-level labels or pixel-level labels across different domains. We perform an ablation study to tease apart the effectiveness of each component, and conduct extensive experiments to demonstrate that our method achieves state-of-the-art results in several datasets.

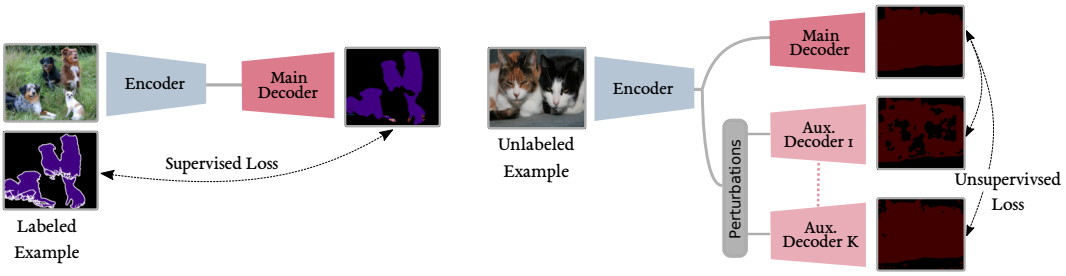


Figure 5.1: Cross-Consistency training (CCT). The encoder and the main decoder are trained in a supervised manner for the labeled examples. For the unlabeled examples, a consistency between the main decoder's predictions and those of the auxiliary decoders is enforced over different types of perturbations, which are applied to the inputs of the auxiliary decoders. Note that in practice, both steps are conducted in a single forward-backward pass using a mini-batch containing both labeled and unlabeled examples.

5.1 Introduction

In recent years, growing attention has been drawn to deep SSL to take advantage of availability of unlabeled data and limit the need for labeled examples. However, at the time of this contribution, the progress in SSL was confined to classification tasks. And its application in image segmentation is still limited. Dominant approaches (Z. Huang et al. 2018; J. Lee et al. 2019; Y. Wei et al. 2017, 2018) focus on weakly-supervised learning, which principle is to generate pseudo-pixel-level labels by leveraging the weak labels that can then be used, together with the limited strongly labeled examples, to train a segmentation network in a supervised manner. Generative Adversarial Networks (GAN) were also adapted for SSL setting (Hung et al. 2018; Souly et al. 2017) by extending the generic GAN framework to pixel-level predictions. The discriminator is then jointly trained with an adversarial loss on all examples and a supervised loss over the labeled examples. Nevertheless, these approaches suffer from some limitations. Weakly-supervised methods require weakly labeled examples and pixel-level labels. Hence, they do not exploit the unlabeled data to extract additional training signal. Methods based on adversarial training on the other hand exploit the unlabeled data, but they can be harder to train.

To address these limitations, we propose a simple consistency-based SSL method for image segmentation. The objective of consistency training is to enforce an invariance of the model's predictions over small perturbations applied to the inputs. As a result, the learned model will be robust to such small changes. The effectiveness of consistency-based training depends heavily on the behavior of the data distribution, *i.e.*, the cluster assumption, where low-density regions must separate the classes. In image segmentation, we do not observe the presence of such low-density regions separating the classes in input space, but rather within the encoder's outputs. Based on this observation, we propose enforcing the consistency over different perturbations applied to the encoder's output. Specifically, we consider a shared encoder and a main decoder which are trained using the labeled examples. To leverage unlabeled data, we then consider multiple auxiliary decoders whose inputs are perturbed versions of the output of the shared encoder. The consistency is then imposed between the main decoder's predictions and that of the auxiliary decoders (see Fig. 5.1). This way, the shared encoder's representations are enhanced by using the

additional training signal extracted from the unlabeled data. The added auxiliary decoders have a negligible amount of parameters compared to the encoder. Additionally, only the main decoder is used during inference, thus reducing the computation overhead both in training and inference.

Chapter's Contributions

To summarize, this chapter's contributions are:

- We propose a Cross-Consistency Training (CCT) method for semi-supervised image segmentation, where the invariance of the predictions is enforced over different perturbations injected into the encoder's output.
- We propose and conduct an exhaustive study of various types of perturbations.
- We extend our approach to use weakly-labeled data and exploit pixel-level labels across different domains to jointly train the segmentation network.
- We demonstrate the effectiveness of our approach with an extensive and detailed experimental results, including a comparison with the state-of-the-art, as well as an in-depth analysis of our approach with a detailed ablation study.

5.2 Related Work

Semi-Supervised Learning. Recently, many efforts have been made to adapt classic SSL methods to deep learning, such as pseudo labeling (D.-H. Lee 2013), entropy minimization (Grandvalet et al. 2005) and graph based methods (Kipf et al. 2016; B. Liu et al. 2019). This chapter focuses mainly on consistency training methods introduced in Section 3.2.3. Consistency training methods are based on the assumption that if a realistic form of perturbation is applied to the unlabeled examples, the predictions should not change significantly. It thus favors models with decision boundaries that reside in low-density regions, giving consistent predictions for similar inputs. For example, Π-Model (Laine et al. 2016) enforces a consistency over two perturbed versions of the inputs under different data augmentations and dropout. A weighted moving average of either the previous predictions (*i.e.*, Temporal Ensembling (Laine et al. 2016)), or the model's parameters (*i.e.*, Mean Teacher (Tarvainen et al. 2017)), can be used to obtain more stable predictions over unlabeled examples. Instead of relying on random perturbations, Virtual Adversarial Training (VAT) (Miyato et al. 2018) approximates the perturbations that alter the model's predictions the most. Similarly, our proposed method enforces a consistency of predictions between the main decoder and the auxiliary decoders over different perturbations that are applied to the encoder's outputs rather than the inputs. The proposed CCT is also loosely related to Multi-View learning (J. Zhao et al. 2017) and Cross-View training (Clark et al. 2018), where each input to the auxiliary decoders can be viewed as an alternate but corrupt representation of the unlabeled examples.

Semi-Supervised Image Segmentation. A significant number of approaches uses a limited number of pixel-level labels together with a larger number of inexact annotations, *e.g.*, region-level (J. Dai et al. 2015a; C. Song et al. 2019) or image-level labels (J. Lee et al. 2019; K. Li et al. 2018;

Y. Wei et al. 2018; B. Zhou et al. 2016). For image-level based weak-supervision, primary localization maps are generated using Class Activation Mapping (CAM) (B. Zhou et al. 2016). After refining the generated maps, they can then be used to train a segmentation network together with the available pixel-level labels. Generative modeling can also be used for semi-supervised image segmentation (Hung et al. 2018; Souly et al. 2017) to take advantage of the unlabeled examples. Under the GAN framework, the discriminator's predictions are extended over pixel classes, and can then be jointly trained with a CE loss over the labeled examples and an adversarial loss over the whole dataset. In comparison, our proposed method exploits the unlabeled examples by enforcing a consistency over multiple perturbations on the hidden representations level, enhancing the encoder's representations and the overall performance with a small additional cost in terms of computation and memory requirements.

5.3 Preliminaries

We start with our observation and analysis of the cluster assumption in image segmentation, motivating the proposal of our CCT approach. A simple way to examine it is to estimate the local smoothness by measuring the local variations between the value of each pixel and its local neighbors. To this end, we compute the average euclidean distance at each spatial location and its 8 intermediate neighbors, for both the inputs and the hidden representations (*i.e.*, the ResNet's (K. He et al. 2016) outputs of a DeepLab v3 (L.-C. Chen et al. 2017b) trained on COCO (T.-Y. Lin et al. 2014)). For the inputs, following (French et al. 2019), we compute the average distance of a patch centered at a given spatial location and its neighbors to simulate a realistic receptive field. For the hidden representations, we upsample the feature map to the input size, and then compute the average distance between the neighboring activations (*i.e.*, 2048-dimensional feature vectors). The results are shown in Fig. 5.2. We observe that the cluster assumption is violated at the input level, given that the low-density regions do not align with the class boundaries. On the contrary, at the representation level, *i.e.*, the output of the encoder of a segmentation network, the cluster assumption is maintained where the class boundaries have high average distance, thus corresponding to low-density regions. Additionally, while the learned feature of a CNNs are generally more homogeneous and at higher layers, the network learns to compose low-level features into semantically meaningful representations while discarding high-frequency information (*e.g.*, texture). We observe that the learned features in a segmentation network seem to have a unique property; the class boundaries correspond to low-density regions. The same behavior is not observed in networks trained on other visual tasks (*e.g.*, classification and object detection). This observation motivates the proposed CCT approach, in which the perturbations are applied to the encoder's outputs rather than the inputs.

5.4 Method

5.4.1 Problem Definition

In SSL, we are provided with a small set of labeled training examples and a larger set of unlabeled training examples. Let $\mathcal{D}_l^{\text{tr}} = \{(\mathbf{x}_1^l, y_1), \dots, (\mathbf{x}_N^l, y_N)\}$ represent the N labeled examples and $\mathcal{D}_u^{\text{tr}} = \{\mathbf{x}_1^u, \dots, \mathbf{x}_M^u\}$ represent the M unlabeled examples, with $\mathbf{x}_i^u$ as the i-th unlabeled input

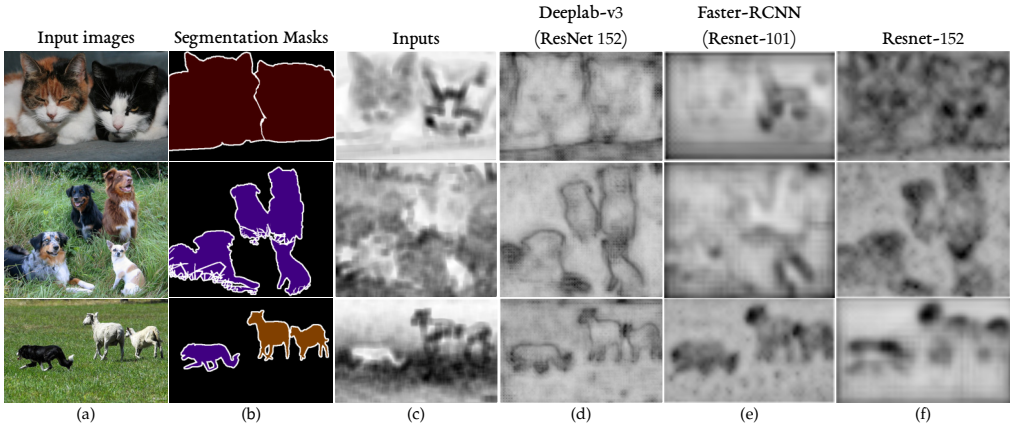


Figure 5.2: The cluster assumption in image segmentation. (a) Examples from PASCAL VOC 2012 train set. (b) Pixel-level ground-truths. (c) Input level. We show the average euclidean distance between each patch of size 20×20 centered at a given spatial location extracted from the input images, and its 8 neighboring patches. (d) Hidden representations level for a segmentation network. We show the average euclidean distance between a given 2048-dimensional activation at each spatial location and its 8 neighbors. Similarly, we show the results at the representations level for both an image classification (e) and an object detection (f) network. The darker the regions, the higher the average distance.

image, and $\mathbf{x}_i^l$ as the i-th labeled input image with spatial dimensions $H \times W$ and its corresponding pixel-level label $y_i \in \mathbb{R}^{C \times H \times W}$, where C is the number of classes.

In SSL, the objective is to exploit the larger number of unlabeled examples ($M \gg N$) to train a segmentation network f to perform well on the test data drawn from the same distribution as the training data. In this chapter, our architecture (see Fig. 5.3) is composed of a shared encoder h and a main decoder g, which constitute the segmentation network $f = g \circ h$. We also introduce a set of K auxiliary decoders g_a^k, with $k \in [1, K]$. While the segmentation network f is trained on the labeled set $\mathcal{D}_l^{\text{tr}}$ in a traditional supervised manner, the auxiliary networks $g_a^k \circ h$ are trained on the unlabeled set $\mathcal{D}_u^{\text{tr}}$ by enforcing a consistency of predictions between the main decoder and the auxiliary decoders. Each auxiliary decoder takes as input a perturbed version of the encoder's output while and the main encoder is fed the uncorrupted intermediate representations. This way, the representation learning of the encoder h is further enhanced using the unlabeled examples and, subsequently, that of the segmentation network f.

5.4.2 Proposed Method

As stated above, to extract additional training signal from the unlabeled set $\mathcal{D}_u^{\text{tr}}$, we rely on enforcing a consistency between the outputs of the main decoder g_m and those of auxiliary decoders g_a^k. Formally, for a labeled training example $\mathbf{x}_i^l$, and its pixel-level label y_i, the segmentation network f is trained using a Cross-Entropy (CE) based supervised loss $\mathcal{L}_s$:

$$\mathcal{L}_s = \frac{1}{|\mathcal{D}_l^{\text{tr}}|} \sum_{\mathbf{x}_i^l, y_i \in \mathcal{D}_l} \mathcal{L}_{\text{CE}}(y_i, f(\mathbf{x}_i^l)) \tag{5.1}$$

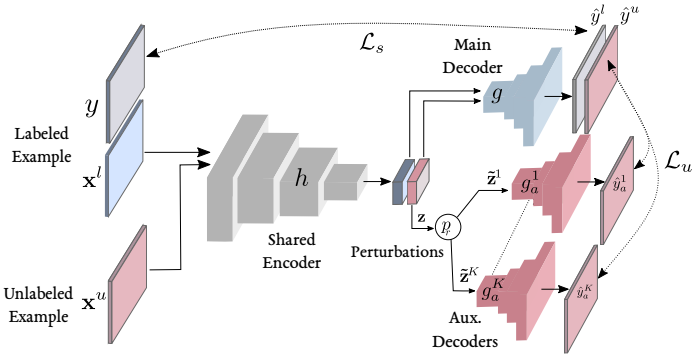


Figure 5.3: A TRAINING ITERATION OF CCT. For one training iteration, we sample a labeled input image $\mathbf{x}^l$ and its pixel-level label y together with an unlabeled image $\mathbf{x}^u$. We pass both images through the encoder and main decoder, obtaining two main predictions $\hat{y}^l$ and $\hat{y}^u$. We compute the supervised loss using the pixel-level label y and $\hat{y}^l$. We apply various perturbations to $\mathbf{z}$, the output of the encoder for $\mathbf{x}^u$, and generate K auxiliary predictions using the perturbed versions of the encoder's features. The unsupervised loss is then computed between the K outputs of these auxiliary decoders and that of the main decoder.

For an unlabeled example $\mathbf{x}_i^u$, an intermediate representation of the input is computed using the shared encoder $\mathbf{z}_i = h(\mathbf{x}_i^u)$[1]. Let us consider R stochastic perturbations functions, denoted as p_r with $r \in [1, R]$, where one perturbation function can be assigned to multiple auxiliary decoders. With various perturbation settings, we generate K perturbed versions $\tilde{\mathbf{z}}_i^k$ of the intermediate representation $\mathbf{z}_i$, so that the k-th perturbed version is to be fed to the k-th auxiliary decoder. For consistency, we consider the perturbation function as part of the auxiliary decoder, (*i.e.*, g_a^k can be seen as $g_a^k \circ p_r$ with p_r as the perturbation function). The training objective is then to minimize the unsupervised loss $\mathcal{L}_u$, which measures the discrepancy between the main decoder's output and that of the auxiliary decoders:

$$\mathcal{L}_u = \frac{1}{|\mathcal{D}_u^{\text{tr}}|} \frac{1}{K} \sum_{\mathbf{x}_i^u \in \mathcal{D}_u^{\text{tr}}} \sum_{k=1}^{K} \mathbf{d}(g(\mathbf{z}_i), g_a^k(\mathbf{z}_i)) \tag{5.2}$$

with $\mathbf{d}(.,.)$ as a distance measure between two output probability distributions (*i.e.*, the outputs of a $\mathrm{softmax}$ function applied over the channel dimension). In this chapter, we choose to use Mean Squared Error (MSE) as a distance measure.

The combined loss $\mathcal{L}$ for consistency-based SSL is then computed as:

$$\mathcal{L} = \mathcal{L}_s + \omega_u \mathcal{L}_u \tag{5.3}$$

where ω_u is an unsupervised loss weighting function. Following (Laine et al. 2016), to avoid using the initial noisy predictions of the main encoder, ω_u ramps up starting from zero along a Gaussian curve up to a fixed weight λ_u.

[1]Throughout this chapter, $\mathbf{z}$ will refer to the output of the encoder corresponding to an unlabeled input image $\mathbf{x}^u$.

Concretely, at each training iteration, an equal number of examples are sampled from the labeled $\mathcal{D}_L^{\text{tr}}$ and unlabeled $\mathcal{D}_U^{\text{tr}}$ sets. The supervised loss is computed using the main encoder's output and pixel-level labels. While for the unlabeled examples, we compute the MSE between the prediction of each auxiliary decoder and that of the main decoder. The total loss is then computed and back-propagated to train the segmentation network f and the auxiliary networks $g_a^k \circ h$. Note that the unsupervised loss $\mathcal{L}_u$ is not back-propagated through the main-decoder g, only the labeled examples are used to train g.

5.4.3 Method Details

Perturbation Functions

An important factor in consistency training is the perturbations to be applied to the hidden representation, *i.e.*, . the encoder's output $\mathbf{z}$. Next, we will introduce the three types of perturbation functions p_r we propose: feature-based, prediction-based, and random perturbations.

Feature-based perturbations. They consist of either injecting noise into or dropping some of the activations of the encoder's output feature maps $\mathbf{z}$.

- F-Noise. We uniformly sample a noise tensor $\mathbf{N} \sim \mathcal{U}(-0.3, 0.3)$ of the same size as $\mathbf{z}$. After adjusting its amplitude by multiplying it with $\mathbf{z}$, the noise is then injected into the encoder's output $\mathbf{z}$ to get $\tilde{\mathbf{z}} = (\mathbf{z} \odot \mathbf{N}) + \mathbf{z}$ with $\odot$ as the element-wise multiplication operation. This way, the injected noise is proportional to each activation.
- F-Drop. We first uniformly sample a threshold $\gamma \sim \mathcal{U}(0.6, 0.9)$. After summing over the channel dimension and normalizing the feature map $\mathbf{z}$ to get $\mathbf{z}'$, we generate a mask $\mathbf{M}_{\text{drop}} = \mathbb{1}_{\mathbf{z}' < \gamma}$[2], which is then used to obtain the perturbed version $\tilde{\mathbf{z}} = \mathbf{z} \odot \mathbf{M}_{\text{drop}}$. This way, we mask 10% to 40% of the most active regions in the feature map.

Prediction-based perturbations. They consist of adding perturbations based on the main decoder's prediction $\hat{y} = g(\mathbf{z})$ or that of the auxiliary decoders. We consider masking-based perturbations (Con-Msk, Obj-Msk and G-Cutout) in addition to adversarial perturbations (I-VAT).

- Guided Masking. Given the importance of context relationships for complex scene understanding (Oliva et al. 2007), the network might be too reliant on these relationships. To limit them, we create two perturbed versions of $\mathbf{z}$ by masking the detected objects (Obj-Msk) and the context (Con-Msk). Using the predicted masks $\hat{y}$, we generate an object mask $\mathbf{M}_{\text{obj}}$ to mask the detected foreground objects and a context mask $\mathbf{M}_{\text{con}} = 1 - \mathbf{M}_{\text{obj}}$, which are then down-sampled to match spatial dimensions of the encoder's features $\mathbf{z}$ and are then applied to them to get its two perturbed versions, *i.e.*, $\tilde{\mathbf{z}}_1 = \mathbf{z} \odot \mathbf{M}_{\text{obj}}$ and $\tilde{\mathbf{z}}_2 = \mathbf{z} \odot \mathbf{M}_{\text{con}}$.
- Guided Cutout (G-Cutout). In order to reduce the reliance on specific parts of the objects, and inspired by Cutout (Devries et al. 2017) that randomly masks some parts of the input image, we first find the possible spatial extent (*i.e.*, . bounding box) of each detected object

[2] $\mathbb{1}_{\text{cond}} \in \{0, 1\}$ as an indicator function evaluating to 1 iff cond is satisfied

using $\hat{y}$. We then zero out a random crop within each object's bounding box from the corresponding feature map $\mathbf{z}$.

- Intermediate VAT (I-VAT). To further push the output distribution to be isotropically smooth around each data point, we investigate using VAT (Miyato et al. 2018) as a perturbation function to be applied to $\mathbf{z}$ instead of the unlabeled inputs. For a given auxiliary decoder, we find the adversarial perturbation r_{adv} that will alter its prediction the most. The noise is then injected into $\mathbf{z}$ to obtain the perturbed version $\tilde{\mathbf{z}} = r_{adv} + \mathbf{z}$.

Random perturbations. (DropOut) Spatial dropout (Tompson et al. 2015) is also applied to $\mathbf{z}$ as a random perturbation.

Practical considerations

At each training iteration, we sample an equal number of labeled and unlabeled samples. As a consequence, we iterate on the set $\mathcal{D}_l^{\text{tr}}$ more times than on its unlabeled counterpart $\mathcal{D}_u^{\text{tr}}$ given that $M \gg N$ with M and N as the number of unlabeled and labeled training data, respectively. We thus risk overfitting the labeled set $\mathcal{D}_l^{\text{tr}}$. As such, to avoid overfitting, and motivated by Pohlen *et al*. (Pohlen et al. 2017) who observed improved results by sampling only 6% of the hardest pixels and Xie *et al*. (Xie et al. 2019) who showed an improvement when gradually releasing the supervised training signal in an SSL setting. We propose an annealed version of the bootstrapped-CE (ab-CE) in (Pohlen et al. 2017). With an output $f(\mathbf{x}_i^l) \in \mathbb{R}^{C \times H \times W}$ in the form of a probability distribution over the pixels, we only compute the supervised loss over the pixels with a probability less than a threshold η:

$$\mathcal{L}_s = \frac{1}{|\mathcal{D}_l^{\text{tr}}|} \sum_{\mathbf{x}_i^l, y_i \in \mathcal{D}_l} \mathbb{1}_{f(\mathbf{x}_i^l) < \eta} \, \mathcal{L}_{\text{CE}}(y_i, f(\mathbf{x}_i^l)) \tag{5.4}$$

During the beginning of training, the threshold parameter η is gradually increased from $\frac{1}{C}$ to 0.9, with C as the number of output classes. This way, we restrict the supervised training signal to examples in which the model is not overly confident.

5.4.4 Extensions

Exploiting weak-labels

In some cases, we might be provided with additional training data that is less expensive to acquire compared to pixel-level labels, *e.g*., image-level labels. Formally, instead of an unlabeled set $\mathcal{D}_u$, we are provided with a weakly labeled set $\mathcal{D}_w^{\text{tr}} = \{(\mathbf{x}_1^w, y_1^w), \ldots, (\mathbf{x}_m^w, y_m^w)\}$ alongside a pixel-level labeled set $\mathcal{D}_l^{\text{tr}}$, *i.e*., the unlabeled training set is replaced with a weakly labeled set, with y_i^w is the i-th image-level label corresponding to the i-th weakly labeled input image $\mathbf{x}_i^w$. The objective is to extract additional information from the weak labeled set $\mathcal{D}_w$ to further enhance the representations of the encoder h. To this end, we add a classification branch g_c consisting of a global average pooling layer followed by a classification layer, and pre-train the encoder for a classification task using binary CE loss.

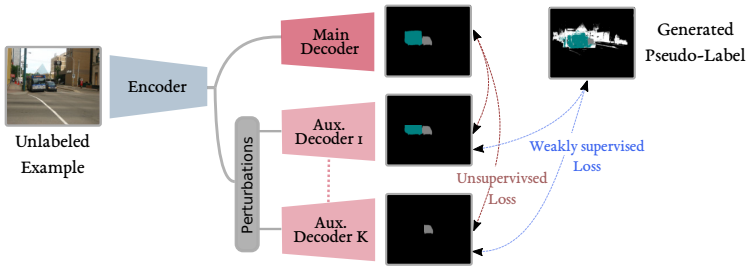


Figure 5.4: CCT WITH WEAK-LABELS. In case image-level labels are available, they can be leveraged to generated pseudo pixel-level labels where only confident regions are assigned a given class. These labels can then be used to train the aux. decoders together with the standard unsupervised consistency loss. Note that in this figure, the white regions of the generated pseudo label refer to ambiguous regions of the image that are not used during training.

Following previous works (Ahn et al. 2018; Z. Huang et al. 2018; J. Lee et al. 2019), the pre-trained encoder and the added classification branch can then be exploited to generate pseudo-pixel-level labels y_p. We start by generating the CAMs (*i.e.*, $M \in \mathbb{R}^{C\times H\times W}$) as in (B. Zhou et al. 2016), then we refine them to obtain the final pseudo labels y_p. To do this, first, we define two thresholds, a background τ_{bg} and a foreground τ_{fg} thresholds. Then, the pixels with attention scores less than τ_{bg} (*e.g.*, 0.05) are considered as background, while the pixels with an attention score larger than τ_{fg} (*e.g.*, 0.30) are assigned the class with the maximal attention score, and the rest of the pixels are ignored. After generating y_p, we conduct a final refinement step using dense CRF (Krähenbühl et al. 2011).

As illustrated in Fig. 5.4, in addition to considering $\mathcal{D}_w^{\text{tr}}$ as an unlabeled set and imposing a consistency over its examples, *i.e.*, enforcing a consistency of predictions at the encoder's level, the generated pseudo-labels are used to train the auxiliary networks $g_a^k \circ h$ using a weakly supervised loss $\mathcal{L}_w$. In this case, the loss in Eq. (5.3) becomes:

$$\mathcal{L} = \mathcal{L}_{\text{s}} + \omega_u \mathcal{L}_u + \omega_w \mathcal{L}_w \tag{5.5}$$

with

$$\mathcal{L}_w = \frac{1}{|\mathcal{D}_w^{\text{tr}}|}\frac{1}{K}\sum_{\mathbf{x}_i^w \in \mathcal{D}_w^{\text{tr}}}\sum_{k=1}^{K}\mathcal{L}_{\text{CE}}(y_p, g_a^k(\mathbf{z}_i)) \tag{5.6}$$

CCT ON MULTIPLE DOMAINS

In this section, we extend the proposed framework to a semi-supervised multi-domain setting. We consider the case in which two datasets $\{\mathcal{D}^{(1)}, \mathcal{D}^{(2)}\}$ with partially or fully non-overlapping label spaces are available. Each domain has its labeled and unlabeled sets, *i.e.*, $\mathcal{D}^{(i)} = \mathcal{D}_l^{(i)} \cup \mathcal{D}_u^{(i)}$. The objective is to simultaneously train a segmentation network to do well on the test data of both datasets, which are drawn from the different distributions.

We assume that enforcing a consistency over both unlabeled sets $\mathcal{D}_u^{(1)}$ and $\mathcal{D}_u^{(2)}$ might impose an invariance of the encoder's representations across the two domains. To this end, on top of the shared encoder h, we add domain-specific main decoder $g^{(i)}$ and auxiliary decoders $g_a^{k(i)}$.

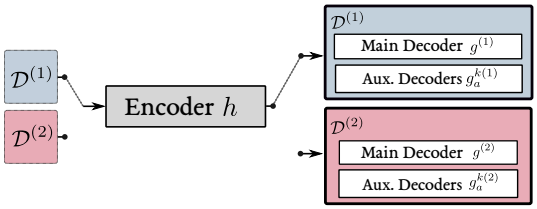


Figure 5.5: CCT ON MULTIPLE DOMAINS. On top of a shared encoder, we add domain specific main decoder and K auxiliary decoders. During training, we alternate between the two domains, sampling labeled and unlabeled examples and training the corresponding decoders and the shared encoder at each iteration.

Specifically, as show in Fig. 5.5, we add two main decoders and $2K$ auxiliary decoders on top of the encoder h. During training, we alternate between the two datasets, and at each iteration, we sample an equal number of labeled and unlabeled examples from each one, we compute the loss in Eq. (5.3) and train the shared encoder and the corresponding main and auxiliary decoders.

5.5 Experimental Results

We conduct detailed experiments to evaluate the proposed method and investigate its effectiveness in different settings. In Section 5.5.2, we present an extensive ablation study to highlight the contribution of each component within the proposed framework. Then in Section 5.5.3, we compare our method to state-of-the-art methods in semi-supervised and semi-supervised multi-domain settings, and show performances above previous methods.

5.5.1 Experimental Details

Network Architecture. In this chapter, we use an encoder-decoder based segmentation network (Section 4.2.2). The encoder is based on a ResNet-50 (K. He et al. 2016) pre-trained on ImageNet (Deng et al. 2009) provided by (You et al. 2019), and consists additionally of a PSP module (H. Zhao et al. 2017) for multi-scale processing. Following previous works (Ahn et al. 2018; Z. Huang et al. 2018; H. Zhao et al. 2017), the last two strided convolutions of ResNet are replaced with dilated convolutions. As for the decoders, taking the efficiency and the number of parameters into consideration, we choose to only use 1×1 Convs. After an initial 1×1 Conv to adapt the depth to the number of classes C, we apply a series of three sub-pixel convolutions (Shi et al. 2016) with ReLU non-linearities to upsample the outputs to the original input size.

Datasets. In a semi-supervised setting, we evaluate the proposed method on PASCAL VOC (Everingham et al. 2010), consisting of 21 classes (*i.e.*, with the background included) and three splits, training, validation, and testing, with 1464, 1449, and 1456 images, respectively. Following the common practice (Z. Huang et al. 2018; H. Zhao et al. 2017), we augment the training set with additional images from (Hariharan et al. 2011). Note that the pixel-level labels are only extracted from the original training set. For semi-supervised multi-domain experiments, for partially overlapping label spaces, we train on both Cityscapes (Cordts et al. 2016a) and CamVid (Brostow et al. 2009). Cityscapes is a finely annotated autonomous driving dataset with 19 classes. We are provided

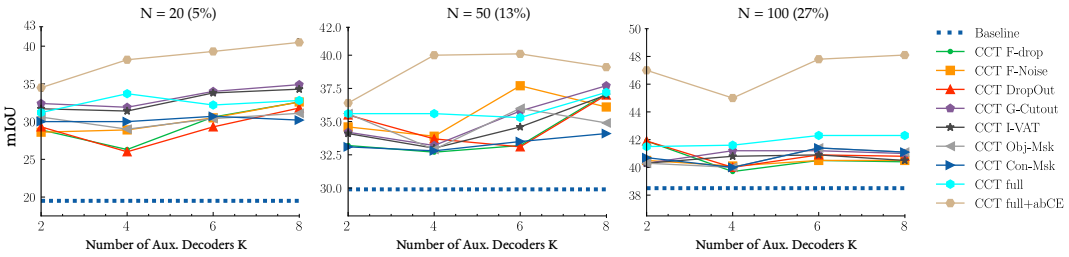


Figure 5.6: Ablation Studies on CamVid with 20, 50 and 100 labeled images. With different types of perturbations and a variable number of auxiliary decoders K, we compare the individual and the combined effectiveness of the perturbations to the baseline in which the model is trained only on the labeled examples. CCT full refers to using all of the 7 perturbations, *i.e.*, . the number of auxiliary decoders is $K \times 7$.

with three splits, training, validation, and testing with 2975, 500, and 1525 images, respectively. CamVid contains 367 training, 101 validation, and 233 testing images. Although originally the dataset is labeled with 38 classes, we use the 11 classes version (Badrinarayanan et al. 2017). For experiments over non-overlapping label spaces, we train on Cityscapes, and SUN RGB-D (S. Song et al. 2015). SUN RGB-D is an indoor segmentation dataset with 38 classes containing two splits, training and validation, with 5285 and 5050 images respectively. Similar to (Kalluri et al. 2019), we train on the 13 classes version (Handa et al. 2016).

Evaluation Metrics. For all of the datasets, we report the mIoU, *i.e.*, mean of class-wise intersection over union defined in Eq. (4.3).

Training Settings. Our implementation is based on the PyTorch 1.1 (Paszke et al. 2019) framework. For optimization, we train for 50 epochs using SGD with a learning rate of 0.01 and a momentum of 0.9. During training, the learning rate is annealed following the *poly* learning rate policy, where at each iteration, the base learning rate is multiplied by $1 - (\text{iter} \,/\, \text{max\_iter})^{0.9}$. For PASCAL VOC, we take crops of size 321×321 and apply random rescaling in the range of $[0.5, 2.0]$ and random horizontal flips. For Cityscapes, Cam-Vid and SUN RGB-D, following (Hung et al. 2018; Kalluri et al. 2019), we resize the input images to 512×1024, 360×480 and 480×640 respectively, without any further data-augmentation.

Inference Settings. For PASCAL VOC, Cityscapes, and SUN RGB-D, we report the results obtained on the validation set and on the test set of CamVid dataset.

Reproducibility. All the experiments are conducted on a V-100 GPUs. The implementation is available at `https://github.com/yassouali/CCT`.

5.5.2 Ablation Results

Aux. Decoders and Perturbation Functions

The proposed method consists of several types of perturbations and a variable number of auxiliary decoders. We thus start by studying the effect of the perturbation functions with different numbers of auxiliary decoders to provide additional insights into their individual performance

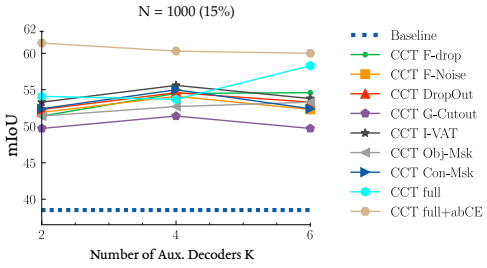


Figure 5.7: Ablation study on PASCAL VOC. Ablation study results with 1000 labeled examples using different perturbations and various numbers of auxiliary decoders K.

Splits	$N = 500$	$N = 1000$
Baseline	51.4	59.2
Mean Teachers	51.3	59.4
VAT	50.0	57.9
CCT	**58.6**	**64.4**

Table 5.1: CCT and Traditional Consistency Methods. We compare the performance of the baseline to the proposed method CCT, VAT, and Mean Teachers on PASCAL VOC. N represents the number of labeled examples.

and combined effectiveness. Precisely, we measure the effect of different numbers of auxiliary decoders K (*i.e.*, $K = 2$, 4, 6 and 8) of a given perturbation type. We refer to this setting of our method as *CCT {perturbation type}*, with seven possible perturbations. We also measure the combined effect of all perturbations resulting in $K \times 7$ auxiliary decoders in total and refer to it as *CCT full*. Additionally, *CCT full*+ab-CE indicates the usage of the annealed-bootstrapped CE as a supervised loss function. We compare the obtained results to the baseline, in which the model is trained only using the labeled examples.

CamVid. We carried out the ablation on CamVid with 20, 50 and 100 labels; the results are shown in Fig. 5.6. We find that each perturbation outperforms the baseline, with the most dramatic differences in the 20-label setting with up to 21 points. Surprisingly, we also observe an insignificant overall performance gap among different perturbations, confirming the effectiveness of enforcing the consistency over the hidden representations for image segmentation, and highlighting the versatility of CCT and its success with numerous perturbations. Increasing K results in a modest improvement overall, with the smallest change for Con-Msk and Obj-Msk due to their lack of stochasticity. Interestingly, we also observe a slight improvement when combining all of the perturbations, indicating that the encoder is able to generate representations that are consistent over many perturbations and, subsequently, improving the overall performance. Additionally, gradually releasing the training signal using ab-CE helps increase the performance with up to 8%, which confirms that overfitting of the labeled examples can cause a significant drop in performance.

PASCAL VOC. In order to investigate the success of CCT on larger datasets, we conduct additional ablation experiments on PASCAL VOC using 1000 labeled examples, The results are summarized in Fig. 5.7. We see similar results, where the proposed method shows notable improvement compared to the baseline with different perturbations, from 10 to 15 points. The combined perturbations yield a small increase in the performance, with the most significant difference obtained using $K = 6$. Furthermore, and similar to CamVid, when using the ab-CE loss, we see a significant gain with up to 7 points compared to CCT full.

Splits	$N = 500$	$N = 1000$
Baseline	51.4	59.2
CCT KL	54.0	62.5
CCT JS	58.4	64.3
CCT MSE	**58.6**	**64.4**

Table 5.2: CCT WITH DIFFERENT DISTANCE MEASURES. We compare the performance of MSE to the KL divergence and the JS-divergence on PASCAL VOC dataset.

	N	mIoU
CCT	1000	67.3 (+3.3)
CCT	1500	73.4 (+4)
CCT +9k Image-lvl labels	1500	**75.1** (+2.9)

Table 5.3: CCT RESULTS WITH MULTI-SCALE INFERENCE. The obtains gains in mIoU when we apply multi-scale inference on PASCAL VOC val set.

Based on the conducted ablation studies, for the rest of the experiments, we use the setting of *CCT full* with $K = 2$ for Con-Msk and Obj-Msk due to their lack of stochasticity, $K = 2$ for I-VAT given its high computational cost, and $K = 6$ for the rest of the perturbations, and refer to it as "CCT".

Comparison with Traditional Consistency Training Methods

In this section, we present the experiments to validate the observation that for image segmentation, enforcing a consistency over different perturbations applied to the encoder's outputs rather than the inputs is more aligned with the cluster assumption. To this end, we compare the proposed method with traditional consistency-based SSL methods. Specifically, we conduct experiments using VAT (Miyato et al. 2018) and Mean Teachers (Tarvainen et al. 2017). In VAT, at each training iteration, the unsupervised loss is computed as the KL divergence between the model's predictions with $\mathbf{x}^u$ and its perturbed version $\mathbf{x}^u + r_{adv}$ as inputs. For Mean Teachers; the discrepancy is measured using MSE between the prediction of the model and the prediction using an exponentially weighted version of it. In this case, the noise is sampled at each training step with SGD. The results are presented in Table 5.1. We see that applying the adversarial noise to inputs with VAT results in lower performance compared to the baseline. When using Mean Teachers, in which the noise is not implicitly added to the inputs, we obtain a similar performance to the baseline. These results confirm our observation that enforcing a consistency over perturbations applied to the hidden representations is more aligned with the cluster assumption, thus yielding better results.

Distance Measures

In all of the other experiments, we used MSE as the distance measure $\mathbf{d}(.,.)$ for the unsupervised loss $\mathcal{L}_u$ to measure the discrepancy between the main and auxiliary predictions. This section investigates the effectiveness of other distance measures between the output probability distributions. Specifically, we compare the performance of MSE to the KL divergence and the JS-divergence. The comparison results are shown in Table 5.2. We observe similar performances with $\mathbf{d}_{\text{MSE}}$ and $\mathbf{d}_{\text{JS}}$, while we only obtain 2.6 and 3.3 points gain for $N = 500$ and $N = 1000$ respectively over the baseline when using $\mathbf{d}_{\text{KL}}$. The low performance of $\mathbf{d}_{\text{KL}}$ might be due to its non-symmetric

Method	Pixel-level Labeled Examples	Image-level Labeled Examples	Val
WSSL (Papandreou et al. 2015b)	1.5k	9k	64.6
GAIN (K. Li et al. 2018)	1.5k	9k	60.5
MDC (Y. Wei et al. 2018)	1.5k	9k	65.7
DSRG (Z. Huang et al. 2018)	1.5k	9k	64.3
Souly *et al*. (Souly et al. 2017)	1.5k	9k	65.8
FickleNet (J. Lee et al. 2019)	1.5k	9k	65.8
Souly *et al*. (Souly et al. 2017)	1.5k	-	64.1
Hung *et al*. (Hung et al. 2018)	1.5k	-	68.4
CCT	1k	-	64.0
CCT	1.5k	-	69.4
CCT	1.5k	9k	**73.2**

Table 5.4: Comparison with the-state-of-the-art SSL methods. CCT performance on PASCAL VOC compared to other semi-supervised segmentation approaches.

nature. With $\mathbf{d}_{\mathrm{KL}}$, the auxiliary decoders are heavily penalized over sharp but wrong predictions, thus pushing them to produce uniform and uncertain outputs and reducing the amount of training signal that can be extracted from the unlabeled examples. However, with $\mathbf{d}_{\mathrm{JS}}$, which is a symmetrized and smoothed version of $\mathbf{d}_{\mathrm{KL}}$, we can bypass the zero avoidance nature of the KL divergence. Similarly, $\mathbf{d}_{\mathrm{MSE}}$ can be seen as a multi-class Brier score (Berthelot et al. 2019), which is less sensitive to completely incorrect predictions, giving it properties similar to $\mathbf{d}_{\mathrm{JS}}$, but with a lower computational cost.

Multi-scale Inference

To further enhance the predictions of our segmentation network, we conduct additional evaluations on PASCAL VOC using multi-scale inference (*i.e.*, image pyramid-based inference) to simulate a similar situation to training where we apply random scaling between 0.5 and 2, random cropping and random horizontal flip. For a given test image, we create 5 versions of it using 5 scales: 0.5, 0.75, 1, 1.25, and 1.5, and then also horizontal flips to each one, resulting in 10 versions of the original test image. The model's predictions are computed for each image, rescaled to the original size, and then aggregated by pixel-wise average pooling. The final segmentation maps are obtained by taking the argmax over the classes for each spatial location. In Table 5.3, we report the results obtained with multi-scale inference and show that with better preprocessing, we can obtain even better results under the CCT framework.

5.5.3 Quantitative Results

Semi-Supervised Setting

To further explore the effectiveness of our framework, we quantitatively compare it with previous semi-supervised image segmentation methods on PASCAL VOC. Table 5.4 compares CCT with such approaches. Our approach outperforms previous works relying on the same level of supervision and even methods that exploit image-level labels. We also observe an increase of 3.8 points when using additional image-level labels, affirming the flexibility of CCT, and the possibility of using it with different types of labels without any learning conflicts.

Method	$N = 50$			$N = 100$		
	CS	CVD	Avg.	CS	CVD	Avg.
Kalluri, *et al.* (Kalluri et al. 2019)	34.0	53.2	43.6	41.0	54.6	47.8
Baseline	31.2	40.0	35.6	37.3	34.4	35.9
CCT	35.0	53.7	**44.4**	40.1	55.7	**47.9**

Table 5.5: CCT applied to CS+CVD. CCT performance when training simultaneously on two datasets with overlapping label spaces, which are Cityscapes (CS) and CamVid (CVD).

Method	Labeled Examples	CS	SUN	Avg.
SceneNet (McCormac et al. 2017)	Full (5.3k)	-	49.8	-
Kalluri, *et al.* (Kalluri et al. 2019)	1.5k	58.0	31.5	44.8
Baseline	1.5k	54.3	38.1	46.2
CCT	1.5k	58.8	45.5	**52.1**

Table 5.6: CCT applied to CS+SUN. CCT performance when trained on both datasets Cityscapes (CS) and SUN RGB-D (SUN) datasets for the case of non-overlapping label spaces.

Semi-Supervised Multi-Domain Setting

In real-world applications, we are often provided with pixel-level labels collected from various sources, thus distinct data distributions. To examine the effectiveness of CCT when applied to multiple domains with a variable degree of labels overlap, we train our model simultaneously on two datasets, Cityscapes (CS) + CamVid (CVD) for partially overlapping labels, and Cityscapes + SUN RGB-D (SUN) for the disjoint case.

Cityscapes + CamVid. The results for CCT on Cityscapes and CamVid datasets with 50 and 100 labeled examples are given in Table 5.5. Similar to the SSL setting, CCT outperforms the baseline significantly, where the model is iteratively trained using only the labeled examples, with up to 12 points for $N = 100$. We even see a modest increase compared to previous work. This confirms our hypothesis that enforcing a consistency over different datasets does indeed push the encoder to produce invariant representation across different domains. And consequently, it increases the performance over the baseline while delivering similar results on each domain individually.

Cityscapes + SUN RGB-D. In the case of two domains with disjoint label spaces, we train on both Cityscapes and SUN RGB-D to demonstrate the capability of CCT to extract useful visual relationships and perform knowledge transfer between dissimilar domains, even in completely different settings. The results are shown in Table 5.6. Interestingly, despite the distribution mismatch between the datasets and the high number of labeled examples ($N = 1500$), CCT still provides a meaningful boost over the baseline with 5.9 points difference and 7.3 points boost compared to previous work. It show that by enforcing a consistency of predictions on the unlabeled sets of the two datasets over different perturbations, we can extract additional training signal and enhance the representation learning of the encoder. It is true even in the extreme case with non-overlapping label spaces and without any notable performance drop when an invariance of representations across both datasets is enforced at the level of the encoder's outputs.

5.5.4 Qualitative Results

Fig. 5.8 shows some qualitative results of the methods proposed in this chapter. Specifically, Fig. 5.8 (a) shows the generated pseudo-pixel-level labels using the available image-level labels that were generated following the method described in Section 5.4.4. We observe that when consid-

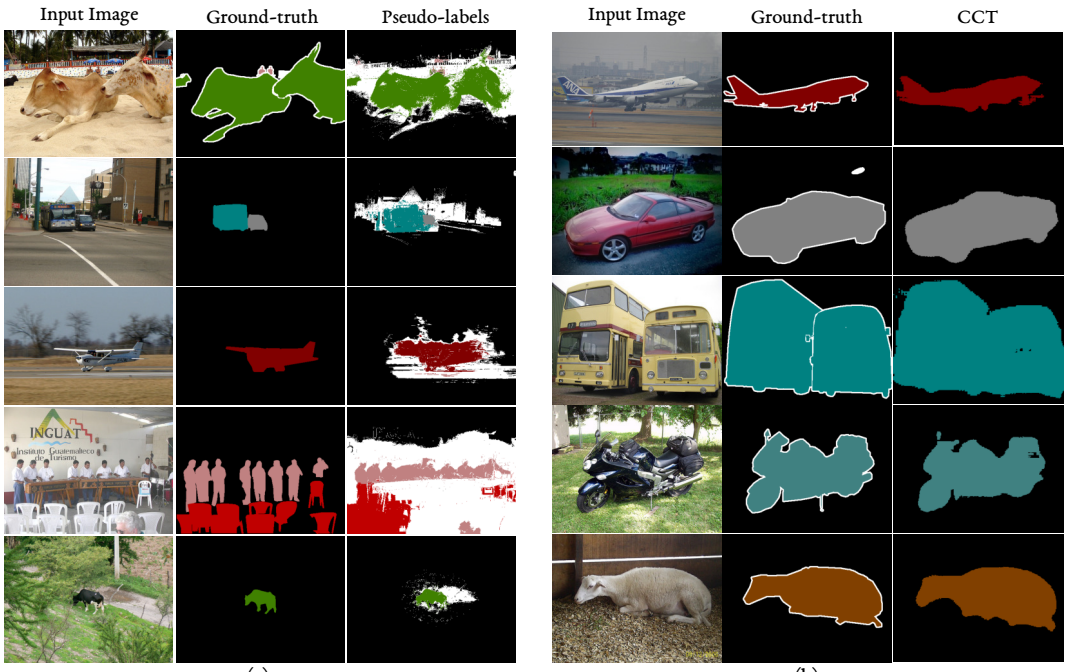


Figure 5.8: Qualitative Results. In (a), we show instances of the generated pseudo-pixel-level labels from PASCAL VOC train set. The white regions correspond to the ignored pixels. In (b), we show image segmentation results on the PASCAL VOC val images with CCT trained with 1.5k pixel-level and 9k image-level labels.

ering regions with high attention scores (*i.e.*, > 0.3), the assigned classes do correspond in most cases to true positives. In Fig. 5.8 (b), we show the obtained segmentation maps on PASCAL VOC *val* images with CCT trained on 1.5k pixel-level labels and 9k image-level labels. We observe that CCT is able to produce high-quality predictions that approach the correct ground-truth segmentations, further confirming the effectiveness of the proposed CCT.

Conclusion

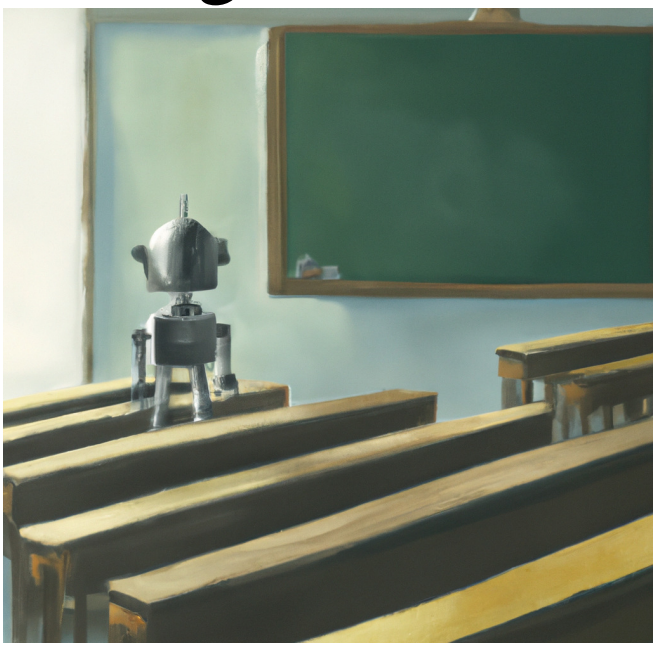
In this chapter, we presented cross-consistency training (CCT), a simple, efficient, and flexible method for a consistency based semi-supervised image segmentation, yielding state-of-the-art results. For future works, a possible direction worth exploring is the usage of other perturbations to be applied at different levels within the segmentation network. It would also be interesting to adapt and examine the effectiveness of CCT in other visual tasks and learning settings, such as unsupervised domain adaptation, a case we will investigate in Chapter 7.

In the next chapter, we will tackle a different learning paradigm, that of unsupervised learning, and set to propose a novel method for the same task of image segmentation.

Page intentionally left blank

CHAPTER 6

Autoregressive Segmentation



6 Autoregressive Segmentation

Chapter's Background

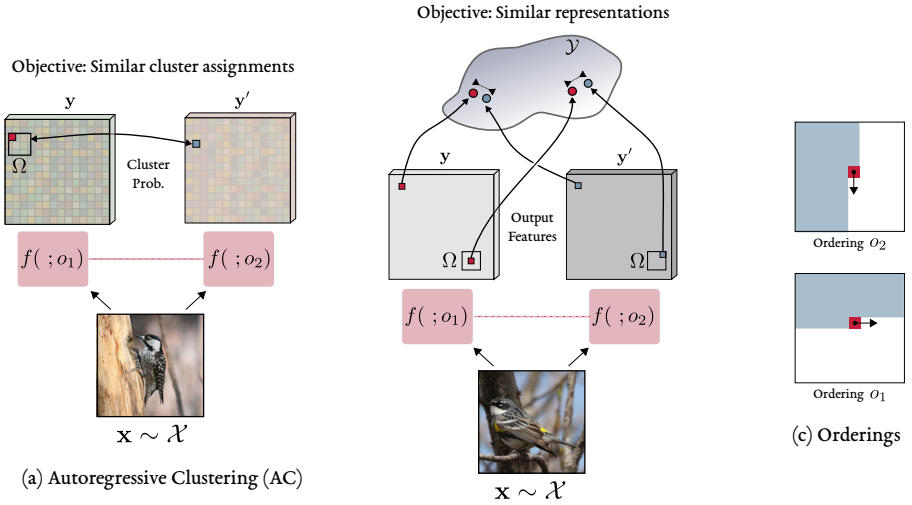
In this chapter, we consider:

- The Unsupervised Learning (UL) paradigm (Section 3.3) .
- The task of image segmentation (Sections 4.1.2 and 4.2.2).

When it comes to label-efficient learning, UL presents itself as the ideal setup by not requiring any labeled training data during the learning process. Additionally, the UL setting is a logical continuation of the setting in Chapter 5 since it is a natural extension of the SSL paradigm, *i.e.*, consists of removing the labeled training set. The choice of the image segmentation task was based on motivations similar to that Chapter 5. At the time of this contribution, DL-based unsupervised learning methods were under-studied in the context of segmentation compared to classification. Moreover, our aim is to be able to tackle a various visual tasks and not limit our applications to the standard classification task so that our contributions can be applicable complex tasks such as the document understanding such as the system presented in the introduction.

Chapter's Summary

In this chapter, we propose a new unsupervised image segmentation approach based on Mutual Information (MI) maximization between different constructed views of the inputs. Taking inspiration from autoregressive generative models (Section 3.3.1) that predict the current pixel from *past* pixels in a raster-scan ordering created with masked convolutions (Fig. 3.2), we propose to use different *orderings* over the inputs using various forms of masked convolutions to construct different *views* of the data. For a given input, and as shown in Fig. 6.1, the model produces a pair of predictions with two valid orderings, and is then trained to maximize the MI between the two outputs. These outputs can either be low-dimensional features for representation learning or output clusters corresponding to semantic labels for clustering. While masked convolutions are used during training, in inference, no masking is applied, and we fall back to the standard convolution where the model has access to the full input. The proposed method outperforms the current state-of-the-art on unsupervised image segmentation. It is simple and easy to implement, can be extended to other visual tasks, and can be integrated seamlessly into existing UL methods requiring different data views.



(a) Autoregressive Clustering (AC)

(b) Autoregressive Representation Learning (ARL)

(c) Orderings

Figure 6.1: Autoregressive Segmentation. Given an encoder-decoder type network f and two valid orderings (o_1, o_2) as illustrated in (c), the goal is to maximize the mutual information between the two outputs over the different *views*, *i.e.*, different *orderings*. (a) For Autoregressive Clustering (AC), we output the cluster assignments in the form of a probability distribution over pixels, and the goal is to have similar assignments regardless of the applied ordering. (b) For Autoregressive Representation Learning (ARL), the objective is to have similar representations at each corresponding spatial location and its neighbors over a window of small displacements Ω.

6.1 Introduction

As detailed in Section 3.3.2, recent discriminative UL works mainly focus on unsupervised representation learning and clustering. Representation learning aims to learn semantic features useful for downstream tasks, be it classification, regression or visualization. In clustering, the unlabeled data points are directly grouped into semantic classes. In both cases, recent works showed the effectiveness of maximizing MI between different *views* of the inputs to learn useful and transferable features (Federici et al. 2020; Hjelm et al. 2018a; A. v. d. Oord et al. 2018a; Tian et al. 2019) or discover clusters that accurately match semantic classes (Z. He et al. 2008; Ji et al. 2019). Another line of study in unsupervised learning is generative modeling. In particular, for image modeling, generative autoregressive models (X. Chen et al. 2018; A. v. d. Oord et al. 2016; Salimans et al. 2017; Van den Oord et al. 2016), such as PixelCNN (Fig. 3.2), are powerful generative models with tractable likelihood computation. In this case, the high-dimensional data, *e.g.*, an image $\mathbf{x}$, is factorized as a product of conditionals over its pixels. The generative model is then trained to predict the current pixel x_i based on the past values $x_{\leq i-1}$ in a raster scan fashion using masked convolutions (Van den Oord et al. 2016) (Fig. 6.3 (a)).

Instead of using a single left-to-right, top-to-bottom ordering, our method proposes to use several orderings obtained with different forms of masked convolutions and attention mechanism. The various *orderings* over the input pixels, or the intermediate representations, are then consid-

ered as different *views* of the input image[1]. The model is then trained to maximize the MI between the outputs over these different views. Our approach is generic and can be applied for both clustering and representation learning (see Fig. 6.1).

For a clustering task (Fig. 6.1 (a)), we apply a pair of distinct orderings over a given input image, producing two pixel-level predictions in the form of a probability distribution over the semantic classes. We then maximize the MI between the two outputs at each corresponding spatial location and its intermediate neighbors. Maximizing the MI helps avoid degenerate (*e.g.*, uniform output distributions) and trivial solutions (*e.g.*, assigning all of the pixels to the same cluster). For representation learning (Fig. 6.1 (b)), we maximize a lower bound of MI between the two output feature maps over the different *views*. We evaluate the proposed method using standard image segmentation datasets: Potsdam (Gerke 2014), and COCO-stuff (Caesar et al. 2018), and show competitive results. We present an extensive ablation study to highlight the contribution of each component within the proposed framework, emphasizing the method's flexibility.

Chapter's Contributions

To summarize, this chapter's contributions are:

- We propose a novel unsupervised method for image segmentation based on autoregressive models and MI maximization.
- We propose various forms of masked convolutions to generate different orderings.
- We propose an attention-augmented version of masked convolutions for a larger receptive field and a larger set of possible orderings.
- We demonstrate improved performances above previous state-of-the-art on unsupervised image segmentation.

6.2 Related Work

Autoregressive models. Many autoregressive models (X. Chen et al. 2018; Child et al. 2019; Germain et al. 2015; Larochelle et al. 2011; Parmar et al. 2018; Salimans et al. 2017; Van den Oord et al. 2016) for natural image modeling have been proposed. They model the joint probability distribution of high-dimensional images as a product of conditionals over the pixels. PixelCNN (A. v. d. Oord et al. 2016; Van den Oord et al. 2016) specifies the conditional distribution of a sub-pixel (*i.e.*, a color channel of a pixel) as a full 256-way softmax, while PixelCNN++ (Salimans et al. 2017) considers a continuous univariate distribution through a mixture of logistic distributions. In both cases, masked convolutions are used to process the initial image $\mathbf{x}$ in an autoregressive manner. In Image (Parmar et al. 2018) and Sparse (Child et al. 2019) transformers, self-attention (Vaswani et al. 2017) is used over the input pixels, while PixelSNAIL (X. Chen et al. 2018) combines both attention and masked convolutions.

[1] Throughout this chapter, a *view* refers to the application of a given *ordering*. Both are used interchangeably.

Clustering and unsupervised representation learning. Recent works in clustering aim at combining traditional clustering algorithms (Hartigan 1972) with deep learning, such as using K-means style objectives when training deep nets training (Caron et al. 2018; Fard et al. 2020; Haeusser et al. 2018). However, such an objective can lead to trivial and degenerate solutions (Caron et al. 2018). IIC (Ji et al. 2019) proposed to use a MI-based objective that is intrinsically more robust to such trivial solutions. Unsupervised learning of representations (Bachman et al. 2019; Gidaris et al. 2018b; Hjelm et al. 2018a; A. v. d. Oord et al. 2018a) rather aims to train a model, mapping the unlabeled inputs into some lower-dimensional space, while preserving semantic information and discarding instance-specific details. The pre-trained model can then be fine-tuned on a downstream task with fewer labels.

Unsupervised learning and MI maximization. Maximizing MI for unsupervised learning is not a new idea (Becker et al. 1992; Hartigan 1972), and recent works demonstrated its effectiveness for unsupervised learning. For representation learning, the training objective is to maximize a lower bound of MI over continuous random variables between distinct views of the inputs. These views can be the input image and its representation (W. Hu et al. 2017), the global and local features (Hjelm et al. 2018a), the features at different scales (Bachman et al. 2019), a sequence of extracted patches from an image in some fixed order (A. v. d. Oord et al. 2018a) or different modalities of the image (Tian et al. 2019). For a clustering objective with discrete random variables as outputs, the exact MI can be maximized over the different views such as IIC (Ji et al. 2019) that maximizes the MI between the image and its augmented version.

Unsupervised Image Segmentation. Fully unsupervised segmentation approaches can be categorized in three families as follows: i) GAN-based methods (Bielski et al. 2019; M. Chen et al. 2019) that extract and redraw the main object in the image for object segmentation. Such methods are limited to only instances with two classes, a foreground, and a background; ii) Iterative methods (Hwang et al. 2019) consisting of a two-step process. The features produced by a CNN are first grouped into clusters using spherical K-means. The CNN is then trained for better feature extraction to discriminate between the clusters. iii) MI maximization based methods (Ji et al. 2019) where the MI between two views of the same instance at the corresponding spatial locations is maximized.

6.3 Method

6.3.1 Problem Definition

We aim to learn a representation that maximizes the MI, denoted as I, between different input views. These views are generated using various orderings, capturing different aspects of the inputs. Following the same notation as in Part I, let $\mathbf{x} \sim \mathcal{X}$ be an unlabeled data point, and $f : \mathcal{X} \to \mathcal{Y}$ be a deep representation to be learned as a mapping between the inputs and the outputs. For clustering, $\mathcal{Y}$ is the set of possible clusters corresponding to semantic classes, and for representation learning, $\mathcal{Y}$ corresponds to a lower-dimensional space of the output features. Let $(o_i, o_j) \in \mathcal{O}$ be two orderings o_i and o_j obtained from the set of possible and valid orderings $\mathcal{O}$ (Fig. 6.2). For two outputs $y \sim f(\mathbf{x}; o_i)$ and $y' \sim f(\mathbf{x}; o_j)$, the objective is to maximize the predictability of y from y' and vice-versa, where $f(\mathbf{x}; o_i)$ corresponds to applying the learning function f with

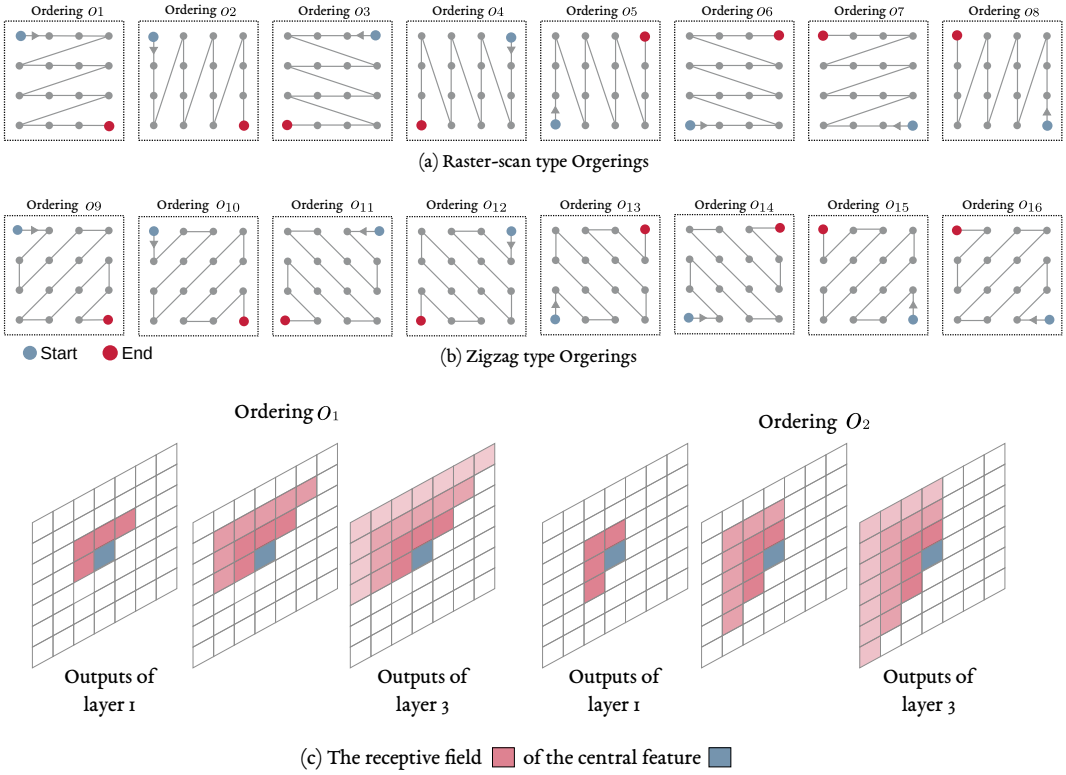


Figure 6.2: Orderings. We show the set of possible orderings $\mathcal{O}$ used in the proposed autoregressive segmentation. The set of possible orderings can contain either raster-type ordering (a), in which the pixels are processed one row at a time, or zigzag-type orderings (b), in which the pixels are processed diagonally or both. (c) By applying a series of convolutions with the correct causal masking, we obtain the desired receptive field where each feature only depends on the input pixels that comes before it depending on the ordering being used. Here we show the receptive fields for two ordering; o_1 and o_2.

a given ordering o_i to process the image $\mathbf{x}$. This objective is equivalent to maximizing the MI between the two encoded variables:

$$\max_{f} I(f(\mathbf{x}; o_i); f(\mathbf{x}; o_j)) \tag{6.1}$$

By maximizing Eq. (6.1) between the corresponding spatial locations of the two outputs, we push the neural network f to discard instance-specific details and high-frequency information, and generate similar and semantically meaningful representations regardless of the applied ordering.

6.3.2 Proposed Method

Orderings

In neural autoregressive modeling (X. Chen et al. 2018; Salimans et al. 2017; Van den Oord et al. 2016), for an input image $\mathbf{x} \in \mathbb{R}^{H \times W \times 3}$ with 3 color channels, a raster-scan ordering is first imposed on

the image (see Fig. 6.2 (a), ordering o_1), where the rows are processed top-to-bottom and the pixels of each row are processed from left to right. Such an ordering, where the pixel x_i only depends on the pixels that come before it, is maintained using masked convolutions[2] (A. v. d. Oord et al. 2016; Van den Oord et al. 2016).

Our proposition is to use all 8 possible raster-scan type orderings as the set of valid orderings $\mathcal{O}$ as illustrated in Fig. 6.2 (a). Additionally, to add more expressiveness and variability to our model, we extend this set with zigzag type orderings (Fig. 6.2 (b)). With zigzag orderings, the outputs at each spatial location will be mostly influenced by the values of the neighboring input pixels. Which can give rise to more semantically meaningful representations than raster-scan orderings. These orderings can be generated either based solely on masked convolutions for raster-scan type orderings, or with an additional attention operation for zigzag type orderings. The generation procedure will be detailed in Section 6.3.3 For an illustration of how the receptive field of a given feature grows using such masked convolutions to obtain the desired ordering, see Fig. 6.2 (c).

Training Objectives

In information theory, the MI $I(X;Y)$ between two random variables X and Y measures the *amount of information* learned from the knowledge of Y about X and vice-versa. The MI can be expressed as the difference of two entropy terms:

$$I(X;Y) = H(X) - H(X|Y) = H(Y) - H(Y|X) \tag{6.2}$$

with $H(.)$ as the marginal entropy and $H(.|.)$ as the conditional entropy. Intuitively, $I(X;Y)$ can be seen as the reduction of uncertainty in one of the variables when the other one is observed. If X and Y are independent, knowing one variable exposes nothing about the other, in this case, $I(X;Y) = 0$. Inversely, if the state of one variable is deterministic when the state of the other is revealed, the MI is maximized. Such an interpretation explains the goal behind maximizing Eq. (6.1). The neural network f must be able to preserve information and extract semantically similar representations regardless of the applied ordering o_i, and learn representations that encode the underlying shared information between the different views. The objective can also be interpreted as having a regularization effect, forcing the function f to focus on the different views and subparts of the input $\mathbf{x}$ to produce similar outputs, reducing the reliance on specific objects or parts of the image.

Let $p(y, y')$ be the joint distribution produced by sampling examples $\mathbf{x} \sim \mathcal{X}$ and then sampling two outputs $y \sim f(\mathbf{x}; o_i)$ and $y' \sim f(\mathbf{x}; o_j)$ with two possible orderings o_i and o_j. In this case, the MI in Eq. (6.1) can be defined as the KL divergence between the joint and the product of the marginals:

$$I(y, y') = D_{\mathrm{KL}}(p(y, y') \| p(y)p(y')) \tag{6.3}$$

To maximize Eq. (6.3), we can either maximize the exact MI for a clustering task over discrete predictions, or maximize a lower bound of it for unsupervised learning of representations over the continuous outputs. Next, we formulate the loss functions $\mathcal{L}_{\mathrm{AC}}$ and $\mathcal{L}_{\mathrm{ARL}}$ of both objectives for a segmentation task.

[2] Note that for a convolution weight tensor of shape $[F, F, d_{in}, d_{out}]$, the masking in applied over all values of both channels, d_{in} and d_{out}.

- Autoregressive clustering (AC). In a clustering task, the goal is to train a neural network f to predict a cluster assignment corresponding to a given semantic class $c \in \{1, \dots, C\}$ with C possible clusters at each spatial location. In this case, the encoder-decoder type network f is terminated with a C-way softmax, outputting $y \in [0, 1]^{H \times W \times C}$ of the same spatial dimensions as the input. Concretely, for a given input image $\mathbf{x}$ and two valid orderings $(o_i, o_j) \in \mathcal{O}$, we forward pass the input through the network producing two output probability distributions $p(y|\mathbf{x}, o_i) = f(\mathbf{x}; o_i)$ and $p(y'|\mathbf{x}, o_j) = f(\mathbf{x}; o_j)$ over the C clusters and at each spatial location. After reshaping the outputs into two matrices of shape $HW \times C$, with each element corresponding to the probability of assigning pixel x_l with $l \in \{1, \dots, HW\}$ to cluster c, we can compute the joint distribution $p(y, y')$ of shape $C \times C$ as follows:

$$p(y, y') = f(\mathbf{x}; o_i)^\top f(\mathbf{x}; o_j) \tag{6.4}$$

The marginals $p(y)$ and $p(y')$ can then be obtained by summing over the rows and columns of $p(y, y')$. Similar to IIC (Ji et al. 2019), we symmetrize $p(y, y')$ using $[p(y, y') + p(y, y')^\top]/2$ to maximize the MI in both directions. The clustering loss $\mathcal{L}_{AC}$ in this case can be written as follows:

$$\mathcal{L}_{\text{AC}} = \mathbb{E}_{\mathbf{x} \sim \mathcal{X}} \left[\mathbb{E}_{p(y,y')} \log \frac{p(y, y')}{p(y)p(y')} \right] \tag{6.5}$$

In practice, instead of only maximizing the MI between two corresponding spatial locations, we maximize it between each spatial location and its intermediate neighbors over small displacements $\mathbf{u} \in \Omega$ (see Fig. 6.1). This can be efficiently implemented using a convolution operation as demonstrated in (Ji et al. 2019).

- Autoregressive representation learning (ARL). Although the clustering objective in Eq. (6.5) can also be used as a pre-training objective for f, Tschannen *et al*. (Tschannen et al. 2019) showed that maximizing the MI does not often result in transferable and semantically meaningful features, especially when the downstream task is a priori unknown. To this end, we follow MI maximization-based representation learning works (Bachman et al. 2019; Hjelm et al. 2018a; A. v. d. Oord et al. 2018a; Tian et al. 2019), where a lower bound estimate of MI (*e.g.*, InfoNCE (A. v. d. Oord et al. 2018a), NWJ (Nguyen et al. 2010)) is maximized between different views of the inputs. These estimates are based on the simple, intuitive idea that, if a critic f is able to differentiate between samples drawn from the joint distribution $p(y, y')$ and samples drawn from the marginals $p(y)p(y')$, then the true MI is maximized.

In our case, with image segmentation as the target downstream task, we maximize the InfoNCE estimator (A. v. d. Oord et al. 2018a) over the continuous outputs. Specifically, with two outputs $(y, y') \in \mathbb{R}^{H \times W \times d}$ as d-dimensional feature maps, The training objective is to maximize the infoNCE based loss $\mathcal{L}_{\text{ARL}}$:

$$\mathcal{L}_{\text{ARL}} = \mathbb{E}_{\mathbf{x} \sim \mathcal{X}} \left[\log \frac{e^{f(y_l, y'_l)}}{\frac{1}{N} \sum_{m=1}^{N} e^{f(y_l, y'_m)}} \right] \tag{6.6}$$

For more details about this objective, refer to Section 3.3.2. For an input image $\mathbf{x}$ and two outputs y and y', let y_l and y'_m correspond to d-dimensional feature vectors at spatial positions

l and m in the first and second outputs, respectively. We start by creating N pairs of feature vectors (y_l, y'_m), with one positive pair drawn from the joint distribution and $N-1$ negative pairs drawn from the marginals. A positive pair is a pair of feature vectors corresponding to the same spatial locations in the two outputs, *i.e.*, a pair (y_l, y'_m) with $m = l$. The negatives are pairs (y_l, y'_m) corresponding to two distinct spatial positions $m \neq l$. In practice, we also consider small displacements Ω (Fig. 6.1) when constructing positives. Additionally, the negatives are generated from two distinct images, since two feature vectors might share similar characteristics even with different spatial positions. By maximizing Eq. (6.6), we push the model f to produce similar representations for the same spatial location regardless of the applied ordering so that the critic function f is able to give high matching scores to the positive pairs and low matching to the negatives. We follow (Hjelm et al. 2018a) and use separable critics $f(y, y') = \phi_1(y)^\top \phi_2(y')$, where the functions ϕ_1 and ϕ_2 non-linearly transform the outputs to a higher vector space to learn more complex representations that capture the underlying similarities of the features, so that $f(y_l, y'_m)$ produces a scalar corresponding to a matching score between the two representations at two spatial positions l and m of the two outputs.

Note that both losses $\mathcal{L}_{\text{AC}}$ and $\mathcal{L}_{\text{ARL}}$ can be applied interchangeably for both objectives, a case we investigate in our experiments (Section 6.4.2). For $\mathcal{L}_{\text{AC}}$, we can consider the clustering objective as an intermediate task for learning useful representations. For $\mathcal{L}_{\text{ARL}}$, during inference, K-means (J. Johnson et al. 2017) algorithm can be applied over the outputs to obtain the cluster assignments.

6.3.3 Method Details

Orderings Generation

- Generating Raster-scan Type Orderings. A simple way to generate raster-scan type orderings is to use a single ordering o_1 with the standard masked convolution (Fig. 6.3 (a)), along with geometric transformations g (*i.e.*, image rotations by multiples of 90 degrees and horizontal flips), resulting in 8 versions of the input image. We can then maximize the MI between the two outputs, *i.e.*, $I(y; g^{-1}(y'))$ with $y' \sim f(g(\mathbf{x}); o_j)$, assuming the transformation g is reversible. In this case, since the masked weights are never trained, we cannot fall back to the normal convolution where the function f has access to the full input during inference, greatly limiting the performance of such an approach.

This point motivates our approach. Our objective is to learn all the weights of the masked convolution during training and use an unmasked version during inference. This can be achieved by using a regular convolution, and for a given ordering o_i, we mask the corresponding weights during the forward pass to construct the desired view of the inputs. Then in the backward pass, we only update the unmasked weights, and the masked weights remain unchanged. In this case, all the weights will be learned, and we will converge to a normal convolution given enough training iterations. During inference, no masking is applied, giving the function f full access to the inputs.

A straightforward way to implement this is to use 8 versions of the standard masked convolution to create the set $\mathcal{O}$ (Fig. 6.3 (d)). However, for each forward pass, the majority of the weights are masked, resulting in a reduced receptive field and a fewer number of weights will be learned at each iteration, leading to some disparity between them.

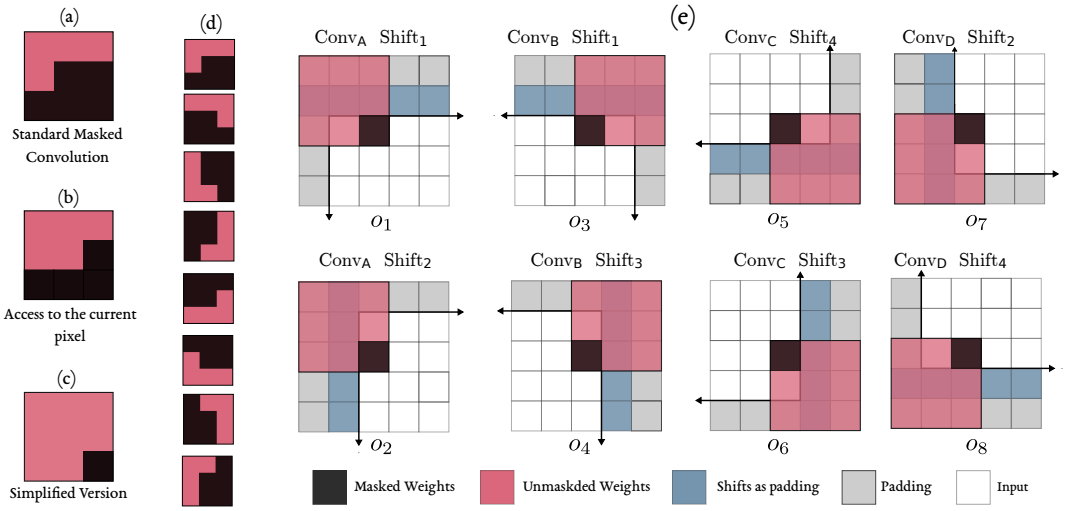


Figure 6.3: Masked Convolutions. (a) Standard masked convolution used in autoregressive generative modeling, yielding an ordering o_1. (b) A relaxed version of standard masked convolution where we have access to the current pixel at each step. (c) A simplified version of masked convolution with fewer masked weights. (d) The 8 versions of the standard masked convolution to construct all of the possible raster-scan type orderings. (e) The proposed types of masked convolutions with the corresponding shifts to obtain all of the 8 desired raster-scan types orderings. Here, we show a convolutional kernel of size $F = 3$.

Given that we are interested in a discriminative task rather than generative image modeling where access to the current pixel is not allowed we start by relaxing the conditional dependency and allow the model to have access to the current pixel, reducing the number of masked locations by one (Fig. 6.3 (b)). To further reduce the number of masked weights, for an $F \times F$ convolution, instead of masking the lower rows, we can shift the input by the same amount (*i.e.*, $\lfloor F/2 \rfloor$) and only mask the weights of the last row. We thus reduce the number of masked weights from $\lfloor F^2/2 \rfloor$ (Fig. 6.3 (b)) to $\lfloor F/2 \rfloor$ (Fig. 6.3 (c)). With four possible masked convolutions: $\{\text{Conv}_\text{A}, \text{Conv}_\text{B}, \text{Conv}_\text{C}, \text{Conv}_\text{D}\}$ and four possible shifts:[3] $\{\text{Shift}_1, \text{Shift}_3, \text{Shift}_2, \text{Shift}_4\}$, we can create all 8 raster-scan orderings as illustrated in Fig. 6.3 (e). The proposed masked convolutions do not introduce any additional computational overhead, neither in training nor inference, making them easy to implement and integrate into existing architectures with minor changes.

- Attention Augmented Masked Convolutions. As pointed out by (Van den Oord et al. 2016), the proposed masked convolutions are limited in terms of expressiveness since they create blind spots in the receptive field (Fig. 6.5). In our case, by applying different orderings, we will have access to all of the input $\mathbf{x}$ over the course of training, and this *bug* can be seen as a *feature* where the blind spots can be considered as an additional restriction. This restricted receptive field, however, can be overcome using the self-attention mechanism (Vaswani et al. 2017). Similar to previous works (Bello et al. 2019; X. Wang et al. 2018; H. Zhang et al. 2018a), we propose to add attention blocks to model long-range dependencies that are hard to access through standalone convolutions. Given

[3] *e.g.*, for Shift_1 and a 3×3 convolution, an image of spatial dimensions $H \times W$ is first padded on the top resulting in $(H + 1) \times W$, the last row is then cropped, going back to $H \times W$.

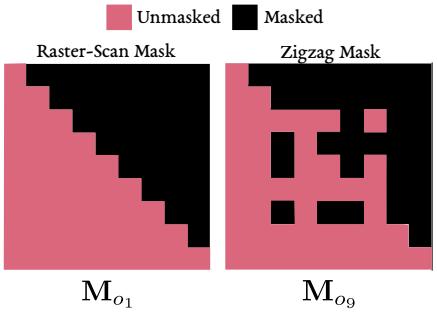


Figure 6.4: Attention Masks. Examples of the different attention masks $\mathbf{M}_{o_i}$ of shape $HW \times HW$ applied for a given ordering o_i. With $HW = 9$.

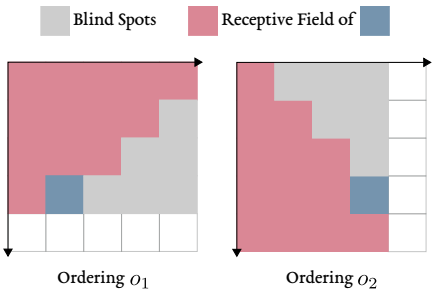


Figure 6.5: Blind Spots. Blind spots in the receptive field of pixel ■ as a result of using a masked convolution for a given ordering o_i.

an input tensor of shape (H, W, d_{in}), after reshaping it into a matrix $X \in \mathbb{R}^{HW \times d_{in}}$, we can apply a masked version of attention (Vaswani et al. 2017) in a straightforward manner. The output of the attention operation is:

$$A = \text{Softmax}((QK^\top) \odot \mathbf{M}_{o_i})V \tag{6.7}$$

with $Q = XW_q$, $K = XW_k$ and $V = XW_v$, where $W_q, W_k \in \mathbb{R}^{d_{in} \times d'}$ and $W_v \in \mathbb{R}^{d_{in} \times d'}$ are learned linear transformations that map the input X to queries Q, keys K and values V, and $\mathbf{M}_{o_i} \in \mathbb{R}^{HW \times HW}$ corresponds to a masking operation to maintain the correct ordering o_i.

The output is then projected into the output space using a learned linear transformation $W^O \in \mathbb{R}^{d' \times d_{in}}$ obtaining $X_{\text{att}} = AW^O$. The output of the attention operation X_{att} is concatenated channel wise with the input X, and then merged using a 1×1 Conv resulting in the output of the attention block.

- Generating Zigzag Type Orderings. Using attention gives us another benefit, we can extend the set of possible orderings to include zigzag type orderings introduced in (X. Chen et al. 2018) (see Fig. 6.2 (b)). With zigzag orderings, the pixels are processed in a diagonal order, where each output at a given spatial location will be mostly influenced by the values of its neighboring pixels, which is a more natural way to build pixel dependencies compared to that of raster-scan orderings. This is done by simply using a mask $\mathbf{M}_{o_i}$ corresponding to the desired zigzag ordering o_i. Resulting in a set $\mathcal{O}$ of 16 possible and valid orderings o_i with $i \in \{1, \ldots, 16\}$ in total. See Fig. 6.4 for an example.

Model

The representation f can be implemented in a general manner using three sub-parts, *i.e.*, $f = h \circ g_{ar} \circ d$, with a feature extractor h, an autoregressive encoder g_{ar} and a decoder d. With such a formulation, the function f is flexible and can take different forms. With h as an identity mapping, f becomes a fully autoregressive network, where we apply different orderings directly over the inputs. Inversely, if g_{ar} is an identity mapping, f becomes a generic encoder-decoder network, where h plays the role of an encoder. Additionally, h can be a simple convolutional

stem that plays an important role in learning local features such as edges or even multiple residual blocks (K. He et al. 2016) to extract higher representations. In this case, the orderings are applied over the hidden features using g_{ar}. g_{ar} is similar to h, containing a series of residual blocks, with two main differences, the proposed masked convolutions are used, and the batch normalization (Ioffe et al. 2015) layers are omitted to maintain the autoregressive dependency, with an optional attention block. The decoder d can be a simple 1×1 Conv to adapt the channels to the number of clusters C, followed by bilinear upsampling and a softmax operation for a clustering objective. For representation learning, d consists of two separable critics, ϕ_1 and ϕ_2, which are implemented as a series of {3×3 Convs - BN - ReLU} and a final 1×1 Conv for projecting the features to a higher dimensional space. In the experimental section, we will investigate different versions of f.

6.4 Experimental Results

After stating the experimental details, we present an extensive ablation study of the proposed autoregressive segmentation method and its various parts and different formulations. We then compare the method to state-of-the-art approaches to unsupervised image segmentation, followed by qualitative results.

6.4.1 Experimental Details

Datasets. The experiments are conducted on the following datasets: i) Potsdam (Gerke 2014) with 8550 RGB-IR satellite images of size 200×200, of which 3150 are unlabeled. We experiment on both the 6-labels variant (roads and cars, vegetation and trees, buildings and clutter) and Potsdam-3, a 3-label variant formed by merging each of the pairs. ii) COCO-Stuff (Caesar et al. 2018), a dataset containing *stuff* classes. Similarly, we use a reduced version of COCO-Stuff with 164k images and 15 coarse labels, reduced to 52k by taking only images with at least 75% stuff pixel. In addition to COCO-Stuff-3 with only 3 labels, sky, ground, and plants.

Evaluation Metrics. We report the pixel classification Accuracy (Acc) (refer to Eq. (4.2)). For a clustering task with a mismatch between the learned and ground truth clusters, we follow the standard procedure and find the best one-to-one permutation to match the output clusters to ground truth classes using the Hungarian algorithm (Kuhn 1955). The Acc. is then computed over the labeled examples.

Implementation Details. The different variations of f are trained using ADAM with a learning rate of 10^{-5} to optimize both objectives in Eqs. (6.5) and (6.6). We train on 200×200 crops for Potsdam and 128×128 for COCO. The training is conducted on NVidia V100 GPUs and implemented using the PyTorch framework (Paszke et al. 2019).

6.4.2 Ablation Results

We start by performing comprehensive ablation studies on the different components and variations of the proposed method. Table 6.1 and Fig. 6.6 show the ablation results for AC, and Table 6.2 shows a comparison between AC and ARL, analyzed as follows:

(a) **Variation of f:** we compare different variants of the network f using different feature extractors f and autoregressive encoders g_{ar}. The decoder d is fixed

Network $f = h \circ g_{ar} \circ d$				
	h	g_{ar}	POS	POS3
	Random		28.5	38.2
f_1	Id	5 Res. blocks	39.3	56.3
f_2	Stem	5 Res. blocks	46.4	**66.4**
f_3	Res. block	4 Res. blocks	**47.9**	64.5
f_4	5 Res. blocks	Id	35.1	63.4
f_5	ResNet-18	Id	40.7	51.9

(b) **Number of orderings:** we compare different sizes of the set $\mathcal{O}$. For $|\mathcal{O}|=2$ and $|\mathcal{O}|=4$ we report the mean and std over 4 runs using different possible pairs and quadruples respectively.

$\lvert\mathcal{O}\rvert$	POS	POS3
2	43.2 ± 2.19	59.5 ± 5.12
4	45.6 ± 3.22	63.55 ± 3.52
8	**46.4**	**66.4**

(c) **Attention:** we add a single attention block at a shallow level, and change the applied masks to get the desired orderings. Output stride = 4 in this instance.

Orderings				
Raster-Scan	Zigzag	Attention	POS	POS3
✓	×	×	45.2	61.0
✓	×	✓	47.9	66.3
×	✓	✓	47.8	**66.5**
✓	✓	✓	**49.3**	65.4

(d) **Sampling of o_i:** we compare different possible sampling procedures of the orderings o_i during training.

Sampling o_i	POS	POS3
Random	46.4	**66.4**
No Rep.	48.6	64.8
Hard	**48.9**	65.2

(e) **Transformations:** we apply a given transformation to the inputs of the second forward pass during a single training iteration.

Type	Transf.	POS	POS3
None	-	46.4	66.4
Photometric	Col. Jittering	47.9	65.5
Geometric	Flip	46.7	68.0
Geometric	Rot.	**48.5**	68.3
Geo. & Pho.	All	**48.5**	**68.3**

(f) **Dropout:** we inspect the addition of dropout to the inner activations of a residual block.

p	POS	POS3
0	46.4	**66.4**
0.1	**47.9**	64.7
0.2	46.9	65.1

Table 6.1: AC Ablations. Ablations studies conducted on Potsdam (POS) and Potsdam-3 (POS3) for Autoregressive Clusterings. We show the pixel classification accuracy (%).

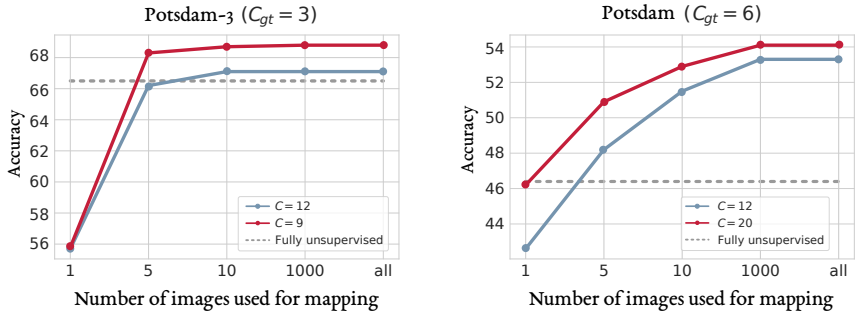


Figure 6.6: Overclustering. The Acc. (%) obtained when using a number of output clusters greater than the number of ground truth classes $C > C_{gt}$. We show in the x-axis a variable number of images is used to find the best many-to-one matching between the outputs and targets.

Variations of f. Table 6.1 (a) compares different variations of the network f. With a fixed decoder d (*i.e.*, a 1×1 Conv followed by bilinear upsampling and a softmax function), we adjust h and g_{ar} going from a fully autoregressive model (f_1) to a normal decoder-encoder network (f_4 and f_5). When using masked versions, we see an improvement over the normal case, with up to 8 points for Potsdam, and to a lesser extent for Potsdam-3 where the task is relatively easier with only three ground truth classes. When using a fully autoregressive model (f_1), and applying the orderings directly over the inputs, maximizing the MI becomes much harder, and the model fails to learn meaningful representations. Inversely, when no masking is applied (f_4 and f_5), the task becomes comparatively more straightforward, and we see a drop in performance. The best results are obtained when applying the orderings over low-level features (f_2 and f_3). Interestingly, the unmasked versions yield results better than random and perform competitively with 3 output classes for Potsdam-3, validating the effectiveness of maximizing the MI over small displacements $\mathbf{u} \in \Omega$. For the rest of the experiments, we use f_2 as our model.

Attention and different orderings. Table 6.1 (c) shows the effectiveness of adding attention blocks to our model. With a single attention block added at a shallow level, we observe an improvement over the baseline for both raster-scan and zigzag orderings and their combination, with up to 4 points for Potsdam. In this case, given the quadratic complexity of attention, we used an output stride of 4 (*i.e.*, the spatial dimensions of the output segmentation maps are $1/4$ of that input's dimensions).

Data augmentations. For a given training iteration, we pass the same image two times through the network, applying two different orderings at each forward pass. We can, however, pass a transformed version of the image as the second input. We investigate using photometric (*i.e.*, color jittering) and geometric (*i.e.*, rotations and H-flips) transformations. We bring the outputs back to the input coordinate space for geometric transformations before computing the loss. Results are shown in Table 6.1 (e). As expected, we obtain relative improvements with data augmentations, highlighting the flexibility of the approach.

Dropout. To add some degree of stochasticity to the network and as an additional regularization, we apply dropout to the intermediate activations within residual blocks of the network. Table 6.1 (f) shows a slight increase in Acc. for Potsdam.

Clustering		
Method	POS	POS3
Random CNN	28.5	38.2
AC	**46.4**	**66.4**
ARL	45.1	57.1

Linear Evaluation		
Method	POS	POS3
AC	**23.7**	**41.4**
ARL	23.7	38.5

Non-Linear Evaluation		
Method	POS	POS3
AC	**68.0**	**81.8**
ARL	47.6	63.5

Table 6.2: Comparing ARL and AC. We compare ARL and AC on a clustering task (left). We investigate the quality of the learned representations by freezing the trained model, and reporting the test Acc. obtained when training a linear (center) and non-linear (right) functions trained on the labeled training examples. We show the pixel classification accuracy (%).

	COCO-Stuff-3	COCO-Stuff	Potsdam-3	Potsdam
Random CNN	37.3	19.4	38.2	28.3
K-means (Pedregosa et al. 2011a)	52.2	14.1	45.7	35.3
SIFT (Lowe 2004)	38.1	20.2	38.2	28.5
Doersch 2015 (Doersch et al. 2015)	47.5	23.1	49.6	37.2
Isola 2016 (Isola et al. 2015)	54.0	24.3	63.9	44.9
DeepCluster 2018 (Caron et al. 2018)	41.6	19.9	41.7	29.2
IIC 2019 (Ji et al. 2019)	72.3	27.7	65.1	45.4
AC	**72.9**	**30.8**	**66.5**	**49.3**

Table 6.3: Unsupervised image segmentation. Comparison of AC with state-of-the-art methods on unsupervised segmentation. We show the pixel classification accuracy (%).

Orderings. Until now, at each forward pass, we sample a pair of possible orderings with replacement from the set $\mathcal{O}$. With such a sampling procedure, we might end up with the same pair of orderings for a given training iteration. As an alternative, we investigate two other sampling procedures. First, with no repetition (No Rep.), where we choose two distinct orderings for each training iteration. Second, using hard sampling, choosing two orderings with opposite receptive fields (*e.g.*, o_1 and o_6). Table 6.1 (d) shows the obtained results. We see 2 points of improvement when using hard sampling for Potsdam. For simplicity, we use random sampling for the rest of the experiments.

Additionally, to investigate the effect of the number of orderings (*i.e.*, the cardinality of $\mathcal{O}$), we compute the Acc. over different choices and sizes of $\mathcal{O}$. Table 6.1 (b) shows that better results can be obtained when using all 8 raster-scan orderings. Interestingly, we observe better results for some choices, which may be due to selecting orderings that do not share any receptive fields, like those used in hard sampling.

Overclustering. To compute the Acc. for a clustering task using linear assignment, the output clusters are chosen to match the ground truth classes, *i.e.*, $C = C_{gt}$. Nonetheless, we can choose a higher number of output clusters, *i.e.*, $C > C_{gt}$, and then find the best many-to-one matching between the output clusters and ground truths based on a given number of labeled examples. In this case, however, we are not in a fully unsupervised case, given that we extract some information, although limited, from the labels. Fig. 6.6 shows that even with a very limited number of labeled examples used for mapping, we can obtain better results than the fully unsupervised case.

AC and ARL. To compare AC and ARL, we apply them interchangeably on both clustering and representation learning objectives. For clustering, while with AC, we obtain the assignment directly, with ARL, we need to apply K-means over the output features after PCA Whitening to get the cluster assignments. As for representation learning, we evaluate the quality of the learned representations for both methods. We use linear and non-linear separability as a proxy for disentanglement and as a measure of MI between representations and class labels. Table 6.2 shows the obtained results, which are as follows:

- Clustering. As expected, AC outperforms ARL on a clustering task, given that the clusters are directly optimized by computing the exact MI during training.

- Quality of the learned representations. Surprisingly, AC outperforms ARL on both linear and non-linear classifications. We hypothesize that unsupervised representation learning objectives that work well on image classification fail in image segmentation due to the dense nature of the task. The model, in this case, needs to output distinct representations over pixels rather than the whole image, which is a more challenging task to optimize. This might also be due to using only a small number of features (*i.e.*, N pairs) for each training iteration. We note that the findings are only preliminary results and are worth further exploration.

6.4.3 Quantitative Results

After a detailed ablation study, we now present a comparison with the state-of-the-art unsupervised image segmentation in Table 6.3. As shown in the results, AC outperforms previous work, and by a good margin for more challenging segmentation tasks with a large number of output classes (*i.e.*, Potsdam and COCO-Stuff), highlighting the effectiveness of maximizing the MI between the different orderings as a training objective. We note that no regularization or data augmentation are used, and we expect that better results can be obtained by combining AC with other procedures as demonstrated in the ablation studies.

6.4.4 Qualitative Results

Fig. 6.7 shows some qualitative results of Autoregressive Clustering (AC) on COCO-stuff 3 *test* set. In Fig. 6.7 (a), we observe that the proposed fully unsupervised method can produce high-quality results and approaches the desired ground-truth results. However, as shown in Fig. 6.7 (b), we present some instances where AC fails. We notice that the model depends heavily on appearance and colors for making predictions. In certain settings, like tennis courts with grass or asphalt floors, the model predicts green or sky classes, and the correct prediction is ground. We postulate that these errors can be corrected with the usage of more elaborate data augmentations to push the model to be invariant to more cases and appearances. In the case where a small amount of labeled data is available, we can utilize either the over clustering approach shown Fig. 6.6, or conduct a small fine-tuning step to explicitly train the model to correct such mistakes.

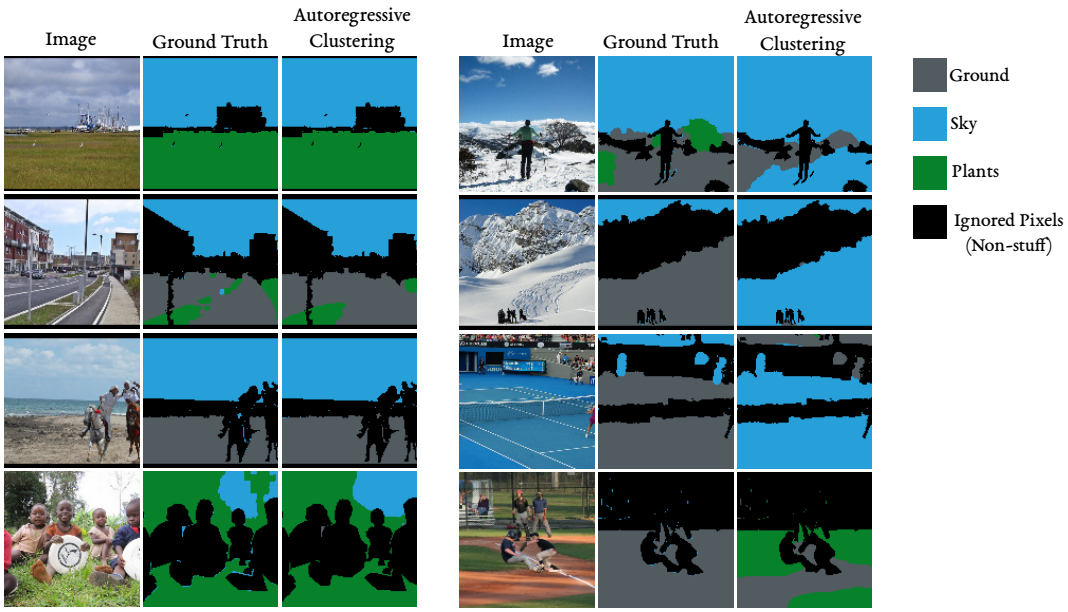


Figure 6.7: Qualitative Results. We show segmentation results on COCO-Stuff 3 (Caesar et al. 2018; Ji et al. 2019) *test* set with Autoregressive Clustering (AC). In (a), we show examples of good-quality segmentation results, while in (b), we show some failure cases of the proposed method.

Conclusion

In this chapter, we presented a novel method to create different *views* of the inputs using different *orderings*. We showed the effectiveness of maximizing the MI over these views for unsupervised image segmentation. We demonstrated that for image segmentation, optimizing over the discrete outputs by computing the exact MI works better for both clustering and unsupervised representation learning due to the dense nature of the task. Given the method's simplicity and ease of adoption, we hope that the proposed approach can be adapted for other visual tasks and used in future works.
In the next chapter, we tackle a different learning paradigm, that of unsupervised domain adaptation, and set to propose a novel method for both image classification and segmentation tasks.

Page intentionally left blank

CHAPTER 7

Target Consistency

7 Target Consistency

Chapter's Background

In this chapter, we consider:

- The Unsupervised Domain Adaptation (UDA) paradigm (Section 3.4).
- The tasks of image classification (Sections 4.1.1 and 4.2.1) and image segmentation Sections 4.1.2 and 4.2.2).

In the first two contributions chapters, the test data was considered to be collected from the same domain as the training data. Thus, it is unrealistic and not aligned with what we encounter in practice. As such, in this chapter, we consider the UDA paradigm that deals with such a setup. UDA can also be seen as a natural extension of Chapter 5, where we take into account a distribution shift between the labeled and unlabeled training sets. Furthermore, under the UDA setting, the amount of human labor required in the annotation process can be significantly reduced by using a synthetic label set for which the labels can be automatically generated. Thus resulting in an annotation process that is expeditious and inexpensive, and making it relevant to us as a label-efficient setting. In terms of the tasks, and given the proposed approach's flexibility, we opted for both image classification, which is very popular in the UDA with strong benchmarks. In addition to the segmentation task, which is an important visual and relevant task for our document understanding use case.

Chapter's Summary

In this chapter, we start by investigating the robustness of domain invariant UDA methods under the prism of the cluster assumption. We bring new evidence that invariance with a low source risk does not guarantee a well-performing target classifier. More precisely, we show that the cluster assumption is violated in the target domain despite being maintained in the source domain, indicating a lack of robustness of the target classifier. To address this problem, we demonstrate the importance of enforcing the cluster assumption in the target domain, named Target Consistency (TC), especially when paired with Class-Level InVariance (CLIV). The proposed approach shows a notable performance improvement in image classification and segmentation benchmarks over state-of-the-art methods based on invariant representations. Importantly, our approach is flexible and easy to implement, making it a complementary technique to existing approaches for improving the transferability of representations.

7.1 Introduction

Deep learning (DL) models often show a weak ability to generalize on samples significantly different from those seen during training (Arjovsky et al. 2019; Beery et al. 2018; Geva et al. 2019). This inability to generalize out of the training distribution presents a significant obstacle to a controlled and safe deployment of DL models in real-world systems (Amodei et al. 2016; Marcus 2020). To bridge such a distributional gap, UDA (Section 3.4) leverages labeled samples from a well-known domain, referred to as source, to generalize on a target domain, where only unlabeled samples are available. If the labeling functions are equal across domains, a situation known as the covariate shift, adaptation can be performed by weighting sample contributions in the loss (Gretton et al. 2012; Quionero-Candela et al. 2009; Sugiyama et al. 2007, 2008). However, for high-dimensional data, such as text or images, it is unlikely that the source and the target distributions share enough statistical support to compute weights (Johansson et al. 2019). Learning domain invariant representations, *i.e.*, representations for which it is impossible to distinguish the domain they were sampled from, can bring together two domains in some feature space which are different in the input space (Ganin et al. 2015; M. Long et al. 2015), thus helping bridge the gap between the source and target domains.

Nevertheless, the invariance of representations does not always guarantee a low target risk. For instance, in the case of images, aligning source and target backgrounds reduces domain discrepancy of representations but is unlikely to improve the model in the target domain. Even worse, it may incorrectly align source and target classes if the background is incorrectly correlated with a given class due to some collection bias (Arjovsky et al. 2019; Beery et al. 2018), a phenomenon known as negative transfer (Torrey et al. 2010). One example of one of the negative impacts domain invariance-based methods can have is its effect on the learned classifier when applied to the target domain, *i.e.*, a sensitive target classifier.

In this chapter, we aim to provide a new understanding of the transferability of representations through the prism of the cluster assumption. The cluster assumption states that if samples are of the same cluster in the input space, they are likely to be of the same output class. When enforced on unlabeled samples, the model benefits from a significant gain in generalization (Chapelle et al. 2009; Sohn et al. 2020; Xie et al. 2019) and robustness (Carmon et al. 2019; Hendrycks et al. 2020). We show that enforcing the cluster assumption in the target domain, coupled with a per-class domain invariant representations (*i.e.*, Class-Level Invariance, or CLIV), we can obtain notable performance gains on various image classification and segmentation UDA benchmarks, thus confirming the effectiveness of our proposed approach.

Chapter's Contributions

To summarize, this chapter's contributions are:

- We show that domain invariance induces a model with significant sensitivity to perturbations in the target domain through an in-depth empirical analysis, indicating that invariance is achieved by disregarding principles of robustness. Such evidence motivates our interest in enforcing the cluster assumption on target and further improving the transferability of domain invariant representations.

- We propose Target Consistency (TC) used to explicitly enforce a consistency of predictions over the target domain and with respect to a set of carefully chosen perturbations.
- We perform a per-class representation alignment with a Class-Level InVariance (CLIV) objective while enforcing the cluster assumption to amplify the effect of TC. This results in a positive feedback loop between decision boundary updates and representation alignment.
- We show, with extensive experiments on both classification and segmentation datasets, that we reach state-of-the-art performances for methods based on invariant representations.

7.2 Related Work

In this chapter, we focus on two lines of works that are relevant to us: i) Consistency regularization-based methods. Which are SSL approaches that train the prediction function to produce similar outputs for similar inputs, and by enforcing such a constraint, the resulting decision boundary will lie in low-density regions, echoing a more robust model (Section 3.2). And ii) invariant representation-based method. Which are UDA approaches that force the feature extractor to produce similar representations regardless of the domain the inputs come from. Such that a classifier trained on the source features is applicable on target (Section 3.4). In this context, the most relevant methods for us are those that integrate consistency-based methods with invariant representations for UDA. To our knowledge, DIRT-T (Shu et al. 2018) is the first work that proposed such a method. Our work differs from it by exploring the connection between class-level invariance and target consistency without enforcing a source consistency. Transferable Adversarial Training (H. Liu et al. 2019) also explored the role of consistency in the representation space for bridging the distributional gap without hurting transferability. Our approach investigates consistency *w.r.t* perturbations in the input space as a strong inductive bias for improving the transferability of representations.

7.3 Preliminaries

7.3.1 Problem Definition

Domain adaptation introduces two domains, the *source* and the *target* domains, on the product space $\mathcal{X} \times \mathcal{Y}$ where $\mathcal{X}$ is the input pace and $\mathcal{Y}$ is the label space. These two domains are defined by their specific joint distributions of inputs $\mathbf{x} \in \mathcal{X}$ and labels $y \in \mathcal{Y}$, noted $p_s(\mathbf{x}^s, y^s)$ and $p_t(\mathbf{x}^t, y^t)$, respectively. Considering a hypothesis class $\mathcal{H}$, a subset of functions from $\mathcal{X}$ to $\mathcal{Y}$, domain adaptation aims to learn an inference function $f \in \mathcal{H}$ which performs well in the target domain, *i.e.*, has a small target risk $\varepsilon^t(f) = \mathbb{E}_{(\mathbf{x}^t, y^t)\sim p_t}[\mathcal{L}(f(\mathbf{x}^t), y^t)]$ where $\mathcal{L}$ is a given loss to be minimized. UDA considers the case where labeled samples are available in the source domain, while the target domain is only represented with unlabeled samples. To tackle such a setting,

many methods start from the theoretical analysis of domain adaptation presented by (Ben-David et al. 2010) in which a lower bound of the target error for any hypothesis $f \in \mathcal{H}$ is introduced:

$$\varepsilon^t(h) \leq \varepsilon^s(h) + d_{\mathcal{H}\Delta\mathcal{H}} + \inf_{h\in\mathcal{H}} \{\varepsilon^t(h) + \varepsilon^s(h)\} \tag{7.1}$$

where $d_{\mathcal{H}\Delta\mathcal{H}}$ is a domain dissimilarity metric known as the symmetric difference hypothesis divergence, and consists of finding a pair of classifiers f and f' with the largest disagreement between the source and the target domains:

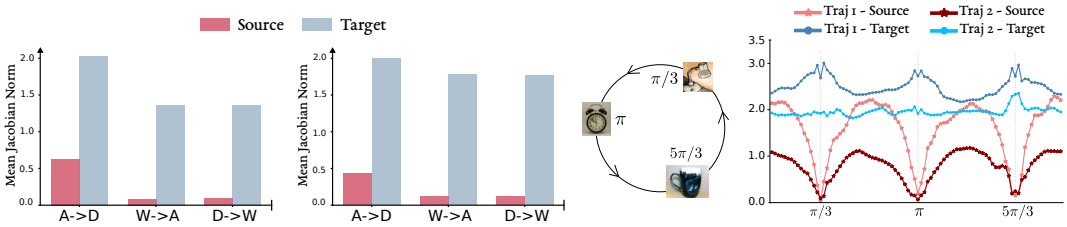
$$d_{\mathcal{H}\Delta\mathcal{H}} = 2 \sup_{f,f'\in\mathcal{H}} |\mathbb{E}_{\mathbf{x}^s\sim p_s}[f(\mathbf{x}^s) \neq f'(\mathbf{x}^s)] - \mathbb{E}_{\mathbf{x}^t\sim p_t}[f(\mathbf{x}^t) \neq f'(\mathbf{x}^t)]| \tag{7.2}$$

One of the most popular UDA approaches for reconciling two non-overlapping data distributions (Ganin et al. 2015; M. Long et al. 2015) is learning domain invariant representations. It consists of adversarially training a deep feature extractor such that a domain discriminator can not distinguish the target's representations and those of source (Ganin et al. 2015). Specifically, our model $f = g \circ h$ is composed of a feature extractor $g : \mathcal{X} \to \mathcal{Z}$ and a classifier $h : \mathcal{Z} \to \mathcal{Y}$, in addition to a domain classifier $d : \mathcal{Z} \to [0, 1]$. As specified in Eq. (3.10), the domain adversarial training objective then consists of training the whole model (*i.e.*, the feature extractor and the classifier) to minimize the source Cross-Entropy (CE) loss. While the domain classifier and the feature extractor are trained adversarially to learn a feature extractor that minimizes the Jensen-Shannon divergence between the features of the two domains. When both the source error and the feature divergence are minimized, we can expect a reasonable degree of target generalization and sufficient adaptation. However, as Eq. (7.2) shows, to obtain adequate adaptation and minimize $d_{\mathcal{H}\Delta\mathcal{H}}$, we need to find a good trade-off between hypotheses with low target and source errors. But since our feature extractor g is often a deep neural network with very high capacity, it can apply arbitrary transformations to the target domain to reduce the feature divergence, while still maintaining the optimal low source error, which does not imply a better target generalization (Shu et al. 2018) nor sufficient adaptation. Thus, such a training objective is not sufficient for acceptable target generalization. Next, we will empirically confirm such an intuition by measuring the model's sensitivity in the target domain, showing that the obtained classifier is indeed not the optimal target classifier. Based on these findings, we will then propose TC as a way to enforce the cluster assumption in the target domain to reject target hypotheses that are clearly suboptimal. Resulting in a more optimal source-target trade-off, better target performances, and overall lower domain divergence.

7.3.2 Target Sensitivity

For target sensitivity analysis, we will compare the robustness of a model trained to minimize either the source error, *i.e.*, standard CE-based training using the labeled source data solely, or the source error and the domain adversarial loss[1] This comparison will be conducted by measuring the model's outputs sensitivity to small perturbations in the input space. To measure this sensitivity,

[1] We will also refer to this domain adversarial loss as Domain Adversarial Neural Networks or DANN objective.



(a) Source CE only (b) Source CE + DANN (c) Trajectory (c) Regions of Sensitivity

Figure 7.1: Sensitivity Analysis. (a) An illustration of the circular trajectory passing through three images of different classes. (b) Jacobian norm of source (D) and target (A) as the input traverses two elliptical trajectories: Trajectory 1: different classes. Trajectory 2: same classes, for a ResNet-50 trained on source only. (c) and (d) The mean Jacobian norm on target and source domains of a ResNet-50 when trained on source only and with a DANN objective on three Office-31 tasks.

we follow (Novak et al. 2018) and compute the mean Jacobian norm as a proxy measure of the local sensitivity of the model on target examples $\mathbf{x}^t$ sampled from the target distribution p_t:

$$\mathbb{E}_{\mathbf{x}^t \sim p_t}\left[\left\|J\left(\mathbf{x}^t\right)\right\|_F\right] \text{ where } J(\mathbf{x}) = \left(\frac{\partial \hat{y}}{\partial \mathbf{x}}\right)^\top \tag{7.3}$$

where $J_{ij}(\mathbf{x}) = \partial \hat{y}_i / \partial x_j$ is the Jacobian matrix, $\|J\|_F$ is the Frobenius norm, x_j is a pixel in the input image $\mathbf{x}$ and $\hat{y}_i$ is the output probability for class i. For comparison, the sensitivity in the source domain can be computed similarly over source instances.

In this section, we will use the Office-31 dataset (Saenko et al. 2010). In Office-31, we are provided with labeled examples depicting 31 object categories from three domains, A (Amazon), (D) DSLR, and W (Webcam), giving us the possibility to construct 6 possible UDA tasks. For a given task, two of the three domains are considered as a labeled source domain and an unlabeled target domain. *E.g.*, A $\rightarrow$ D signifies a task where the Amazon domain is the source, and the DSLR domain is the target. Specifically, we will present the sensitivity analysis on 3 transfer tasks: A $\rightarrow$ D, W $\rightarrow$ A and D $\rightarrow$ W, which are shown in Fig. 7.1 (a) and Fig. 7.1 (b). As suspected, the target sensitivity is significantly higher than the source sensitivity. Importantly, when enforcing an invariance of representations with DANN, sensitivity in the target domain decreases for tasks W $\rightarrow$ A and D $\rightarrow$ W while remaining significantly higher than the source sensitivity. This validates our concern that even after feature alignment, the resulting classifier still violates the cluster assumption in the target domain.

To further investigate the regions of sensitivity, we examine the behavior of the function on and off the data manifold as it approaches and moves away from three anchor points. Following Novak *et al*. (Novak et al. 2018), we analyze the behavior of the model near and away from target and source data along two types of trajectories: i) an ellipse passing through three data points of different classes as illustrated in Fig. 7.1 (c), and ii) an ellipse passing through three data points of the same class. Since linear combinations of images of the same class are likely to look more like realistic images, the second trajectory is expected to traverse closer to the data manifold overall. Fig. 7.1 (d) shows the obtained results. According to the Jacobian norm, we observe that the model's sen-

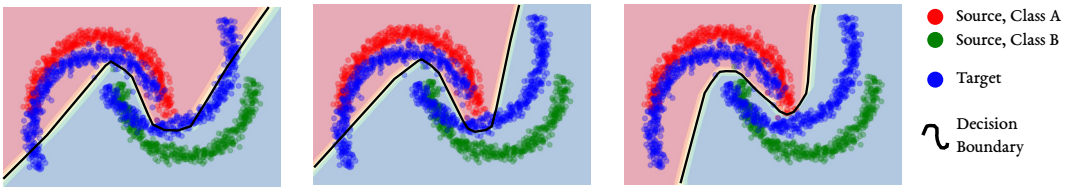


Figure 7.2: Effect of TC on a toy dataset. Effect of TC on two moons dataset. Red and green points are the instances of the two classes of the source domain. Blue points are target samples generated by rotating source samples. The black line shows the learned decision boundary with three training objectives: source CE, with a DANN objective, and with target consistency.

sitivity in the vicinity of target data is comparable to its sensitivity off the data manifold. Inversely, the model remains relatively stable in the neighborhood of source data, and becomes unstable only away from them, further confirming our hypothesis that the target classifier's decision boundary is not well placed.

7.4 Method

7.4.1 Proposed Method

Target Consistency. To promote a more robust model and mitigate target sensitivity, we regularize the model's predictions to be invariant to a set of perturbations applied to inputs from the target domain. Concretely, we add to the objective function an additional TC loss term defined as follows:

$$\begin{aligned}\mathcal{L}_{\text{TC}} &= \mathcal{L}_{\text{VAT}} + \mathcal{L}_{\text{AUG}} \\ &= \mathbb{E}_{\mathbf{x}^t \sim p_t}\left[\max_{\|r\| \leq \epsilon} ||(f(\mathbf{x}^t) - f(\mathbf{x}^t + r)||^2\right] + \mathbb{E}_{\mathbf{x}^t \sim p_t}\left[||(f(\mathbf{x}^t) - f(\tilde{\mathbf{x}}^t)||^2\right]\end{aligned} \tag{7.4}$$

Similar to (Shu et al. 2018), the first term incorporates the locally-Lipschitz constraint by applying Virtual Adversarial Training (VAT) (Miyato et al. 2018) which forces the model to be consistent within the norm-ball neighborhood of each target sample $\mathbf{x}^t$. Additionally, the second term forces the model to embed a target instance $\mathbf{x}^t$ and its augmented version $\tilde{\mathbf{x}}^t$ similarly to push for smooth network responses in the vicinity of each target data. With a carefully chosen set of augmentations, such a constraint makes sense since the semantic content of a transformed image is approximately preserved. Note that for more stable training, we follow Mean Teachers (MT) (Tarvainen et al. 2017) and use of an exponential moving average of the model to compute the target pseudo-labels (*i.e.*, $f(\mathbf{x}^t)$). Overall, $\mathcal{L}_{\text{TC}}$ is in line with the cluster assumption since it promotes a consistency of predictions over a various set of input perturbations, thus forcing the decision boundary to not cross the high-density regions.

To illustrate the effect of TC on the decision boundary, we conduct a toy experiment on the rotating two moons dataset. The target samples are obtained by rotating the source points by $45°$. We compare the learned decision boundary when training with source CE only, with a source CE

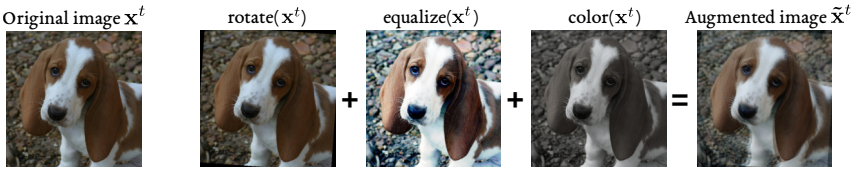


Figure 7.3: Mixing Augmentations. An example of an augmented image with $K = 3$. We start by sampling 3 operations, rotate, equalize, and color, which are then applied to the original image. The augmented image can then be obtained using an element-wise convex combination of the augmented images, resulting in a semantically similar image while injecting a higher degree of noise.

& DANN objective, and when adding a TC (*i.e.*, using only Gaussian noise) term to both. As shown in Fig. 7.2, the TC term helps push the decision boundary away from dense target regions, resulting in a more optimal target classifier.

Augmentations. For visual domain adaptation, and based on the recent success of supervised image augmentations (Cubuk et al. 2019a,b; Lim et al. 2019) in semi-supervised learning (Sohn et al. 2020; Xie et al. 2019) and robust deep learning (Hendrycks et al. 2020; Yin et al. 2019), we propose to use a rich set of state-of-the-art data augmentations to inject noise and enforce consistency of predictions on the target domain. Specifically, we use augmentations from AutoAugment (Cubuk et al. 2019a). Upon each application, we sample a given augmentation aug from all possible augmentations $\mathrm{Augs} = \{\text{equalize}, \ldots, \text{brightness}\}$. If the augmentation aug is applicable with varying severities, we also uniformly sample the severity, and apply aug to obtain the augmented target image, *i.e.*, $\tilde{\mathbf{x}}^t = \mathrm{aug}(\mathbf{x}^t)$. However, applying a single operation might be solved easily with a high-capacity model by memorizing the specific perturbations. To overcome this, we generate more diverse augmentations by mixing multiple augmented images (see Fig. 7.3). We start by randomly sampling K operations from Augs and K convex coefficients α_i sampled from a Dirichlet distribution, *i.e.*, $(\alpha_1, \ldots, \alpha_K) \sim \mathrm{Dir}(1, \ldots, 1)$. The augmented image $\tilde{\mathbf{x}}^t$ can then be obtained with an element-wise convex combination of the K augmented instances of $\mathbf{x}^t$, *i.e.*, $\tilde{\mathbf{x}}^t = \sum_{i=1}^{K} \alpha_i \, \mathrm{aug}_i(\mathbf{x}^t)$, impelling the model to be stable, consistent, and insensitive across a more diverse range of inputs (Hendrycks et al. 2020; Kannan et al. 2018; Zheng et al. 2016).

7.4.2 Extensions

Effects of Target Consistency. Enforcing the target consistency gives us the ability to control the trade-off between a low target sensitivity, *i.e.*, a low violation of the cluster assumption on target, and a low source risk. As described in (Shu et al. 2018), adding $\mathcal{L}_{\mathrm{TC}}$ to the objective function reduces the hypothesis class $\mathcal{H}$ to only include classifiers that are robust on both target and source domains, noted $\mathcal{H}_{\mathrm{TC}}$. Through the lenses of domain adaptation theory (*i.e.*, Eq. (7.2)), by constraining the hypothesis space $\mathcal{H}$ to contain stable classifiers across domains, small changes to the hypothesis in the source domain will not induce large changes in the target domain (Shu et al. 2018), which reduces the domain discrepancy $d_{\mathcal{H}_{\mathrm{TC}}\Delta\mathcal{H}_{\mathrm{TC}}} \leq d_{\mathcal{H}\Delta\mathcal{H}}$ given that $\mathcal{H}_{\mathrm{TC}} \subset \mathcal{H}$.

However, viewing the effect of TC as a constraint on the hypothesis space does not explain the hidden interactions between TC and invariant representations. To this purpose, consider a

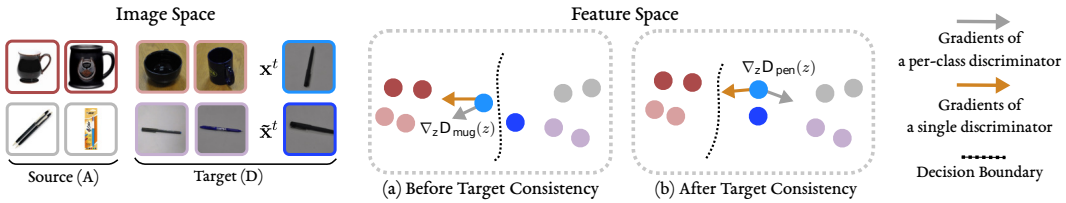


Figure 7.4: TC AND DOMAIN DISCRIMINATORS. We illustrate the effect of TC on the learned representations for two cases, i) with the standard DANN loss, and ii) with the proposed per-class domain discriminator. Mugs and pens from the source (Amazon, A) and target (DSLR, D) domains of Office-31 are pictured. At first (a), the blue squared pen, a target sample, is incorrectly classified as a mug due to spurious correlations *e.g.*, upward orientation and black color. Input augmentations wipe out spurious correlations induced by the orientation, and the TC pushes the decision boundary to low-density regions, correcting the predicted class (b). Before the TC update, the per-class discriminator encourages the pen to reach the high-density region of the incorrect class, *i.e.*, the mug class. However, the TC update allows the sample to cross the decision boundary and change the discriminator used for feature alignment, which now correctly aligns the pen's features. Thus resulting in a positive feedback loop between TC and CLIV. Comparatively, the gradient of a vanilla domain discriminator (*i.e.*, DANN) interacts poorly with the TC update since it does leverage the effect of TC when aligning the features.

target sample $\mathbf{x}^t$ near the decision boundary which is hard to adapt. Thus, its augmented version $\tilde{\mathbf{x}}^t$, is also likely to have a different predicted class. By enforcing TC, the model embeds $\mathbf{x}^t$ and $\tilde{\mathbf{x}}^t$ similarly to push the decision boundary far from class boundaries incrementally. Such incremental change might result in correcting the predicted class label. However, the underlying representations remain approximately the same, and the discriminator feedback does not reflect the predicted labels change. Now, consider that domain invariance is achieved by leveraging one discriminator per a given class, *i.e.*, class-level invariance. The change of the predicted label due to the TC update will result in a switch of the discriminator used, subsequently reflecting the label change in the domain adversarial loss. This interaction between class-level invariance and decision boundary update is the key to the success of the proposed TC. See Fig. 7.4 for an illustration of such an interaction.

Per-class domain discriminators. Similar to (Pei et al. 2018) and (Cicek et al. 2019) that jointly align the input distributions and output classes for fine-grained alignment. We introduce CLIV, a well-suited Class-Level InVariance adversarial loss for our use case, which leverages one discriminator per given class. Let us redefine our domain discriminator d to be a set of C discriminators, *i.e.*, $d = (d_c)_{1 \leq c \leq C}$ for $\mathbf{z} \in \mathcal{Z}$, $d(\mathbf{z}) \in [0,1]^C$, and noting $\cdot$ as the scalar product in $\mathbb{R}^C$, CLIV is defined as follows:

$$\mathcal{L}_{\text{CLIV}} = \mathbb{E}_{(\mathbf{z}^s, y^s) \sim p_s}[y^s \cdot \log(d(\mathbf{z}^s))] + \mathbb{E}_{\mathbf{z}^t \sim p^t}[\hat{y}^t \cdot \log(1 - d(\mathbf{z}^t))] \tag{7.5}$$

where a given sample $\mathbf{x}$, with representation $\mathbf{z} = g(\mathbf{x})$ and prediction $\hat{y} = h(\mathbf{z})$, we weight the importance of discriminator d_c in the adversarial loss using the output $\hat{y}$. This action results in a class conditioning of the domain adversarial loss, where the ground truths are used in the source

domain and the predictions in the target domain. For a theoretical analysis of this loss, see (Bouvier et al. 2021).

Total Training Objective. To summarize, our model is trained by minimizing a trade-off between source CE, CLIV and TC, with λ_{CLIV} and λ_{TC} as tunable hyperparameters to control the contributions of each term:

$$\mathcal{L} = \mathcal{L}_{\text{CE}} + \lambda_{\text{CLIV}}\mathcal{L}_{\text{CLIV}} + \lambda_{\text{TC}}\mathcal{L}_{\text{TC}} \tag{7.6}$$

7.5 Experimental Results

We consider both image classification and segmentation tasks for the experiments in this chapter. First, we present details about the experimental setup we consider. We then offer a comprehensive ablation study and a set of qualitative and quantitative results of the proposed method.

7.5.1 Experimental Details

Network Architecture. For image classification, we use the same architecture as CDAN (M. Long et al. 2018) and adopt ResNet-50 (K. He et al. 2016) as a base network pre-trained on ImageNet dataset (Deng et al. 2009). For image segmentation, we follow ADVENT (Vu et al. 2019) and use Deeplab-V2 (L.-C. Chen et al. 2017a) as the base image segmentation architecture with a ResNet-101 backbone and a DCGAN as our domain discriminator (Radford et al. 2015).

Datasets. For image classification, we will conduct experiments on 4 UDA benchmarks, Office-31, ImageCLEF-DA, Office-Home and VisDA-2017. Office-31 (Saenko et al. 2010) is the standard dataset for visual domain adaptation, containing 4652 images depicting 31 categories divided across three domains: Amazon (A), Webcam (W), and DSLR (D). We use all six possible transfer tasks to evaluate our model. ImageCLEF-DA[2] is a dataset with 12 classes and 2400 images assembled from three public datasets: Caltech-256 (C), ImageNet (I), and Pascal VOC 2012 (P), where each one is considered as a separate domain. We evaluate all six possible pairs of the three domains. Office-Home (Venkateswara et al. 2017) is a more difficult dataset compared to Office-31, consisting of 1500 images across 65 categories in office and home settings. The dataset consists of four widely different domains: Artistic images (Ar), Clip Art (Ca), Product images (Pr), and Real-World images (Rw), see Fig. 3.4 for some examples. We conduct experiments on all twelve transfer tasks. VisDA-2017 (Peng et al. 2017) presents a challenging synthetic-to-real dataset with two very distinct domains. Synthetic, with renderings of 3D models with different lighting conditions and from many angles. And Real, containing real-world images. We conduct evaluations on the Synthetic $\rightarrow$ Real task. For image segmentation experiments, we evaluate our method on the challenging GTA5 $\rightarrow$ Cityscapes VisDA-2017 segmentation task with 19 categories. The synthetic source domain is GTA5 (Richter et al. 2016b) dataset with 24966 labeled images, while the real target domain is Cityscapes (Cordts et al. 2016b) dataset consisting of 5000 images.

[2] https://www.imageclef.org/2014/adaptation

	Ar→Cl	Cl→Ar	Pr→Ar	Pr→Cl	Rw→Cl	Avg
$\mathcal{L}_{\text{CLIV}}$	52.6	60.1	60.6	52.1	58.3	56.7
$+\mathcal{L}_{\text{VAT}}$	52.4	60.1	61.2	53.1	58.9	57.1
$+\mathcal{L}_{\text{AUG}}$	53.1	62.3	62.6	53.1	59.5	58.1
$+\mathcal{L}_{\text{VAT}} + \mathcal{L}_{\text{AUG}}$	53.0	**62.8**	62.8	53.8	**60.8**	58.6
$+\mathcal{L}_{\text{VAT}} + \mathcal{L}_{\text{AUG}}$ w/ MT	**53.1**	62.6	**63.8**	**54.4**	60.4	**58.9**

Table 7.1: TC loss ablations. We show the obtained Acc. (%) on the 5 hardest Office-Home tasks for different variations of the TC loss.

$\mathcal{L}_{\text{adv}} =$	$\mathcal{L}_{\text{DANN}}$	L_{CDAN}	$\mathcal{L}_{\text{CLIV}}$
$\mathcal{L}_{\text{adv}}$	47.6	53.4	56.7
$+\mathcal{L}_{\text{VAT}}$	48.0	55.1	57.1
$+\mathcal{L}_{\text{AUG}}$	51.3	55.7	58.1
$+\mathcal{L}_{\text{VAT}} + \mathcal{L}_{\text{AUG}}$	51.4	56.9	58.6
$+\mathcal{L}_{\text{VAT}} + \mathcal{L}_{\text{AUG}}$ w/ MT	51.0	56.0	**58.9**

Table 7.2: Adversarial loss ablations. We show the avg Acc. (%) obtained on the 5 hardest Office-Home tasks with the TC loss, but coupled with different types of adversarial losses. Mainly the standard DANN loss, CDAN loss (M. Long et al. 2018), and the proposed CLIV loss.

Evaluation Metrics. For image classification, we report either per-task accuracy or average accuracy over all tasks for compactness. As for image classification, we report the standard per-task mean Intersection-over-Union (mIoU) metric defined in Eq. (4.3).

Training details. For the training procedure, we follow the standard protocols for UDA (L.-C. Chen et al. 2017a; M. Long et al. 2017, 2018). We train on all labeled source samples and unlabeled target samples. In terms of hyperparameters, overall, we follow CDAN (M. Long et al. 2018) and ADVEN (Vu et al. 2019), and use their setup for all parameters not related to our proposed method for a fair comparison. As for method specific hyperparameters, *i.e.*, the loss weights λ_{CLIV} and λ_{TC} for both CLIV and TC and the number of mixed augmentations K. Specifically, we set $K = 4$, $\lambda_{\text{CLIV}} = 1$ and $\lambda_{\text{TC}} = 10$. We note that our method performs comparatively on a wide range of hyperparameter values, making it robust to hyperparameter selection and easier to deploy in practical applications.

Reproducibility. We employ PyTorch (Paszke et al. 2019) and base our implementation on the official and publicly available implementations of CDAN (M. Long et al. 2018) and ADVEN (Vu et al. 2019), and conduct all experiment on NVidia V100 GPUs.

7.5.2 Ablation Results

To examine the effect of each component of our proposed method, we conduct several ablations on the 5 most challenging tasks of Office-Home, with and without the TC term, and with different variations of the TC loss. The results are reported in Table 7.1. We observe that adding a consistency term, either VAT, or AUG, results in a higher accuracy across tasks, with better results when smoothing in the vicinity of each target data point within the data manifold with AUG in-

stead of the adversarial direction using VAT. Their combination with Mean Teacher (MT) results in an overall more performing model.

Most importantly, to show the importance of coupling TC with CLIV, we pair TC with DANN and CDAN losses. The obtained results in Table 7.2 show lower average accuracy and minimal gains when enforcing the cluster assumption in conjunction with such adversarial losses, confirming the importance of imposing class-level invariance when applying TC.

We also conduct an ablation study on the effect of varying the mixing number K to produce more diverse target images. Fig. 7.6 shows the results. Overall, we observe a slight improvement and more stable results when K is increased. Still, over a certain threshold, the degree of noise becomes significant, heavily modifying the semantic content of the inputs and hurting the model's performance.

DeepLab v2	
Method	mIoU
Adapt-SegMap (Tsai et al. 2018)	42.4
AdvEnt (Vu et al. 2019)	43.8
Ours	**44.9**
AdvEnt+MinEnt* (Vu et al. 2019)	**45.5**

Figure 7.5: Image Segmentation Results. We show the obtained mIoU on the GTA5 → Cityscapes adaptation task of the proposed approach. *AdvEnt+MinEnt** is an ensemble of two models.

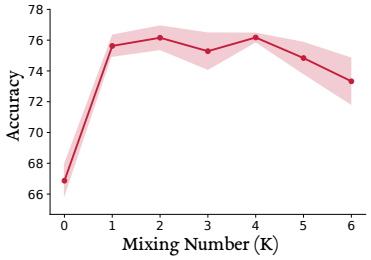


Figure 7.6: Mixing augmentations. We show the effect of the number of mixed augmentations K on the obtained accuracy on VisDA Synthetic → Real task.

7.5.3 Quantitative Results

Method	Office-31	ImageCLEF-DA	Office-Home	VisDA	VisDA (ResNet-101)
ResNet	76.1	80.7	46.1	45.6	52.4
DANN (Ganin et al. 2016)	82.2	85.0	57.6	55.0	57.4
CDAN (M. Long et al. 2018)	87.7	87.7	65.8	70.0	73.7
TAT (H. Liu et al. 2019)	88.4	88.9	65.8	71.9	-
BSP (X. Chen et al. 2019)	88.5	-	66.3	-	75.9
TransNorm (X. Wang et al. 2019)	89.3	88.5	67.6	71.4	-
Ours	**89.6**	**89.5**	**69.0**	**77.5**	**79.0**

Table 7.3: Classification Results. Average accuracy (%) of all tasks on image classification benchmarks for UDA. We compare our approach with similar methods based on invariant representations, evaluated using the same protocol. Results are obtained with a ResNet-50 unless specified otherwise.

For image classification results, we will present the results of the average accuracy obtained over all the tasks on each one of the 4 standard UDA classification benchmarks. These results are reported in Table 7.3. The proposed method outperforms previous adversarial methods on all datasets. The gains are substantial when the source and target domain are more dissimilar, as in

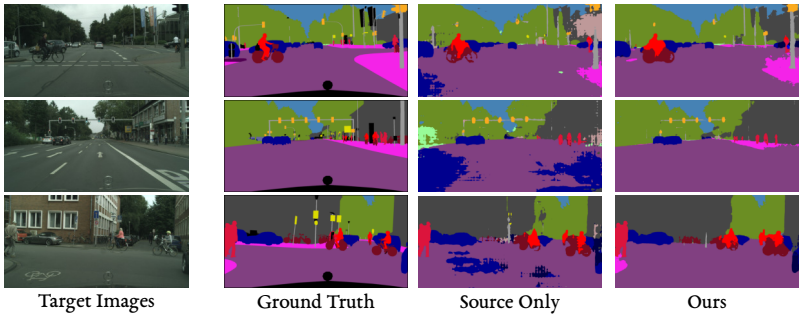
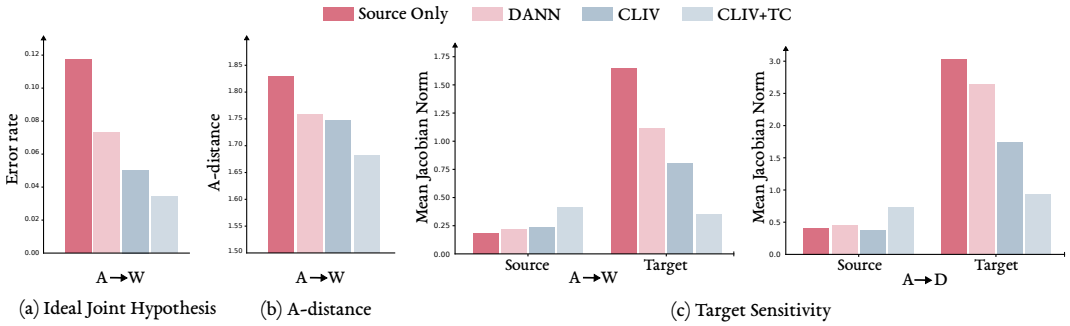


Figure 7.8: Segmentation results. We show segmentation results on the GTA5 → Cityscapes segmentation task.

VisDA dataset. We conjuncture that this is a result of a large number of target instances available, enabling us to extract a significant amount of training signal with TC objective term to enforce the cluster assumption. Additionally, the method performs well with a large number of categories, as is the case for Office-Home dataset. Such gain results from the class-level invariance, which is empowered as the number of classes grows. We observe overall smaller improvements on Office-31 due to its limited size and ImageCLEF-DA since the three domains are visually more similar. For image segmentation results and to demonstrate the generality of the proposed method, we conduct additional experiments on GTA5 → Cityscapes. The results are shown in Fig. 7.5, and we observe that the proposed method provides similar gains as the classification task, with a 2.5 points increase over the competitive baseline Adapt-SegMap (Tsai et al. 2018), further confirming the flexibility of TC and its applicability across various visual tasks.



(a) Ideal Joint Hypothesis (b) A-distance (c) Target Sensitivity

Figure 7.7: Qualitative analyses. We show (a) the error rate of the ideal joint hypothesis, (b) the A-distance as a measure of domain discrepancy, both on the A → W Office-31 task, and in (c) we show the effect of TC on the target and source sensitivity for A → W and A → D Office-31 tasks.

7.5.4 Qualitative Results

Ideal Joint Hypothesis. We evaluate the performances of the ideal joint hypothesis, which can be found by training a linear classifier on top of a frozen features extractor on the source and the

target data with the ground-truth labels on both domains. Fig. 7.7 (a) provides empirical evidence that TC produces a better joint hypothesis, thus more transferable representations.

Distributions Discrepancy. As proxy measure of domain discrepancy (Ben-David et al. 2010), we compute the A-distance, defined as $d_A = 2(1 - 2\varepsilon)$, with ε as the error rate of a domain classifier trained to discriminate source and target domains. Fig. 7.7 (b) shows that TC decreases d_A, implying a better invariance of the features under the proposed CLIV and TC training objective, confirming the positive feedback we have hoped for between these two losses.

Sensitivity Analysis. To revisit the sensitivity analysis of Section 7.3.2, we investigate the impact of TC on the model's sensitivity and compare the mean Jacobian norm of models trained with various objectives. As shown in Fig. 7.7 (c), TC coupled with CLIV greatly improves the model's robustness on target, with a small increase in source sensitivity.

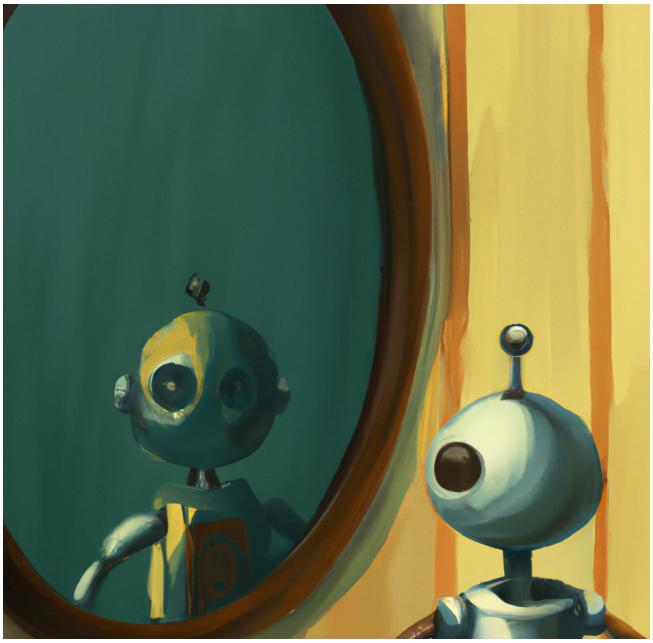
Segmentation Results. Fig. 7.8 shows some qualitative segmentation results of the proposed method. Specifically, we show the segmentation maps obtained with the baseline trained solely using the source CE, nd our proposed method, both on the GTA5 $\rightarrow$ Cityscapes adaptation task, The obtained segmentation maps show that the proposed approach help obtain segmentation results with high-quality, showing both locally consistent and globally coherent predictions.

Conclusion

In this chapter, we presented TC, a new approach to address the lack of target robustness under the domain invariant UDA setting by enforcing the cluster assumption in the target domain. Crucially, we show that TC interacts strongly with a per-class alignment of representations, substantially improving their transferability. Through extensive experiments, our approach outperforms other methods based on invariant representations, validating our analysis and our approach. Finally, TC has the advantage of being orthogonal to recent UDA works (X. Chen et al. 2019; H. Liu et al. 2019) for improving the transferability of invariant representations, making the idea of coupling TC with other UDA methods a research direction worth exploring.
In the next chapter, we will tackle a different learning paradigm, that of few-shot learning, and propose a novel method for the task of few-shot classification.

CHAPTER 8

Spatial-Contrastive Learning



8 Spatial Contrastive Learning

Chapter's Background

In this chapter, we consider:

- The Few-Shot Learning (FSL) paradigm (Section 3.5).
- The task of image classification (Sections 4.1.1 and 4.2.1).

The FSL setting deals with the learning problem under conditions with a minimal amount of labeled data. Thus, it perfectly complements the setting we tackled in previous chapters. It considers the rarity of the data as a whole, a case that was not considered under the UL setup, and the rarity of labeled data with no access to unlabeled data, which is a case that was not considered under the SSL and UDA setups. For the task, and since most of the popular FSL benchmarks are classification-based, we chose the image classification task to train and evaluate our method and compare it with established prior works. However, the proposed method was designed with the image segmentation task in mind and can be directly applied to a few-shot segmentation task with minimal changes.

Chapter's Summary

In this chapter, we investigate the usage of contrastive learning for few-shot classification. We propose to use it as an additional auxiliary training objective acting as a data-dependent regularizer to promote more general and transferable features. In particular, we present a novel attention-based spatial contrastive objective to learn locally discriminative and class-agnostic features. As a result, our approach overcomes some limitations of the cross-entropy loss, such as its excessive discrimination toward seen classes, which reduces the transferability of features to unseen classes. With extensive experiments, we show that the proposed method outperforms state-of-the-art approaches, confirming the importance of learning good and transferable embeddings for few-shot learning.

8.1 Introduction

FSL (Section 3.5) has emerged as an alternative to SL to simulate more realistic settings that mimic human capabilities, and in particular, it consists of reproducing the learner's ability to rapidly and efficiently adapt to novel tasks. In this chapter, we tackle the problem of few-shot image classification, which aims to equip a learner with the ability to learn novel visual concepts and recognize unseen classes with limited supervision.

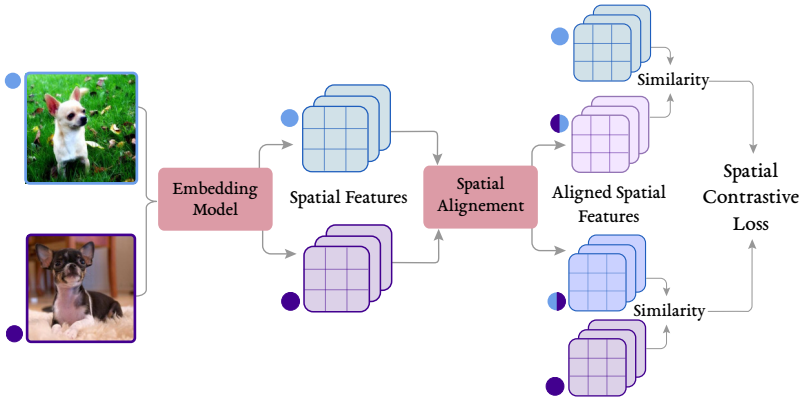


Figure 8.1: Spatial Contrastive Learning (SCL). To learn more locally class-independent discriminative features, we propose to measure the similarity between a given pair of samples using their spatial features as opposed to their global features. We first apply an attention-based alignment, aligning each input with respect to the other. Then, we measure the one-to-one spatial similarities and compute the Spatial Contrastive (SC) loss.

A popular paradigm to solve this problem is meta-learning (Naik et al. 1992; Thrun 1998) consisting of two disjoint stages, meta-training, and meta-testing. During meta-training, the goal is to acquire transferable knowledge from a set of tasks sampled from the meta-training tasks so that the learner can quickly adapt to novel tasks. This fast adaptability to unseen classes is evaluated at test time by the average test accuracy over several meta-testing tasks. Such transferable knowledge can be acquired from the meta-training tasks with optimization-based methods (C. Finn et al. 2017; Ravi et al. 2017) or metric-based methods (Snell et al. 2017; F. Sung et al. 2018; Vinyals et al. 2016).

Recently, a growing line of work (W.-Y. Chen et al. 2019; Dhillon et al. 2020; Tian et al. 2020a) showed that learning good representations results in fast adaptability at test time, suggesting that feature reuse (Raghu et al. 2019) plays a more important role in few-shot classification than the meta-learning aspect of existing algorithms. Such methods consider an extremely simple transfer learning baseline (see Fig. 3.7), in which the model is first pre-trained using the standard CE loss on the meta-training set. Then, at test time, a linear classifier is trained on the meta-testing set on top of the pre-trained model. The pre-trained model can either be fine-tuned (Afrasiyabi et al. 2020; Dhillon et al. 2020) together with the classifier, or fixed and used as a feature extractor (W.-Y. Chen et al. 2019; Tian et al. 2020a). While promising, we argue that using the CE loss during the pre-training stage hinders the quality of the learned representations since the model only acquires the necessary knowledge to solve the classification task over seen classes at train time. As a result, the learned visual features are excessively discriminative against the training classes, rendering them sub-optimal for test time classification tasks constructed from an arbitrary set of unseen and novel classes.

To alleviate these limitations, we propose to leverage contrastive representation learning (Section 3.3.2) as an auxiliary objective, where instead of only mapping the inputs to fixed targets, we also optimize the features, pulling together semantically similar (*i.e.*, positive) samples in the embedding space while pushing apart dissimilar (*i.e.*, negative) samples. By integrating the contrastive loss into the learning objective, we give rise to discriminative representations between dis-

similar instances while maintaining an invariance towards visual similarities. Subsequently, the learned representations are more transferable and capture more prevalent patterns outside of the seen classes. Additionally, by combining both losses, we leverage the stability of the CE loss and its effectiveness on small datasets and small batch sizes, while also leveraging the benefits of the contrastive loss as a data-dependent regularizer promoting more general-purpose embeddings.

Specifically, we propose a novel attention-based spatial contrastive loss (see Fig. 8.1) as the auxiliary objective to further promote class-agnostic visual features and avoid suppressing local discriminative patterns. It consists of measuring the local similarity between the spatial features of a given pair of samples after an attention-based spatial alignment mechanism instead of the global features (*i.e.*, avg., pooled spatial features) used in the standard contrastive loss. We also adopt the supervised formulation (P. Khosla et al. 2020) of the contrastive loss to leverage the provided label information when constructing the positive and negative samples.

However, directly optimizing the features and promoting the formation of clusters of similar instances in the embedding space might result in extremely disentangled representations. Such an outcome can be undesirable for few-shot learning, where the testing tasks can be notably different from the tasks encountered during training, *e.g.*, training on generic categories, and testing on fine-grained sub-categories. To solve this, we propose feature distillation to reduce the compactness of the features in the embedding space and provide additional refinement of the representations.

Chapter's Contributions

To summarize, this chapter's contributions are:

- We propose a novel Spatial Contrastive Learning (SCL) objective with an attention-based alignment mechanism to spatially compare a pair of features, further promoting class-independent discriminative patterns.
- We employ feature distillation to avoid excessive disentanglement of the learned embeddings and improve the performances.
- We demonstrate the effectiveness of the proposed method with extensive experiments on standard and cross-domain few-shot classification benchmarks, achieving state-of-the-art performances.
- We show the universality of the proposed method by applying it to a standard metric learning approach, resulting in a notable performance boost.

8.2 Related Work

Few-Shot Classification. In (inductive) few-shot classification, the objective is to learn to recognize unseen novel classes with few labeled examples in each class. Meta-learning remains the most popular paradigm to tackle this problem. As detailed in Section 3.5, such approaches can be divided into two categories. Optimization-based, or *learning to learn* methods (Andrychowicz et al. 2016; C. Finn et al. 2017; K. Lee et al. 2019; Ravi et al. 2017; Rusu et al. 2019; Q. Sun et al. 2019;

Y.-X. Wang et al. 2016) integrate the fine-tuning process in the meta-training algorithm to rapidly adapt to model to the unseen classes with limited supervision. Metric-based, or *learning to compare* methods (Doersch et al. 2020; Oreshkin et al. 2018; Scott et al. 2018; Snell et al. 2017; F. Sung et al. 2018; Vinyals et al. 2016), that learn a common embedding space in which the similarities between the data can help distinguish between different novel categories with a given distance metric. Most relevant to our work are the methods that follow the standard transfer learning strategy (Afrasiyabi et al. 2020; W.-Y. Chen et al. 2019; Dhillon et al. 2020; Tian et al. 2020a). They consist of two stages, a pre-training stage with the CE loss on the meta-training set, then a fine-tuning stage on the meta-testing set. Despite their apparent simplicity, Tian *et al*. (Tian et al. 2020a) showed that such a strategy yields state-of-the-art results on standard benchmarks.

Cross-Entropy Loss. The CE loss continues to be the prominent SL objective used for training deep networks, in which the model is trained to predict the corresponding class label in the form of a one-hot vector (Section 3.1). However, despite its success, some works showed many possible drawbacks (P. Khosla et al. 2020), *e.g*., noise sensitivity (Sukhbaatar et al. 2014; Z. Zhang et al. 2018), adversarial examples (Nar et al. 2019), and suboptimal margins (K. Cao et al. 2019; Elsayed et al. 2018). Other works proposed some alternative approaches, such as changing the label distribution (Müller et al. 2019; Szegedy et al. 2016; Yun et al. 2019; H. Zhang et al. 2018b) or leveraging the contrastive losses (P. Khosla et al. 2020).

Contrastive Learning. Instead of training the network to match the input to a fixed target, contrastive learning acts directly on the low-dimensional representations with contrastive losses (Gutmann et al. 2010; Hadsell et al. 2006; Salakhutdinov et al. 2007), and consist of measuring the similarities of different samples in the embedding space. Recently, contrastive learning-based methods have emerged as the state-of-the-art approaches for self-supervised representation learning (Section 3.3.2). The main difference between them is the way they construct and choose the positive samples. In this chapter, we differentiate between self-supervised contrastive methods (T. Chen et al. 2020; K. He et al. 2020; Henaff 2020; Hjelm et al. 2018b; A. v. d. Oord et al. 2018b; Tian et al. 2019; Z. Wu et al. 2018b) that leverage data augmentations to construct the positive pairs and supervised contrastive methods (Kamnitsas et al. 2018; P. Khosla et al. 2020; Salakhutdinov et al. 2007; Z. Wu et al. 2018a) that leverage the provided labels to sample the positive examples.

Self-Supervised Learning and Few-Shot Classification. Relevant to us in this chapter are methods that try to build on the insights and advances in contrastive learning, or more broadly self-supervised learning, to improve the few-shot classification task. Such methods (Doersch et al. 2015; Y. Gao et al. 2021; Gidaris et al. 2019; Medina et al. 2020; J.-C. Su et al. 2020) integrate various types of self-supervised training objectives into different few-shot learning frameworks to learn more transferable features and improve the few-shot classification performance. In this paper, we propose a novel contrastive learning objective based on the spatial features to further promote general purpose and robust representations suited for few-shot classification. In this context, a similar idea was proposed in (Doersch et al. 2015). In their approach, a contrastive pre-training stage is first conducted followed by the standard ProtoNet (Snell et al. 2017) fine-tuning stage, where spatial features are used to compute the similarity between the training and testing instances. In our proposed method, contrary to (Doersch et al. 2015), we integrate the spatial information directly into the contrastive learning loss. The proposed loss is then incorporated into the training as an

Figure 8.2: Analysis of the Learned Representations. (a) k-Nearest Neighbors Analysis. For a given test image from *mini*-ImageNet dataset, we compute the nearest neighbors in the embedding space on the test set, and we observe that they are semantically dissimilar. This suggests that the learned embeddings are excessively discriminative towards features used to solve the training classification tasks (*e.g.*, the beer bottles in the second test image), which are not useful for recognizing the novel classes at test time. (b) GradCAM results. To obtain the class activation maps (CAMs) explaining such an outcome, we train a linear classifier on the whole test set on top of the frozen embedding model and compute the CAMs. We see that the dominant discriminative features are not the ones useful for test-time classification.

auxiliary loss, resulting in a far more effective, flexible, and general framework usable in various few-shot learning scenarios.

8.3 Preliminaries

8.3.1 Problem Definition

Few-shot classification usually involves a meta-training set $\mathcal{I}$ and a meta-testing set $\mathcal{S}$ with disjoint label spaces. The meta-training set discerns *seen* classes, while the meta-testing set discerns novel and *unseen* classes. Each one of the meta sets consists of several classification tasks where each task describes a pair of training (*i.e.*, support) and testing (*i.e.*, query) sets with few examples, *i.e.*, $\mathcal{I} = \{(\mathcal{D}_t^{\text{tr}}, \mathcal{D}_t^{\text{test}})\}_{t=1}^{T}$ and $\mathcal{S} = \{(\mathcal{D}_q^{\text{tr}}, \mathcal{D}_q^{\text{test}})\}_{q=1}^{Q}$, with each dataset containing pairs of images $\mathbf{x}$ and their ground-truth labels y.

The goal of few-shot classification is to learn a classifier f_θ parametrized by θ capable of exploiting the few training examples provided by the dataset $\mathcal{D}^{\text{tr}}$ to correctly predict the labels of the test examples from $\mathcal{D}^{\text{test}}$ for a given task. However, given the high dimensionality of the inputs and the limited number of training examples, the classifier f_θ suffers from high variance. As such, the training and testing inputs are replaced with their corresponding features, which are produced by an embedding model f_ϕ parametrized by ϕ and then used as inputs to the classifier f_θ.

To this end, the objective of meta-training algorithms is to learn a good embedding model f_ϕ so that the average test error of the classifier f_θ is minimized. This usually involves two stages: first, a meta-training stage inferring the parameters ϕ of the embedding model using the meta-training set $\mathcal{I}$, followed by a meta-testing stage evaluating the embedding model's performance on the meta-testing set $\mathcal{S}$.

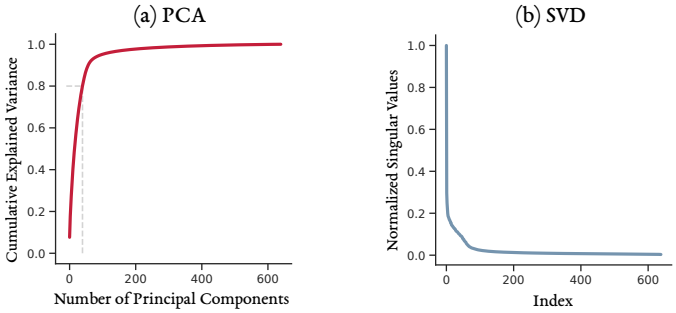


Figure 8.3: Spectral Analysis. Results of the spectral analysis on the embedding matrix. Plot (a) shows the explained cumulative variance of the learned features as the number of principal components used. We observe that 80% of the variance can be explained with only 30 components, indicating that embeddings lie in a lower dimensional space and are discriminative towards a small number of visual structures. Similarly, by computing the singular values of the embedding matrix, we see in (b) that the first singular values dominate the rest, indicating the same behavior.

8.3.2 Transfer Learning Baseline

In this chapter, we consider the simple transfer learning baseline of (Tian et al. 2020a), in which, as explained in Section 3.5, the embedding model f_ϕ is first pre-trained on the merged tasks from the meta-training set using the CE loss. Then, the model is carried over to the meta-testing stage and fixed during evaluation. Concretely, we start by merging all the meta-training tasks $\mathcal{D}_t^{\text{tr}}$ from $\mathcal{I}$ into a single training set $\mathcal{D}^{\text{new}}$ of seen classes, *i.e.*, $\mathcal{D}^{\text{new}} = \cup\{\mathcal{D}_1^{\text{tr}}, \dots, \mathcal{D}_t^{\text{tr}}, \dots, \mathcal{D}_T^{\text{tr}}\}$. Then, during the meta-training stage, the embedding model f_ϕ can be pre-trained on the resulting set of seen classes using the standard CE loss $\mathcal{L}_{\text{CE}}$:

$$\phi = \arg\min_{\phi} \mathcal{L}_{\text{CE}}(\mathcal{D}^{\text{new}}; \phi). \tag{8.1}$$

The pre-trained model f_ϕ is then fixed (*i.e.*, no fine-tuning is performed) and leveraged as a feature extractor during the meta-testing stage. For a given task $(\mathcal{D}_q^{\text{tr}}, \mathcal{D}_q^{\text{test}})$ sampled from $\mathcal{S}$, a linear classifier f_θ is first trained on top of the extracted features to recognize the unseen classes using the training dataset $\mathcal{D}_q^{\text{tr}}$:

$$\theta = \arg\min_{\theta} \mathcal{L}_{\text{CE}}(\mathcal{D}_q^{\text{tr}}; \theta, \phi) + \mathcal{R}(\theta), \tag{8.2}$$

where $\mathcal{R}$ is a regularization term and the parameters $\theta = \{\mathbf{W}, \mathbf{b}\}$ consist of weight and bias terms, respectively. The predictor f_θ can then be used on the features of the test dataset $\mathcal{D}_q^{\text{test}}$ to obtain the class predictions and evaluate f_ϕ.

8.3.3 Analysis of the Learned Representations

Although the baseline of Section 8.3.2 delivers impressive results, we hypothesize that the usage of the CE loss during the meta-training stage can hinder performances. Our intuition is that the learned representations lack general discriminative visual features since the CE loss induces em-

beddings tailored for solving the classification task over the seen classes. As a result, their transferability to novel domains with unseen classes is reduced, and especially if the domain gap between the training and testing stages is significant.

To empirically validate this hypothesis, we conduct a k-nearest neighbor search (J. Johnson et al. 2017) on the learned embedding space. First, we train a model with the CE loss on the meta-training set of *mini*-ImageNet (Vinyals et al. 2016) as in Eq. (8.1). Then, we search for its neighbors from the meta-testing set for a given test image. The results are shown in Fig. 8.2. For a fast test-time adaptation of the predictor f_θ, the desired outcome is to have visually and semantically similar images adjacent in the embedding space. However, we observe that the neighboring images are semantically dissimilar. Using Grad-CAM (Selvaraju et al. 2017), we notice that dominant discriminative features acquired during training might not be useful for discriminating between unseen classes at test time. In the case of *mini*-ImageNet, this observation is reinforced by the fact that the meta-training and meta-testing sets are closely related, in which better transferability of the learned features is expected when compared to other benchmarks. We note that similar behavior was also observed by (Doersch et al. 2015) for metric-learning based approaches.

We conduct a spectral analysis of the learned features to investigate this behavior further. As shown in Fig. 8.3, we inspect the variance explained by a varying number of principal components and notice that almost all of the variance can be captured with a limited number of components, indicating that the CE loss only preserves the minimal amount of information required to solve the classification task. Similarly, by applying singular value decomposition to compute the eigenvalues of the feature matrix, we observe that the maximal singular values are significantly larger than the remaining ones, diminishing the amount of informative signal that can be captured.

8.4 Method

8.4.1 Proposed Method

Contrastive Learning

We explore contrastive learning as an auxiliary pre-training objective. Specifically, we wish to use it to learn general-purpose visual embeddings capturing discriminative features usable outside of the meta-training set. Thus facilitating the test time recognition of unseen classes. In FSL, and given that we are provided with the class labels in such a setting, we examine the supervised formulation (P. Khosla et al. 2020) of the contrastive loss which leverages the label information to construct the positive and negative samples.

Formally, let f_ϕ be an embedding model mapping the inputs $\mathbf{x}$ to *spatial* features $\mathbf{z}^{\mathrm{s}} \in \mathbb{R}^{HW \times d}$, followed by an average pooling operation to obtain the *global* features $\mathbf{z}^{\mathrm{g}} \in \mathbb{R}^{d}$, which are then mapped into a lower dimensional space using a projection head h, *i.e.*, $\mathbf{f} = h(\mathbf{z}^{\mathrm{g}})$ with $\mathbf{f} \in \mathbb{R}^{d'}$. Additionally, let a global similarity function $\mathrm{sim}_{\mathrm{g}}$ be defined as the cosine similarity between a pair of projected global features $\mathbf{f}_i$ and $\mathbf{f}_j$ (*i.e.*, dot product between the L_2 normalized features). First, we sample a batch of N pairs of images and labels from the merged meta-training set $\mathcal{D}^{\mathrm{new}}$ and augment each example in the batch, resulting in $2N$ data points. Then, the supervised con-

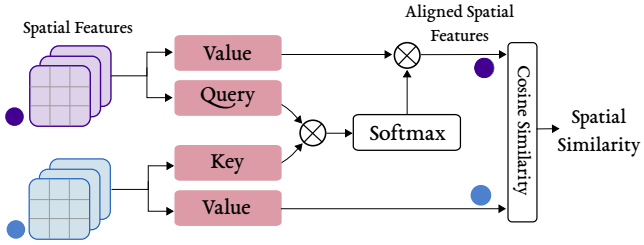


Figure 8.4: Attention-based Spatial Alignment. To compute the spatial similarity between a pair of features (purple and blue), we first spatially align the first features (purple) with respect to the second (blue) features with the attention mechanism (see Eq. (8.4)). Then we can compare the aligned value of the first features with the value of the second features. Note that the same process is applied in reverse to compute the final spatial similarity (see Eq. (8.5)).

trastive loss (P. Khosla et al. 2020), referred to as the Global Contrastive (GC) loss, can be computed as follows:

$$\mathcal{L}_{\text{GC}} = \sum_{i=1}^{2N} \frac{1}{2N_{y_i} - 1} \sum_{j=1}^{2N} \mathbb{1}_{i \neq j} \cdot \mathbb{1}_{y_i = y_j} \cdot \ell_{ij} \tag{8.3}$$

$$\text{where } \ell_{ij} = -\log \frac{\exp(\text{sim}_{\text{g}}(\mathbf{f}_i, \mathbf{f}_j)/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{i \neq k} \cdot \exp(\text{sim}_{\text{g}}(\mathbf{f}_i, \mathbf{f}_k)/\tau)}$$

with $\mathbb{1}_{\text{cond}} \in \{0, 1\}$ as an indicator function evaluating to 1 iff cond is satisfied, N_{y_i} as the total number of images with the same label y_i, and τ as a scalar temperature parameter. By using the GC loss of Eq. (8.3) as an additional pre-training objective with the CE loss, we push the embedding model f_ϕ to learn the visual similarities between instances of the same class. Instead of only maintaining the useful features for the classification task over the seen classes results in more useful and transferable embeddings.

Spatial Contrastive Learning

Although the GC loss is capable of producing good embeddings, using the global features $\mathbf{z}^{\text{g}}$ might suppress some local discriminative features present in the spatial features $\mathbf{z}^{\text{s}}$ that can be informative at the meta-testing stage (*e.g.*, suppressing object-specific features while overemphasizing the irrelevant background features). Additionally, encoding the relevant spatial information into the learned representations can play a critical role in increasing the robustness of the embeddings and reducing their sensitivity to domain changes, which is a highly desirable property for few-shot tasks. To this end, we propose a novel SC loss as an alternative objective, leveraging the spatial features $\mathbf{z}^{\text{s}}$ to compute the similarity between a given pair of examples. However, to locally compare a pair of spatial features $\mathbf{z}_i^{\text{s}}$ and $\mathbf{z}_j^{\text{s}}$ and compute the SC loss, we first need to define a mechanism to align them spatially. To this end, we employ the attention mechanism (Vaswani et al. 2017) to compute the spatial attention weights to align the features $\mathbf{z}_i^{\text{s}}$ with respect to $\mathbf{z}_j^{\text{s}}$ and vice-versa. Then, we measure the one-to-one spatial similarity and compute the SC loss. See Fig. 8.4 for an illustration of this process.

Attention-based Spatial Alignment. Let h_v, h_q, and h_k denote the value, query, and key projection heads, taking as input the spatial features $\mathbf{z}^{\mathrm{s}}$ and outputting the value $\mathbf{v}$, query $\mathbf{q}$ and key $\mathbf{k}$ of d'-dimensional features, *i.e.*, $\mathbf{v}, \mathbf{q}, \mathbf{k} \in \mathbb{R}^{HW \times d'}$. Given a pair of spatial features $\mathbf{z}_i^{\mathrm{s}}$ and $\mathbf{z}_j^{\mathrm{s}}$ of two instances i and j, we want to compute the aligned values of i with respect to j, denoted as $\mathbf{v}_{i|j}$. Such an alignment can be obtained using the key $\mathbf{k}_i$ and the query $\mathbf{q}_j$ to compute the attention weights $\mathbf{a}_{ij} \in \mathbb{R}^{HW \times HW}$, which can then be applied to $\mathbf{v}_i$ to obtain $\mathbf{v}_{i|j}$. Concretely, this can be computed as follows:

$$\mathbf{v}_{i|j} = \mathbf{a}_{ij}\mathbf{v}_i \;\text{ where }\; \mathbf{a}_{ij} = \mathrm{softmax}\left(\frac{\mathbf{q}_j \mathbf{k}_i^\top}{\sqrt{d'}}\right) \tag{8.4}$$

Similarly, we compute $\mathbf{v}_{j|i}$ aligning the value of j with respect to i using the key $\mathbf{k}_j$ and the query $\mathbf{q}_i$.

Time Complexity. The spatial alignment mechanism has a time complexity of $O(N^2H^2W^2d'^2)$, which varies with the batch size, the size of the spatial features, and the dimensionality of the values $\mathbf{v}$. To avoid excessive cost, for large input images, we apply an adaptive average pooling to reduce the size of the spatial features, in addition to using a small dimensionality d' and relatively small batches.

Spatial Similarity. Given a pair of values $\mathbf{v}_i$ and $\mathbf{v}_j$, together with their two aligned versions $\mathbf{v}_{i|j}$ and $\mathbf{v}_{j|i}$ computed using the attention mechanism detailed above, and with $\mathbf{v}_*^r$ denoting a feature vector at a spatial location $r \in [1, HW]$, we first perform an L_2 normalization step of the values $\mathbf{v}_*^r$ at each spatial location r. Then, we compute the total spatial similarity $\mathrm{sim}_{\mathrm{s}}(\mathbf{z}_i^{\mathrm{s}}, \mathbf{z}_j^{\mathrm{s}})$ between a pair of spatial features as follows:

$$\mathrm{sim}_{\mathrm{s}}(\mathbf{z}_i^{\mathrm{s}}, \mathbf{z}_j^{\mathrm{s}}) = \frac{1}{HW}\sum_{r=1}^{HW}\left[(\mathbf{v}_i^r)^\top \mathbf{v}_{j|i}^r + (\mathbf{v}_j^r)^\top \mathbf{v}_{i|j}^r\right] \tag{8.5}$$

Spatial Contrastive Learning. With the spatial similarity function $\mathrm{sim}_{\mathrm{s}}$ defined in Eq. (8.5), and similar to the GC loss in Eq. (8.3), the SC loss can be computed as follows:

$$\mathcal{L}_{\mathrm{SC}} = \sum_{i=1}^{2N}\frac{1}{2N_{y_i}-1}\sum_{j=1}^{2N}\mathbb{1}_{i\neq j}\cdot\mathbb{1}_{y_i=y_j}\cdot\ell_{ij} \tag{8.6}$$
$$\text{where }\; \ell_{ij} = -\log\frac{\exp(\mathrm{sim}_{\mathrm{s}}(\mathbf{z}_i^{\mathrm{s}}, \mathbf{z}_j^{\mathrm{s}})/\tau')}{\sum_{k=1}^{2N}\mathbb{1}_{i\neq k}\cdot\exp(\mathrm{sim}_{\mathrm{s}}(\mathbf{z}_i^{\mathrm{s}}, \mathbf{z}_k^{\mathrm{s}})/\tau')}$$

with τ' as a scalar temperature parameter.

Pre-training Objective

Based on the contrastive objectives in Eq. (8.3) and Eq. (8.6), the pre-training objective can take different forms. We mainly consider the case where the pre-training objective $\mathcal{L}_{\mathrm{T}}$ is the summation

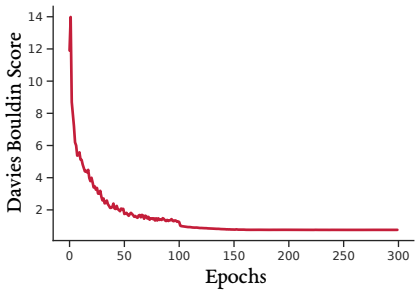


Figure 8.5: Degree of Clustering. The plot shows the evolution of the intra-class variation using the Davies-Bouldin index (Davies et al. 1979) during the course of training on *mini*-ImageNet when using the contrastive loss. We see that the learned embeddings of each class are significantly over-clustered. An outcome that might be undesirable in many few-shot classification scenarios.

of the CE and SC losses, with λ_{CE} and λ_{SC} as scaling weights to control the contribution of each term:

$$\mathcal{L}_{\text{T}} = \lambda_{\text{CE}}\mathcal{L}_{\text{CE}} + \lambda_{\text{SC}}\mathcal{L}_{\text{SC}} \tag{8.7}$$

However, we also explore other alternatives, such as replacing $\mathcal{L}_{\text{SC}}$ with $\mathcal{L}_{\text{GC}}$ or training with both $\mathcal{L}_{\text{GC}}$ and $\mathcal{L}_{\text{SC}}$ as auxiliary losses with their corresponding weighting terms. Additionally, we also consider the self-supervised formulations of the GC and SC losses, where the label information is discarded and the only positives considered are the augmented versions of each example (*i.e.*, $y_i = i \mod N$). We refer to them as SS-GC and SS-SC (Self-Supervised Global and Spatial Contrastive) losses, respectively.

Using the total loss $\mathcal{L}_{\text{T}}$, the embedding model f_ϕ can be trained together with the projection head and the attention modules during the meta-training stage. Specifically, let ψ represent the parameters of the projection head p and the attention modules h_v, h_q and h_k. The parameters are obtained as follows:

$$\{\phi, \psi\} = \arg\min_{\{\phi,\psi\}} \mathcal{L}_{\text{T}}(\mathcal{D}^{\text{new}}; \{\phi, \psi\}) \tag{8.8}$$

After the pre-training stage, the parameters ψ are discarded, and the embedding model f_ϕ is then fixed and carried over from the meta-training to the meta-testing stage.

8.4.2 Extensions

Since the contrastive objectives encourage closely aligned embeddings of instances of the same class while distributing all of the normalized features uniformly on the hypersphere (T. Wang et al. 2020), we have to consider a possible over-clustering of the features of the same class (see Fig. 8.5). Such an outcome can be desired for closed-set recognition. But in a few-shot setting, in which the discrepancy between the meta-training and meta-testing sets might differ significantly from one case to the other (*e.g.*, training on coarse seen categories and testing on fine-grained unseen sub-categories), and this might lead to sub-optimal performances. As such, to avoid an excessive disentanglement of the learned features and to further improve the generalization of the embed-

ding model, we propose Feature Distillation (FD) to reduce the compactness of the features in the embedding space.

Feature Distillation. Given a teacher model f_{ϕ_t} pre-trained with the objective in Eq. (8.7), we transfer its knowledge to a student model f_{ϕ_s} using the standard knowledge distillation (Hinton et al. 2015) objective $\mathcal{L}_{\text{KL}}$ (*i.e.*, the KL divergence between the student's predictions and the soft targets predicted by the teacher), but with an additional feature distillation loss $\mathcal{L}_{\text{CD}}$. This loss consists of maximizing the inner dot product between the L_2 normalized global features of the teacher $\mathbf{z}^{gt}$ and that of the student $\mathbf{z}^{gs}$, which corresponds to minimizing the squared Euclidean distance, formally:

$$\mathcal{L}_{\text{FD}} = \frac{1}{N}\sum_{i=1}^{N} \|\mathbf{z}_i^{\text{gt}} - \mathbf{z}_i^{\text{gs}}\|_2^2 \tag{8.9}$$

To summarize, the student's parameters are learned as follows:

$$\phi_s = \arg\min_{\phi_s} \lambda_{\text{CD}}\mathcal{L}_{\text{FD}}(\mathcal{D}^{\text{new}}; \phi_s, \phi_t) \; + \lambda_{\text{KL}} L_{\text{KL}}(\mathcal{D}^{\text{new}}; \phi_s, \phi_t) \tag{8.10}$$

As a result of using this distillation loss, we will only maximize the similarity between the pairs of features without using any negatives, thus relaxing the uniformity constraint of the contrastive loss, which in turn reduces the disentanglement of the learned embeddings.

8.5 Experimental Results

8.5.1 Experimental Details

Network Architecture. For the embedding model f_ϕ, we follow (Tian et al. 2020a) and use a ResNet-12 consisting of 4 residual blocks with Dropblock as a regularizer and 640-dimensional output features (*i.e.*, $d = 640$). For the projection head and the attention modules, we use an MLP with one hidden layer and a ReLU non-linearity similar to SimCLR, outputting 80-dimensional features (*i.e.*, $d' = 80$).

Training Setup. For optimization, we use SGD with a momentum of 0.9, a weight decay of 5×10^{-4}, a learning rate of 5×10^{-2}, and a batch size of 64. For the loss functions, we set the temperature parameters τ and τ' to 0.1 and the scaling weights λ_{CE}, λ_{SC}, and λ_{GC} to 1.0, except for CIFAR-FS where we set them to 0.5. For distillation, we set λ_{CD} to 10.0 and λ_{KL} to 1.0 and use a temperature of 4.0 for the KL loss. All the experiments are conducted on a V-100 GPUs. The implementation is available at `https://github.com/yassouali/SCL`.

Data Augmentation. During meta-training, for a given augmented batch of $2N$ examples, and consistent with other approaches (K. Lee et al. 2019; Tian et al. 2020a), the first N instances are obtained using standard augmentations, *i.e.*, random crop, color jittering and random horizontal flip. The remaining N instances are obtained with SimCLR-type augmentations, *i.e.*, random resized crop, color jittering, random horizontal flips and random grayscale conversion. During the meta-testing stage, we follow (Tian et al. 2020a) and create 5 augmented versions of each training image to overcome the problem of data insufficiency and train the linear classifier f_θ.

Loss Function	Aug.	*mini*-ImageNet, 5-way 1-shot	*mini*-ImageNet, 5-way 5-shot	CIFAR-CS, 5-way 1-shot	CIFAR-CS, 5-way 5-shot
CE		61.8 ± 0.7	79.7 ± 0.6	71.3 ± 0.9	86.1 ± 0.6
CE	✓	61.8 ± 0.8	78.6 ± 0.5	71.9 ± 0.9	86.3 ± 0.5
CE + SS-GC	✓	62.7 ± 0.7	81.0 ± 0.6	70.9 ± 0.9	84.5 ± 0.6
CE + SS-SC	✓	64.0 ± 0.8	81.5 ± 0.5	72.1 ± 0.8	86.2 ± 0.6
CE + SS-GC + SS-SC	✓	62.8 ± 0.8	81.1 ± 0.6	69.0 ± 0.9	85.0 ± 0.6
CE + GC	✓	65.0 ± 0.8	81.6 ± 0.5	74.0 ± 0.8	87.3 ± 0.6
CE + SC	✓	$\mathbf{65.7 \pm 0.8}$	$\mathbf{82.5 \pm 0.5}$	75.0 ± 0.9	87.4 ± 0.6
CE + GC + SC	✓	65.0 ± 0.8	81.3 ± 0.5	$\mathbf{76.0 \pm 0.7}$	$\mathbf{87.5 \pm 0.5}$

Table 8.1: Loss Function. Comparison of the mean Acc. obtained on *mini*-ImageNet and CIFAR-FS with different training objectives."Aug." indicates the usage of SimCLR type augmentations.

Evaluation Setup. During meta-testing, and given a pre-trained embedding model f_ϕ, we follow (Tian et al. 2020a) and consider a linear classifier as the predictor f_θ, implemented in scikit-learn (Pedregosa et al. 2011b) and trained on the L_2 normalized features produced by f_ϕ. Specifically, we sample a number of C-way K-shot testing classification tasks constructed from the unseen classes of the meta-testing set, with C as the number of classes and K as the number of training examples per class. After training f_θ on the train set, the predictor is then applied to the features of the test set to obtain the prediction and compute the accuracy. In our case, we evaluate the model over 600 randomly sampled tasks and report the median accuracy over 3 runs with 95% confidence intervals, wherein each run, the accuracy is the mean accuracy of the 600 sampled tasks.

8.5.2 Ablation Results

We start by conducting detailed ablation studies to analyze the contribution of each component of the proposed method, from the choices of the loss function to the hyperparameters of the SC loss.

Loss Functions. To investigate the effect of the contrastive losses when used as auxiliary training objectives, we evaluate the performances obtained with various loss functions as detailed in Section 8.4.1. The results are shown in Table 8.1. We observe a notable gain in performance when adopting auxiliary contrastive losses, be it supervised or self-supervised, with better improvements when using the supervised formulation, highlighting the benefits of using the label information when constructing the positive and negative samples. More importantly, the SC loss outperforms the standard GC loss, confirming the effectiveness of using spatial rather than global features. Additionally, using both the SC and GC losses does not result in distinct gains over the SC loss. Thus, we adopt the SC as a sole auxiliary loss for the rest of this section.

Spatial Contrastive Loss. Next, we examine different variations and hyperparameters of the SC loss when used as an auxiliary objective along with the CE loss. In particular, we consider the following variations:

- *Hyperparameters.* To inspect the SC loss's hyperparameter stability, we conduct experiments with different batch sizes and temperature values. As can be seen in Fig. 8.6, by combining the CE and the SC losses, we leverage the stability of the CE and obtain consistent results across several

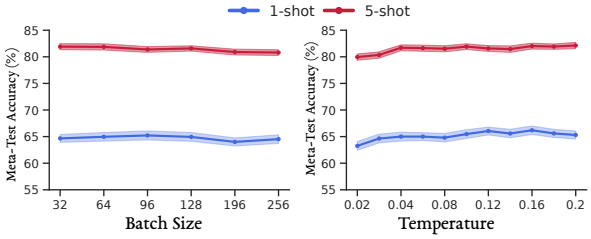


Figure 8.6: Hyperparameters. Comparison of the mean Acc. obtained on *mini*-ImageNet across various batch sizes and SC loss temperatures.

Augmentation	*mini*-ImageNet, 5-way		CIFAR-CS, 5-way	
	1-shot	5-shot	1-shot	5-shot
Standard	64.3 ± 0.7	80.6 ± 0.5	74.9 ± 0.8	86.3 ± 0.6
SimCLR	$\mathbf{65.7 \pm 0.8}$	$\mathbf{82.5 \pm 0.5}$	$\mathbf{75.0 \pm 0.9}$	87.4 ± 0.6
AutoAugment	65.2 ± 0.7	82.1 ± 0.5	74.0 ± 0.9	86.7 ± 0.6
Stacked RandAug.	64.9 ± 0.8	81.6 ± 0.6	$\mathbf{75.0 \pm 0.9}$	$\mathbf{87.6 \pm 0.6}$

Table 8.2: Data Augmentation. Comparison of the mean Acc. obtained on *mini*-ImageNet and CIFAR-FS with different augmentation strategies used to obtain the additional N augmented instances within a minibatch.

batch sizes, circumventing the need for very large batches when training with only the contrastive losses, as is the case in the unsupervised representation learning setting. For the temperatures, disregarding the low temperatures in which the SC loss is dominated by the small distances, rendering the actual distances between widely separated representations almost irrelevant, we see comparable performances for temperatures above 0.05, further confirming the stability of the approach.

- *Augmentations.* Although we mainly use SimCLR-type augmentations to produce the additional N augmented examples within a given batch, other augmentations can also be used. Specifically, we consider the standard augmentations used when training with only the CE loss, AutoAugment (Cubuk et al. 2019c) and Stacked RandAugment (Tian et al. 2020b). Table 8.2 shows that the SimCLR type augmentations yield the best results overall. We speculate that for the standard augmentation, without any novel transformations that the model is forced to be invariant under the gains are minimal. As for strong augmentations (*i.e.*, AutoAugment and Stacked RandAugment), the augmented inputs might be substantially deformed, making the spatial alignment insufficient and reducing the effect of the SC loss.

- *Aggregation Function.* Table 8.3 presents the results obtained with various aggregation functions used to aggregate the one-to-one spatial similarities into an overall measure. We observe that when using the mean as the aggregate, we obtain overall better performances across the different datasets and settings.

Distillation. To improve the generalization of the embedding model, we investigate the effect of knowledge distillation by training a new (*i.e.*, student) model using a pre-trained (*i.e.*, teacher) network with various training objectives. Table 8.4 shows a clear performance gain with the pro-

Aggregation	*mini*-ImageNet, 5-way		CIFAR-CS, 5-way	
	1-shot	5-shot	1-shot	5-shot
Sum	65.2 ± 0.8	81.2 ± 0.5	$\mathbf{75.3 \pm 0.8}$	87.3 ± 0.5
Mean	$\mathbf{65.7 \pm 0.8}$	$\mathbf{82.5 \pm 0.5}$	75.0 ± 0.9	$\mathbf{87.4 \pm 0.6}$
Maximum	65.5 ± 0.7	82.0 ± 0.5	73.4 ± 0.8	86.4 ± 0.6
LogSumExp	64.8 ± 0.8	81.7 ± 0.6	74.2 ± 0.8	87.0 ± 0.6

Table 8.3: Aggregation Function. Comparison of the mean Acc. obtained on *mini*-ImageNet and CIFAR-FS with different aggregation functions used to amount the total similarity from the one-to-one spatial similarities.

Loss Function	*mini*-ImageNet, 5-way		CIFAR-CS, 5-way	
	1-shot	5-shot	1-shot	5-shot
Teacher	65.7 ± 0.8	82.5 ± 0.5	75.0 ± 0.9	87.4 ± 0.6
KL	66.0 ± 0.8	82.5 ± 0.5	75.9 ± 0.9	87.4 ± 0.6
KL+FD	$\mathbf{67.4 \pm 0.8}$	$\mathbf{82.7 \pm 0.5}$	$\mathbf{76.5 \pm 0.9}$	$\mathbf{87.6 \pm 0.6}$

Table 8.4: Distillation. Comparison of the mean Acc. obtained on *mini*-ImageNet and CIFAR-FS with different distillation objectives.

posed FD objective as an additional loss term, confirming the benefits of optimizing the learned features and relaxing the compactness of the embedding space.

Additionally, we explore sequential self-distillation similar to Born-again networks (Furlanello et al. 2018), where we consider the student model as the teacher and repeat the distillation process. As detailed in Fig. 8.7, we notice a clear drop in performances beyond a single distillation step. We suspect this might result from an over-disentanglement of the features induced by the FD loss. As such, for the rest of the paper, we only apply a single distillation step to refine the features further while preserving the learned structures.

Evaluation. Until now, we primarily trained a linear classifier on top of the global features during the meta-testing stage. Nonetheless, given that we explicitly optimize the spatial features during training, which increases their discriminability, we investigate their usage as inputs to the linear classifier. To this end, we compare the performance when training over the global, spatial, or both features, where we train two classifiers and aggregate their predictions. Table 8.5 shows the evaluation results. Overall, using the global features to train the linear classifier offers slightly better results than the spatial features. We suspect this might result from slight overfitting of the classifier given that the spatial features increase the number of parameters to be learned, negatively impacting the performances. However, when leveraging both the spatial and global features, we obtain better results confirming the usefulness of the spatial feature even during the meta-testing stage.

8.5.3 Quantitative Results

Few-shot Classification

For a comparison with state-of-the-art, and based on the results of the ablation studies, we fix the training objective as SC+CE during the meta-training stage and use both the spatial and global

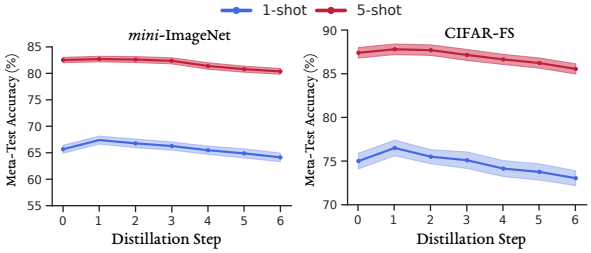


Figure 8.7: Sequential Distillation. Comparison of the mean Acc. obtained on *mini*-ImageNet and CIFAR-FS with sequential distillation.

Features Used	*mini*-ImageNet, 5-way		CIFAR-CS, 5-way	
	1-shot	5-shot	1-shot	5-shot
Spatial	64.5 ± 0.8	82.1 ± 0.5	$\mathbf{75.0 \pm 0.9}$	87.1 ± 0.6
Global	65.7 ± 0.8	82.5 ± 0.5	$\mathbf{75.0 \pm 0.9}$	87.4 ± 0.6
Glo. & Spa. (Max)	65.6 ± 0.8	82.1 ± 0.5	74.2 ± 0.8	87.3 ± 0.5
Glo. & Spa. (Sum)	$\mathbf{65.7 \pm 0.8}$	$\mathbf{83.1 \pm 0.5}$	$\mathbf{75.6 \pm 0.9}$	$\mathbf{87.6 \pm 0.6}$

Table 8.5: Features Used at Test-Time. Comparison of the mean Acc. obtained on *mini*-ImageNet and CIFAR-FS with different evaluation settings, in which we use either the global features, the spatial features, or both.

Method	Backbone	*mini*-ImageNet, 5-way		*tiered*-ImageNet, 5-way	
		1-shot	5-shot	1-shot	5-shot
MAML (C. Finn et al. 2017)	$32-32-32-32$	48.70 ± 1.84	63.11 ± 0.92	51.67 ± 1.81	70.30 ± 1.75
Matching Networks (Vinyals et al. 2016)	$64-64-64-64$	43.56 ± 0.84	55.31 ± 0.73	-	-
Prototypical Networks† (Snell et al. 2017)	$64-64-64-64$	49.42 ± 0.78	68.20 ± 0.66	53.31 ± 0.89	72.69 ± 0.74
Relation Networks (F. Sung et al. 2018)	$64-96-128-256$	50.44 ± 0.82	65.32 ± 0.70	54.48 ± 0.93	71.32 ± 0.78
SNAIL (N. Mishra et al. 2018)	ResNet-12	55.71 ± 0.99	68.88 ± 0.92	-	-
TADAM (Oreshkin et al. 2018)	ResNet-12	58.50 ± 0.30	76.70 ± 0.30	-	-
Shot-Free (Ravichandran et al. 2019)	ResNet-12	$59.04\pm$ n/a	$77.64\pm$ n/a	$63.52\pm$ n/a	$82.59\pm$ n/a
MetaOptNet (K. Lee et al. 2019)	ResNet-12	62.64 ± 0.61	78.63 ± 0.46	65.99 ± 0.72	81.56 ± 0.53
Diversity w/ Coop. (Dvornik et al. 2019)	ResNet-18	59.48 ± 0.65	75.62 ± 0.48	-	-
Boosting (Gidaris et al. 2019)	WRN$-28-10$	63.77 ± 0.45	80.70 ± 0.33	70.53 ± 0.51	84.98 ± 0.36
Fine-tuning (Dhillon et al. 2020)	WRN$-28-10$	57.73 ± 0.62	78.17 ± 0.49	66.58 ± 0.70	85.55 ± 0.48
LEO-trainval† (Rusu et al. 2019)	WRN$-28-10$	61.76 ± 0.08	77.59 ± 0.12	66.33 ± 0.05	81.44 ± 0.09
RFS (Tian et al. 2020a)	ResNet-12	62.02 ± 0.63	79.64 ± 0.44	69.74 ± 0.72	84.41 ± 0.55
RFS-Distill (Tian et al. 2020a)	ResNet-12	64.82 ± 0.60	82.14 ± 0.43	71.52 ± 0.69	86.03 ± 0.49
Ours	ResNet-12	65.69 ± 0.81	83.10 ± 0.52	71.48 ± 0.89	86.88 ± 0.53
Ours-Distill	ResNet-12	$\mathbf{67.40 \pm 0.76}$	$\mathbf{83.19 \pm 0.54}$	$\mathbf{71.98 \pm 0.91}$	$\mathbf{86.19 \pm 0.59}$

Table 8.6: State-of-the-art comparison on ImageNet derivatives. Comparison with prior few-shot classification works on ImageNet derivatives. We show the mean Acc. and 95% confidence interval. †results obtained by training on both train and validation sets.

feature during the meta-testing phase with a sum aggregate, and compare our approach with other popular few-shot classification methods on both ImageNet and CIFAR derivatives.

ImageNet derivatives. The *mini*-ImageNet benchmark is a standard dataset used for few-shot image classification. It consists of 100 randomly selected classes from ImageNet (Russakovsky et al. 2015). Following (Ravi et al. 2017), the classes are split into 64, 16, and 20 classes for meta-training, meta-validation, and meta-testing, respectively. Each class contains 600 images of size 84×84.

Method	Backbone	CIFAR-FS, 5-way		FC100, 5-way	
		1-shot	5-shot	1-shot	5-shot
MAML (C. Finn et al. 2017)	$32-32-32-32$	58.9 ± 1.9	71.5 ± 1.0	-	-
Relation Networks (F. Sung et al. 2018)	$64-96-128-256$	55.0 ± 1.0	69.3 ± 0.8	-	-
R2D2 (Bertinetto et al. 2019)	$96-192-384-512$	65.3 ± 0.2	79.4 ± 0.1	-	-
TADAM (Oreshkin et al. 2018)	ResNet-12	-	-	40.1 ± 0.4	56.1 ± 0.4
Shot-Free (Ravichandran et al. 2019)	ResNet-12	$69.2\pm$ n/a	$84.7\pm$ n/a	-	-
TEWAM (Qiao et al. 2019)	ResNet-12	$70.4\pm$ n/a	$81.3\pm$ n/a	-	-
Prototypical Networks† (Snell et al. 2017)	ResNet-12	72.2 ± 0.7	83.5 ± 0.5	37.5 ± 0.6	52.5 ± 0.6
Boosting (Gidaris et al. 2019)	WRN-28-10	73.6 ± 0.3	86.0 ± 0.2	-	-
MetaOptNet (K. Lee et al. 2019)	ResNet-12	72.6 ± 0.7	84.3 ± 0.5	41.1 ± 0.6	55.5 ± 0.6
RFS (Tian et al. 2020a)	ResNet-12	71.5 ± 0.8	86.0 ± 0.5	42.6 ± 0.7	59.1 ± 0.6
RFS-Distill (Tian et al. 2020a)	ResNet-12	73.9 ± 0.8	86.9 ± 0.5	44.6 ± 0.7	60.9 ± 0.6
Ours	ResNet-12	75.6 ± 0.9	87.6 ± 0.6	44.4 ± 0.8	60.8 ± 0.8
Ours-Distill	ResNet-12	$\mathbf{76.5 \pm 0.9}$	$\mathbf{88.0 \pm 0.6}$	$\mathbf{44.8 \pm 0.7}$	$\mathbf{61.4 \pm 0.7}$

Table 8.7: State-of-the-art comparison on CIFAR derivatives. Comparison with prior few-shot classification works on CIFAR-10 derivatives. We show the mean Acc. and 95% confidence interval. †results obtained by training on both train and validation sets.

The *tiered*-ImageNet (Ren et al. 2018) benchmark presents a larger subset of ImageNet, with 608 classes and images of size 84×84 assembled into 34 super-categories. These are split into 20 categories for meta-training and 6 categories for both meta-validation and meta-testing, aiming to minimize the semantic similarity between the split.

CIFAR derivatives. CIFAR-CS (Bertinetto et al. 2019) and FC100 (Oreshkin et al. 2018) are both CIFAR-100 (Krizhevsky et al. 2010) derivatives, containing 100 classes and images of size 32×32. For CIFAR-CS, the classes are divided into 64, 16, and 20 classes for meta-training, meta-validation, and meta-testing, respectively. For FC100, the classes are grouped into 20 super-categories, split into 12 categories for meta-training and 4 for meta-validation and meta-testing.

Results. The results of 5-way classification are summarized in Table 8.6 and Table 8.7 for ImageNet and CIFAR derivatives respectively. Our method outperforms previous works and achieves state-of-the-art performances across different datasets and evaluation settings. This suggests that our attention-based SCL approach, coupled with the CE loss, improves the transferability of the learned embeddings without any meta-learning techniques. With additional improvements using a feature distillation step. These results also show the potential of integrating contrastive losses as auxiliary objectives for various few-shot learning scenarios.

8.5.4 Cross-Domain Few-shot Classification

To further affirm the improved transferability of the learned embedding with our approach, we explore the effects of an increased domain difference between the seen and unseen classes, *i.e.*, the discrepancy between the meta-training and meta-testing stages. Precisely, we follow the same procedure as (Tseng et al. 2020) where we first train on the whole *mini*-ImageNet dataset using the same setting as detailed above. Then, we evaluate the embeddings model on four different domains: CUB (Welinder et al. 2010), Cars (Krause et al. 2013), Places (B. Zhou et al. 2017), and Plantae (Van Horn et al. 2018). We show the obtained results in Table 8.8, and see a notable gain in perfor-

Method	CUB, 5-way		Cars, 5-way		Places, 5-way		Plantae, 5-way	
	1-shot	5-shot	1-shot	5-shot	1-shot	5-shot	1-shot	5-shot
MatchingNet (Vinyals et al. 2016)	35.89 ± 0.5	51.37 ± 0.7	30.77 ± 0.5	38.99 ± 0.6	49.86 ± 0.8	63.16 ± 0.8	32.70 ± 0.6	46.53 ± 0.6
MatchingNet w/ FT (Tseng et al. 2020)	36.61 ± 0.6	55.23 ± 0.8	29.82 ± 0.4	41.24 ± 0.6	51.07 ± 0.7	64.55 ± 0.7	34.48 ± 0.5	41.69 ± 0.6
RelationNet (F. Sung et al. 2018)	42.44 ± 0.7	57.77 ± 0.7	29.11 ± 0.6	37.33 ± 0.7	48.64 ± 0.8	63.32 ± 0.8	33.17 ± 0.6	44.00 ± 0.6
RelationNet w/ FT (Tseng et al. 2020)	44.07 ± 0.7	59.46 ± 0.7	28.63 ± 0.6	39.91 ± 0.7	50.68 ± 0.9	66.28 ± 0.7	33.14 ± 0.6	45.08 ± 0.6
GNN (Garcia et al. 2018)	45.69 ± 0.7	62.25 ± 0.6	31.79 ± 0.5	44.28 ± 0.6	53.10 ± 0.8	70.84 ± 0.6	35.60 ± 0.5	52.53 ± 0.6
GNN w/ FT (Tseng et al. 2020)	47.47 ± 0.6	66.98 ± 0.7	31.61 ± 0.5	44.90 ± 0.6	55.77 ± 0.8	73.94 ± 0.7	35.95 ± 0.5	53.85 ± 0.6
Ours	49.58 ± 0.7	67.64 ± 0.7	34.46 ± 0.6	$\mathbf{52.22 \pm 0.7}$	59.37 ± 0.7	76.46 ± 0.6	$\mathbf{40.23 \pm 0.6}$	59.38 ± 0.6
Ours-Distill	$\mathbf{50.09 \pm 0.7}$	$\mathbf{68.81 \pm 0.6}$	$\mathbf{34.93 \pm 0.6}$	51.72 ± 0.7	$\mathbf{60.32 \pm 0.8}$	$\mathbf{76.51 \pm 0.6}$	39.75 ± 0.8	$\mathbf{59.91 \pm 0.6}$

Table 8.8: State-of-the-art comparison on cross-domain few-shot benchmarks. Comparison with prior works on cross-domain few-shot classification benchmarks. We train the model on the *mini*-ImageNet domain and evaluate the trained model on other domains. We show the mean Acc. and 95% confidence interval.

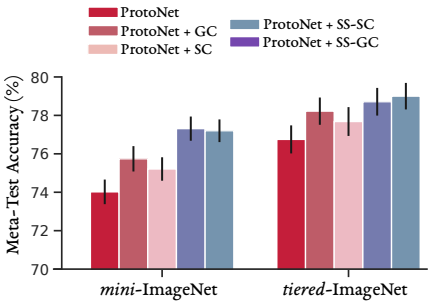


Figure 8.8: ProtoNet ablations. The obtained improvement when adding the contrastive objectives as auxiliary losses. We show the mean Acc. and 95% confidence interval for 5-way 5-shot classification across ImagetNet derivatives.

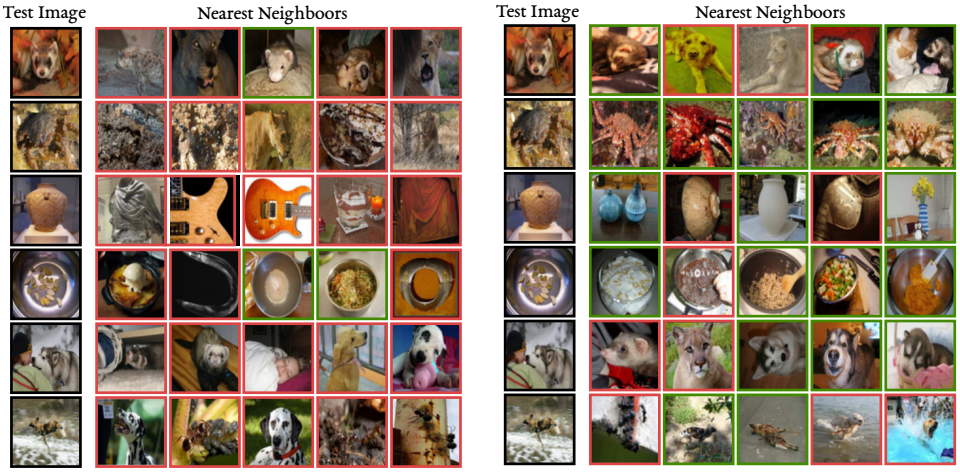
Method	Image Size	Backbone	Aux. Loss	Acc. (%)
ProtoNet (J.-C. Su et al. 2020)	224×224	ResNet-18	-	75.2
			Rotation	76.0
			Jigsaw	76.2
			Rot.+Jig.	76.6
Ours	224×224	ResNet-18	-	74.0
			GC	75.2
			SC	75.2
			SS-GC	77.3
			SS-SC	77.2
			SS-GC+SS-SC	**77.6**

Table 8.9: ProtoNet experiments on *mini*-ImageNet. Comparison with prior works on *mini*-ImageNet. We report the mean Acc. for 5-shot 5-way classification with implementation details including image size, backbone model, and auxiliary training losses for each method.

mance using the proposed method, from 2% gain on CUB dataset, up to 7% gain on Cars dataset, indicating a clear enhancement in terms of the generalization of the embedding model.

ProtoNet Experiments

To demonstrate the generality of the proposed approach and its applicability in different settings, in this section, we provide additional metric-learning based experiments in which we integrate the contrastive losses into the ProtoNet (Snell et al. 2017) framework. ProtoNet is a distance-based learner trained in an episodic manner, so that both the meta-training and meta-testing stages have matching conditions. During meta-training, for a C-way K-shot setting, we construct a meta-training set $\mathcal{I} = \{(\mathcal{D}_t^{\text{tr}}, \mathcal{D}_t^{\text{test}})\}_{t=1}^{T}$ where each given task $(\mathcal{D}_t^{\text{tr}}, \mathcal{D}_t^{\text{test}})$ depicts C randomly chosen classes from the seen classes, with K images per class for the training (*i.e.*, support) set $\mathcal{D}_t^{\text{tr}}$, and N images per classes for the test (*i.e.*, query) set $\mathcal{D}_t^{\text{test}}$. At each training iteration, after sampling a given task from $\mathcal{I}$, we first compute the class prototypes for classification using the support set. Then, the embeddings model is trained to minimize the CE loss where each query example is classified based on its distance with respect to the support's class prototypes. To add the con-



(a) CE (b) CE + SC

Figure 8.9: Nearest Neighbors Analysis. For a given test image from *mini*-ImageNet dataset, we compute 5 of its nearest test set neighbors in the embedding space for two cases, when a model is trained with either, (a) the standard CE, or (b) the proposed SC as an additional auxiliary objective. We observe that the neighboring images in the embedding space found when the SC loss is used are more semantically similar compared to the standard case with the CE loss. This suggests that the quality of the learned embeddings is increased with the usage of the SC as an auxiliary loss as a result of optimizing for more general-purpose features.

trastive objectives as auxiliary losses to the ProtoNet training objective, we simply merge the query and support set, augment each exampled within it, and compute the contrastive losses detailed in Section 8.4 over this merged and augmented set.

Experimental Details. For the experimentation, we follow (W.-Y. Chen et al. 2019) and base our implementation on their few-shot learning code base. In particular, we use a ResNet-18 network as the embedding model with 512-dimensional output features. We train on ImagetNet derivatives using ADAM optimizer with a learning rate of 10^{-3} for $60,000$ episodes and use 5-way (classes) 5-shot (examples per-class) with 16 query images. For contrastive learning, similar to transfer learning experiments, we use a two-layer MLP for the projection head and the attention modules with an output dimensionality of 64, and set λ_{CE} to 1.0, and λ_{GC} and λ_{SC} to 0.5. For meta-testing, we report the mean accuracy and 95% confidence interval over 600 randomly sampled tasks, where each class consists of 5 support images and 16 query images.

Results. To investigate the impact of the contrastive losses on the performances of ProtoNet, we report the obtained results for a 5-way 5-shot classification task on ImageNet derivatives with different training objectives. The results in Fig. 8.8 show a notable performance gain over the ProtoNet baseline. Surprisingly, when disregarding the labels and training with the self-supervised formulation of the contrastive objectives, we obtain better results. The SS-SC and SS-GC losses perform comparatively on *mini*-ImageNet with a 3.2% gain, and with the SS-SC loss performing slightly better on *tiered*-ImageNet with a 2.2% gain. We suspect that the obtained gain when using the self-supervised formulation might be a result of using a larger number of negatives as op-

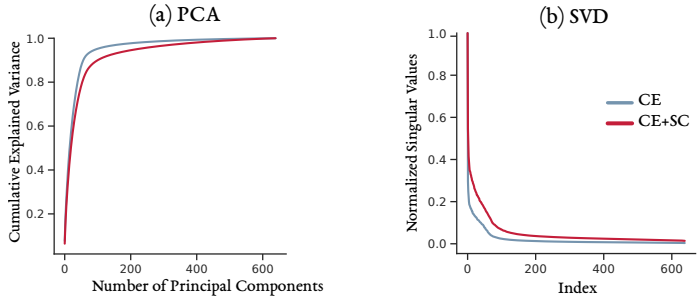


Figure 8.10: Spectral Analysis with SCL. Results of the spectral analysis on the embedding matrix using CE or CE+SC as training objectives. Plot (a) shows the explained cumulative variance of the learned features as the number of principal components used and (b) shows the max-normalized singular values. We observe that the SC loss increases the number of dominant principal components and the weight of the remaining singular values, indicating that the SC loss does indeed help retain more informative signals that might be useful outside the meta-training classification tasks.

posed to the supervised formulation, since each batch of examples only contains 5 unique classes. Additionally, we compare the performances of our approach with other self-supervised auxiliary losses, *i.e.*, rotation prediction (Gidaris et al. 2018c) and jigsaw puzzle solving (Noroozi et al. 2016), for which Su *et al*. (J.-C. Su et al. 2020) provided their integration into the ProtoNet framework. As shown in Table 8.9, we observe that a larger performance gain can be obtained with the contrastive objectives as auxiliary losses compared to other self-supervised objectives, especially when using both the SS-SC and SS-GC losses with a 3.6% gain over the baseline, which further confirms the effectiveness of the proposed SC loss.

8.5.5 Qualitative Results

In this section, we revisit the analysis conducted in Section 8.3.3 to qualitatively show the improvement of the learned representations. Specifically, we conduct an empirical analysis of the embeddings to assess the quality of the learned features in two cases: i) when the model is trained with only the CE loss, and ii) when adding the SC as an additional auxiliary objective. Figs. 8.9 and 8.10 show the results. We observe a clear improvement in terms of the obtained nearest neighbors when using the SC loss. We also see an increase in terms of the amount of informative signal retrained within the embedding matrix, indicating an enhancement in the quality of the learned embeddings.

Conclusion

In this chapter, we investigated contrastive losses as auxiliary training objectives along the CE loss to compensate for its drawbacks and learn richer and more transferable features. With extensive experiments, we showed that integrating contrastive learning into existing few-shot learning frameworks results in a notable boost in performance, especially with our spatial contrastive learning objective. Future work could investigate the spatial con-

trastive method extension for other few-shot learning scenarios and adapt it for different visual tasks, such as unsupervised representation learning.
In the next chapter, we will tackle a different learning paradigm, that of few-sample fine-tuning, building upon the pre-train, then fine-tune framework by proposing a novel fine-tuning method for the various NLP tasks.

Page intentionally left blank

CHAPTER 9

Relative Bias Fine-tuning

9 Relative Bias Fine-tuning

Chapter's Background

In this chapter, we consider:

- The Few-Sample Fine-Tuning (FSFT) setting (Section 3.1.2),
- Natural Language Processing (NLP) tasks (Section 4.1.3).

Up until now, all the tasks we considered were visual tasks. However, the textual modality is also an important component of many real-world DL-based systems (*e.g.*, the examples in Chapter 1). We thus choose to consider it in this chapter and set to solve various popular NLP downstream tasks. For the learning paradigm, we consider the label-efficient version of the fine-tuning stage, *i.e.*, the FSFT setting. As explained in the introduction, this paradigm is currently very popular for both its simplicity and practicality: large pre-trained language models (LMs), which are becoming readily available, can fine-tuned to solve the desired downstream NLP task. Specifically, in this chapter, we start from a pre-trained LM and set to introduce a label-efficient fine-tuning method to adapt the model to a given downstream task of interest.

Chapter's Summary

In this chapter, we consider the pre-train, then fine-tune paradigm based on large pre-trained LMs. In recent years, fine-tuning large pre-trained models has demonstrated its effectiveness in many NLP downstream tasks. However, the standard supervised fine-tuning approach suffers many drawbacks and limitations. First, when provided with few labeled samples, the fine-tuning process becomes brittle and prone to instability, over-fitting, and representation collapse, leading to sub-optimal performances. Additionally, fine-tuning is extremely parameter and compute-inefficient, requiring an entire model to be fine-tuned for every task. To solve this, we introduce RelBitFit, a method that builds on the recently introduced BitFit (Zaken et al. 2021). Instead of only fine-tuning the model's biases as in BitFit, the proposed RelBitFit consists of: i) first, introducing relative biases into the attention operation as new parameters, and ii) then, sparse fine-tuning of all bias terms of the model, *i.e.*, the newly introduced relative biases, and the model's pre-trained biases. With this simple and efficient approach, we demonstrate competitive performances with limited labeled data on the popular GLUE benchmark (A. Wang et al. 2018) consisting of a diverse set of NLP tasks.

9.1 Introduction

After the introduction of the transformer architecture (Section 4.2.3), large pre-trained LMs have become the go-to approach for solving most NLP tasks, thanks mainly to the pre-train then fine-tune strategy. This strategy takes advantage of the large quantities of unlabeled textual data available to train large LMs in an unsupervised manner. These pre-trained models are then fine-tuned on the downstream tasks of interest, resulting in both a reduction of the amount of target labeled data necessary and in a better performing model.

However, while this transfer learning-based approach is highly effective and have produced many state-of-the-art results (Brown et al. 2020; Devlin et al. 2019; Y. Liu et al. 2019; Radford et al. 2019), it suffers from some critical drawbacks. First, the fine-tuning process can often be volatile and easily disposed to over-fitting and representation collapse (Aghajanyan et al. 2021a; Mosbach et al. 2020), especially on small datasets (T. Zhang et al. 2021). As a consequence, the fine-tuned models are prone to spurious biases in the training data (Niven et al. 2019), limiting their generalization to unseen examples. Additionally, given the size of such large LMs and the resources required for fine-tuning and storing them, this approach becomes infeasible and impractical as the number of downstream tasks increases

This chapter focuses on Parameter-Efficient Fine-Tuning (PEFT) approaches, which were first popularized by (Houlsby et al. 2019). Instead of updating all the parameters of the pre-trained LM where we optimize the model to *learn* and *acquire* new task-related knowledge, PEFT approaches argue that large pre-trained LMs already contain all the desired knowledge, and at the fine-tuning stage, the goal should be limited to simply *extracting* and *exposing* the knowledge to solve the downstream task (Zaken et al. 2021). To this end, PEFT approaches consist of only updating few parameters for new tasks, while the majority of the model's weights remain unchanged, *i.e.*, taking their pre-trained values. The updated parameters can either be: i) newly introduced weights and modules, such as adapter (Houlsby et al. 2019) or hypernetwork (Mahabadi et al. 2021) modules or low-rank weights (E. J. Hu et al. 2022), or ii) can consist of fine-tuning a small subset of the pre-trained weights, *e.g.*, BitFit (Zaken et al. 2021) which consists of fine-tuning the biases only, which are less than 1% of the overall number of parameters. As a result, the fine-tuning process becomes more stable and less prone to over-fitting in low label regimes. Additionally, fine-tuning becomes computationally efficient and requires less storage to save a fine-tuned model for a given task since we only need to store the newly introduced modules or the fine-tuned weights instead of the whole model.

Given the effectiveness demonstrated by BitFit under the constraint of limited labeled data and given its computational efficiency, we build upon it and propose an improved version named RelBitFit (RELative BIas-Term FIne-Tuning). RelBitFit is based on a straightforward analysis where we show, using a simple expansion of the attention operation, that the key bias term does not contribute to the computation, thus rendering it unuseful and unutilized since it does not receive any gradients during fine-tuning. To solve this, we propose a reformulation of the attention computation, where we replace the absolute key bias-related terms with relative biases. As a result, all of the model's biases become trainable, thus adding to its flexibility during the fine-tuning stage. It also results in a more performing model, while retaining the computational efficiency of the original approach. With extensive experiments using the popular RoBERTa model (Y. Liu et al. 2019) on the diverse GLUE benchmark (A. Wang et al. 2018), we show that the proposed approach offers

better performances compared to the vanilla fine-tuning approach and other PEFT methods in limited labeled data regimes.

Chapter's Contributions

To summarize, this chapter's contributions are:

- A straightforward analysis of the attention operation thats a portion of the model's biases remains untrained, thus used during the fine-tuning stage.
- A new method, named RelBitFit, where we reformulate the attention operation by replacing the absolute key bias-related terms with relative biases, making them trainable and increasing the model's flexibility at the fine-tuning stage.
- Extensive experiments that show that RelBitFit, despite its simplicity and efficiency, outperforms previous methods on the popular GLUE benchmark.

9.2 Related Work

Pre-training, then Fine-tuning. With the advance of large-scale self-supervised pre-training of large language models (Devlin et al. 2019; Lewis et al. 2019; Y. Liu et al. 2019), the two-stage NLP pipeline of joint pre-training then task-specific fine-tuning has become standard practice. Such a process is often capable of delivering impressive performances by leveraging the knowledge acquired when a network, generally transformer-based (Vaswani et al. 2017), is first trained on raw text with self-supervision, then fine-tuned on the downstream task. The fine-tuning stage consists generally of adding a fully connected layer on top of the contextualized embeddings produced by the network, *e.g.*, embeddings of the `[CLS]` token for sentence-level classification. For more details, please refer to Sections 4.1.3 and 4.2.3. We follow a similar process in this work and use a pre-trained language model, *i.e.*, RoBERTa (Y. Liu et al. 2019), as a starting point for our few-sample fine-tuning approach.

The Limitations of Fine-tuning. Despite its inherent simplicity and competitiveness, the fine-tuning process still has some shortcomings. As pointed by (Aghajanyan et al. 2021a; Dodge et al. 2020; Mosbach et al. 2020), these large pre-trained LM models can suffer from instability caused by optimization difficulties, possible over-fitting to the training data limiting the model's transferability, and representation collapse. Moreover, fine-tuning such large models[1] to be used over a large set of diverse tasks requires an enormous amount of computing to adapt the model for every single task and large amount of disk storage to save the fine-tuned weights. It thus makes this strategy expensive, and renders it infeasible and impractical in many real-world scenarios.

Parameter-Efficient Fine-Tuning. To solve some limitations above, PEFT methods consist of updating or adding only few parameters per task instead of updating all the pre-trained parameters

[1] State-of-the-art models contain hundreds of billions of parameters. GPT-3 (Brown et al. 2020), for instance, is a 175B parameter model and requires more than 350GB of memory for storage.

of the model. This way, the fine-tuning process is constrained to extract and expose the task-specific knowledge already learned by the model to solve a given task. As a result, the optimization process becomes more stable, given that only a limited number of parameters is learned. The first PEFT method proposed to add adapters (Houlsby et al. 2019; Rebuffi et al. 2017), which are feed-forward networks injected into each layer of a given pre-trained network. Follow-up work investigated various efficient fine-tuning methods such as:

- Updating a sparse subset of the pre-trained parameters (Guo et al. 2020; Y.-L. Sung et al. 2021).
- Learning low-rank updates of the model's weights (E. J. Hu et al. 2022).
- Performing the fine-tuning in a low-dimensional space (Aghajanyan et al. 2021b).
- Replacing the adapter modules by hypernetworks (Mahabadi et al. 2021).
- Limiting the weight updates to only the bias terms of each layer of the pre-trained model, while the rest remains frozen, *i.e.*, BitFit (Zaken et al. 2021).

In this chapter, we propose an improved version of BitFit, named RelBitFit, where we introduce relative biases to replace key biased-based terms that are not learned during fine-tuning.

9.3 Preliminaries

BitFit: Bias-Term Fine-Tuning

To start, we introduce BitFit. With BitFit, the fine-tuning approach is simple and straightforward. For a given new task of interest, all the parameters of the pre-trained models are frozen, and only the bias terms are updated in addition to the output classification layer. This way, the fine-tuning stage becomes computationally efficient, all while matching or even surpassing the results obtained under the standard fine-tuning approach, where all parameters are updated.

Concretely, and as detailed in Section 4.2.3, each transformer block consists of three main components: attention, Layer Normalization (LN) (Xu et al. 2019), and the feed-forward network. Let $\mathbf{z} \in \mathbb{R}^{L \times d}$ be a set of d-dimensional input features to a given transformer block corresponding to an input sequence of L tokens. First, before updating the features via the attention operation, the features are mapped into a set of queries $\mathbf{q}$, keys $\mathbf{k}$, and values $\mathbf{v}$ using the corresponding feed-forward layers as follows[2]:

$$\begin{aligned} \mathbf{q} &= \mathbf{z}\mathbf{W}_q + \mathbf{b}_q \\ \mathbf{k} &= \mathbf{z}\mathbf{W}_k + \mathbf{b}_k \\ \mathbf{v} &= \mathbf{z}\mathbf{W}_v + \mathbf{b}_v \end{aligned} \tag{9.1}$$

where each feed-forward layer is parameterized by its weight matrix $\mathbf{W}_{(.)} \in \mathbb{R}^{d \times d}$ and its bias term $\mathbf{b}_{(.)} \in \mathbb{R}^{d}$. After the computation of the keys, queries, and values, the features $\mathbf{z}$ can then be updated via the self-attention operation as follows:

$$\begin{aligned} \text{Att}(\mathbf{q}, \mathbf{k}, \mathbf{v}) &= \text{softmax}\left(\frac{\mathbf{q}\mathbf{k}^\top}{\sqrt{d}}\right)\mathbf{v} \\ \mathbf{h}_1 &= \text{Att}(\mathbf{q}, \mathbf{k}, \mathbf{v}) \end{aligned} \tag{9.2}$$

[2]Note that for conciseness and clarity, we only present the computation used for a single attention head.

The updated features $\mathbf{h}_1$ are then fed into the output feed-forward layer, followed by a LN with a skip connection, which is parameterized by two weights, the gain $\mathbf{g}_{\text{LN}_{(.)}} \in \mathbb{R}^d$ and the bias $\mathbf{b}_{\text{LN}_{(.)}} \in \mathbb{R}^d$:

$$\begin{aligned} \mathbf{h}_2 &= \mathbf{h}_1 \mathbf{W}_o + \mathbf{b}_o \\ \mathbf{h}_3 &= \mathbf{g}_{\text{LN}_1} \odot \frac{(\mathbf{h}_2 + \mathbf{z}) - \mu}{\sigma} + \mathbf{b}_{\text{LN}_1} \end{aligned} \tag{9.3}$$

where $\odot$ is a dot product operation, and μ and σ are the per token mean and standard deviation of input $\mathbf{h}_2$.

Finally, the outputs $\mathbf{h}_3$ are passed through the feed-forward network, *i.e.*, the application of two feed-forward layers with weights $\mathbf{W}_{\text{FF}_1}$ and $\mathbf{W}_{\text{FF}_2}$ and biases $\mathbf{b}_{\text{FF}_1}$ and $\mathbf{b}_{\text{FF}_2}$, and with a GELU non-linearity (Hendrycks et al. 2016) in between, followed by an LN and skip connection as in Eq. (9.3), resulting in the final outputs $\mathbf{z}'$ of the transformer block, to then be fed into the next block:

$$\begin{aligned} \mathbf{h}_4 &= \text{GELU}(\mathbf{h}_3 \mathbf{W}_{\text{FF}_1} + \mathbf{b}_{\text{FF}_1}) \\ \mathbf{h}_5 &= \mathbf{h}_4 \mathbf{W}_{\text{FF}_2} + \mathbf{b}_{\text{FF}_2} \\ \mathbf{z}' &= \mathbf{g}_{\text{LN}_2} \odot \frac{(\mathbf{h}_5 + \mathbf{h}_3) - \mu}{\sigma} + \mathbf{b}_{\text{LN}_2} \end{aligned} \tag{9.4}$$

During the fine-tuning stage, and in addition to the output classification layer[3], all the original weights of the pre-trained model are frozen, and only the bias terms $\mathbf{b}_{(.)}$, indicated in red in Eqs. (9.1), (9.3) and (9.4), are fine-tuned. For a given model with M transformer blocks, and since the bias vectors are additive and are d-dimensional, except $\mathbf{b}_{\text{FF}_1}$ which is $4d$-dimensional[4], the total number of weights that is updated and that will need to be stored for a given task of interest is $11dM$. This number is small and negligible, consisting of less than 1% of total parameters. Thus, BitFit, in addition to being stable and performant in low-label settings, is extrelely efficient

Attention and the key bias $\mathbf{b}_k$

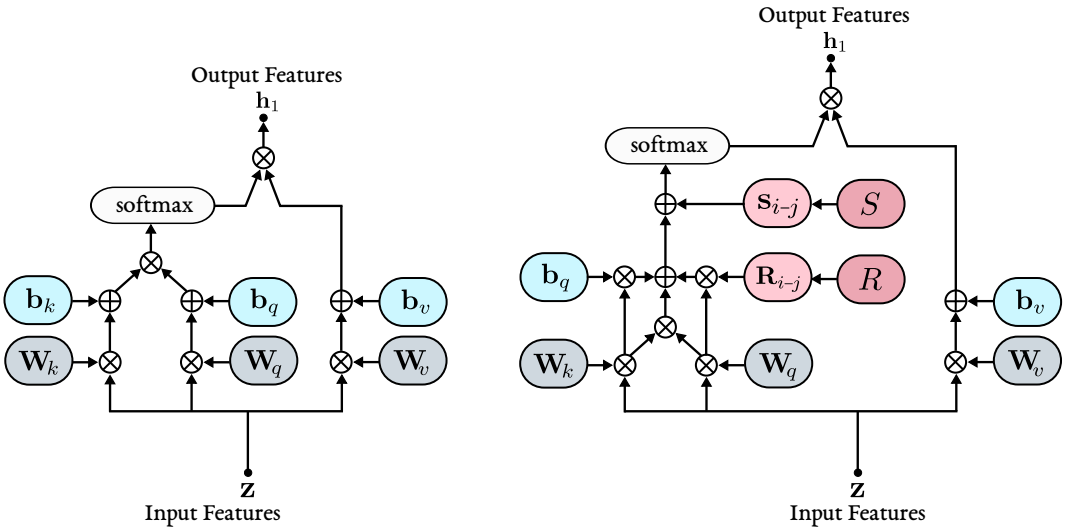
In this section, we investigate the attention operation described in Eq. (9.2) via a simple expansion, and show that the key bias term $\mathbf{b}_k$ does not contribute to the attention computation and is thus unused. This results in a reduction of the expressiveness and effectiveness of the BitFit approach.

While in the original implementation of attention (Vaswani et al. 2017), the biases were omitted, many of the following implementations, such as BERT (Devlin et al. 2019) and RoBERTa (Y. Liu et al. 2019), used an attention version with key and query bias terms, resulting in different queries-keys dot product. To investigate the contribution of these biases, we replace $\mathbf{q}$ and $\mathbf{k}$ in the dot product of Eq. (9.2) with their formulas in Eq. (9.1), obtaining the following expression:

$$\begin{aligned} \mathbf{q}\mathbf{k}^\top &= (\mathbf{z}\mathbf{W}_q + \mathbf{b}_q)(\mathbf{z}\mathbf{W}_k + \mathbf{b}_k)^\top \\ &= \underbrace{\mathbf{z}\mathbf{W}_q\mathbf{W}_k^\top\mathbf{z}^\top}_{1} + \underbrace{\mathbf{b}_q\mathbf{W}_k^\top\mathbf{z}^\top}_{2} + \underbrace{\mathbf{z}\mathbf{W}_q\mathbf{b}_k^\top}_{3} + \underbrace{\mathbf{b}_q\mathbf{b}_k^\top}_{4} \end{aligned} \tag{9.5}$$

[3] Which is a single weight matrix $\mathbf{W}_{\text{cls}} \in \mathbb{R}^{d \times C}$, mapping the representations of the sentence-level token `[CLS]` into logits over the C output classes.

[4] This is due to the expansion factor in the feed-forward network, which is generally set to 4 for transformer models.



(a) Standard Attention Computation

(b) Attention Computation with Relative Biases

Figure 9.1: The Attention Operation for BitFit and RelBitFit. We show the difference between (a) the standard attention operation, which is used in BitFit, and its proposed adjustment in (b). The version used in the proposed RelBitFit consists of replacing the key bias with two relative biases, content biased relative bias $\mathbf{R}_{i-j}$, and fixed relative biases $\mathbf{s}_{i-j}$.

After the decomposition of the attention dot product shown in Eq. (9.5), we obtain four terms. The first term (1) is the standard dot product similarity, comparing the content of each query with the content of the rest of the keys. The second term (2) is fixed addressing where the attention is computed based only on the content of keys. However, for the remaining two terms, (3) and (4), we find that for a given query at position i, the contribution of these two terms to its attention score with the rest of the keys at positions $j \in [1, L]$ is fixed. And since the $\mathrm{softmax}$ operation is shift invariant (*i.e.*, $\mathrm{softmax}(\mathbf{a} + \mathrm{const}) = \mathrm{softmax}(\mathbf{a}), \forall \mathrm{const}$), the key-bias related terms do not contribute to the attention weights and are thus not learned and can be disabled without any impact on the model's activations. Note that this behavior was also observed empirically in BitFit, and was also previously documented by (Cordonnier et al. 2020a). Nonetheless, in PEFT where each parameter must be carefully selected to be maximally useful during the fine-tuning stage, this simple analysis shows a large portion of the fine-tuning parameters in BitFit is not unused. To solve this, in the next section, we propose a simple reformulation of attention computation by replacing the two terms, (3) and (4), with relative biases (Z. Dai et al. 2019; Shaw et al. 2018).

9.4 Method

In this section, and based on the analysis presented in Section 9.3, we propose RelBitFit, an extension of BitFit. It consists of leveraging relative embeddings introduced in (Z. Dai et al. 2019; Shaw et al. 2018), and using them as relative biases for a plugin replacement of the two key-bias related terms in the attention computation (*i.e.*, terms (3) and (4) in Eq. (9.5)). In RelBitFit, instead of addressing all the key positions for a given query position with the same absolute value, which

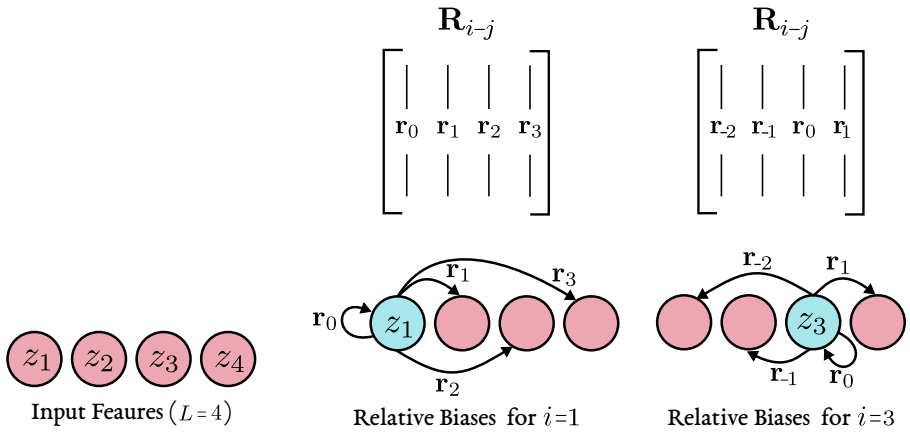


Figure 9.2: The Construction of The Content-based Relative Biases $\mathbf{R}_{i-j}$. We show an illustrative example describing how content-based relative biases are constructed. For an input sequence consisting of 4 tokens, *i.e.*, $L = 4$, our learnable matrix R will be of size $7 \times d$, consisting of all possible relative distances, from -3 up to 3. Then, during the attention operation, and for each query position i, we construct a set of relative biases $\mathbf{R}_{i-j}$, where each column $\mathbf{r}_{(.)}$ of $\mathbf{R}_{i-j}$ is fetched from the learnable matrix R.

renders the contribution of the key-bias terms futile, we propose relative biases that depend on the *relative* distance between position i of the query and position j of a given key. In Fig. 9.1, we show the standard version and the proposed modified version. Concretely, we propose to rewrite the dot product of the attention computation in Eq. (9.2) as follows:

$$\mathbf{q}\mathbf{k}^\top = \mathbf{z}\mathbf{W}_q\mathbf{W}_k^\top\mathbf{z}^\top + \mathbf{b}_q\mathbf{W}_k^\top\mathbf{z}^\top + \mathbf{z}\mathbf{W}_q\mathbf{R}_{i-j} + \mathbf{s}_{i-j} \tag{9.6}$$

with $\mathbf{R}_{i-j} \in \mathbb{R}^{d\times L}$ and $\mathbf{s}_{i-j} \in \mathbb{R}^{L}$ as the two newly introduced relative bias terms. These two terms are constructed from two learnable parameters, the content-based relative bias matrix $R \in \mathbb{R}^{d\times(2L-1)}$ and the fixed relative bias vector $S \in \mathbb{R}^{2L-1}$. The two learnable parameters have sizes of $2L-1$ in order to cover all possible relative distances $i-j$, from $-L$ up to L. Specifically, before computing the dot product in Eq. (9.6), we first construct $\mathbf{R}_{i-j}$ for each possible query position i by considering all possible relative distances to all the keys at positions $j \in [1, L]$. To construct this matrix at position i, we fetch L d-dimensional vectors from the columns of R, starting from index $k = \max(0, i-L)$, and up to index $l = \min(L, i+L)$, *i.e.*, using tensor-style slicing, we have $\mathbf{R}_{i-j} = R_{[:,k:l]}$. The same process is also applied to S to construct $\mathbf{s}_{i-j}$, *i.e.*, $\mathbf{s}_{i-j} = S_{[k:l]}$. For an illustration of this process, see Fig. 9.2.

To summarize, RelBitFit consists of two main points: 1) the introduction of two newly learnable parameters, R and S, which replace the key bias terms in BitFit and are trained from a random initialization for each new task of interest. 2) Similar to BitFit, all the rest of the biases (*i.e.*, the remaining 7 biases, see Section 9.3 for details), are updated together with the R and S, while the rest of the model's parameters are frozen. As such, RelBitFit maintains the efficiency of BitFit, all while having more flexibility and expressiveness due to the newly introduced relative biases.

An increased efficiency

While the current formulation of RelBitFit is more expressive, it suffers from an inflated number of trainable parameters. In BitFit, the total number of learnable parameters outside the output classification layer is $11dM$ for d-dimensional biases and with M transformer blocks. However, with RelBitFit, this number increases to $(10d + (2L-1)(d+1)) \times M$. This increase is due to the size of the learnable matrix R used to construct the relative biases $\mathbf{R}_{i-j}$. To solve this, we propose the following tricks to reduce its size:

- **Parameter Tying (PT)**. Instead of learning separate relative biases per-each layer, we propose to learn single joint parameters R and S for the whole model. This reduces the number of learnable parameters by a factor of L.

- **Low Rank (LR).** In order to reduce the size of R, we propose to decompose it into separate smaller matrices, *i.e.*, $R = R_1 R_2$. The first matrix $R_1 \in \mathbb{R}^{d \times d'}$, with $d' < d$, is shared between all layers, while the second matrix $R_2 \in \mathbb{R}^{d' \times (2L-1)}$ can either be shared (*i.e.*, parameter tying) or learned on a per-layer basis. At a given layer, the matrix R can be reconstructed using these two matrices. Then, the attention computation can be done as described in Eq. (9.6). This reduces the number of learnable parameters by almost a factor of d/d'.

- **Clipping the Relative Distances (Clip.).** Instead of learning a matrix R consisting of $(2L-1)$ d-dimensional columns taking into consideration all possible relative distance $i-j$, we follow (Shaw et al. 2018), and only consider $L_{\max} < (2L-1)$ possible relative distances, *i.e.*, $R \in \mathbb{R}^{d \times L_{\max}}$. Then, during the construction of the relative biases $\mathbf{R}_{i-j}$, we set its columns to either the minimal or maximal value if the relative distance is outside the range $L_{\max}$. This reduces the number of learnable parameters by a factor of $(2L-1)/L_{\max}$[5].

Depending on the downstream task of interest, and the available computational resources, one or all of the tricks mentioned above can be used to significantly reduce the number of learnable parameters required in the proposed RelBitFit. They all while maintaining acceptable performances as we demonstrate in the next experimental section.

9.5 Experimental Results

To evaluate the proposed fine-tuning method and investigate its effectiveness under the constraint of limited labeled data, we carry out various experiments on different NLP downstream tasks. First, in Section 9.5.1, we present the experimental details of RelBitFit. Then, in Section 9.5.2, we offer an ablation study to investigate the effectiveness of the parameter reduction tricks and the performance of RelBitFit under a variable number of labeled data. Finally, in Section 9.5.3, we compare our method to the standard Fine-tuning method, and with BitFit, and show performances above previous methods.

[5] Note that the same process is also applied over S, resulting in a smaller vector $S \in \mathbb{R}^{L_{\max}}$.

Dataset	Task	Domain	#Train	#Classes
RTE	Textual Entailment	News/Wikipedia	2.5k	2
MRPC	Paraphrase	News	3.7k	2
STS-B	Textual Similarity	News/Others	7k	Reg.
CoLA	Grammatical Correctness	Linguistic Publications	8.5k	2
SST-2	Sentiment Analysis	Movie Reviews	67k	2
QNLI	Question Answering/Textual Entailment	Wikipedia	105k	2
QQP	Question Answering/Semantic Equivalence	Quora	364k	2
MNLI	Textual Entailment	Multi-domain	393k	3

Table 9.1: GLUE Benchmark. Details about the various NLP downstream datasets present in the GLUE Benchmark that we consider in this chapter.

Dataset	Metric
QNLI	Accuracy
SST-2	Accuracy
MNLI	Matched Accuracy and Mismatched Accuracy
CoLA	Matthews Correlation
MRPC	F1
STS-B	Spearman Correlation
RTE	Accuracy
QQP	F1

Table 9.2: GLUE Evaluation Metrics. We list the metrics used for evaluation on the different datasets of the GLUE Benchmark. Note that for MNLI dataset, we have two evaluation sets, MNLI_{m} denoted MNLI Matched, MNLI_{mm} denoted MNLI Mismatched. We report the Accuracy on both of these sets.

9.5.1 Experimental Details

Network Architecture. In this chapter, we will use $\text{RoBERTa}_{\text{BASE}}$ (Y. Liu et al. 2019) as our pre-trained model, and then we fine-tune for a given downstream task following the proposed RelBitFit approach. $\text{RoBERTa}_{\text{BASE}}$ is an encoder-based transformer consisting of $M = 12$ transformer blocks, and with hidden size $d = 768$, and taking as input a sequence of length $L = 128$. For the implementation, we use HuggingFace (Wolf et al. 2020) open-source interface, and implement our RelBitFit method with their framework.

Datasets. For the NLP downstream datasets, we train and evaluate on the set of available datasets on the widely used GLUE benchmark (A. Wang et al. 2018). Consistent with previous work (Houlsby et al. 2019; Zaken et al. 2021), we use all the datasets except WNLI, on which BERT-like models do not perform well. Table 9.1 presents a brief summary of the GLUE dataset we will use in this chapter.

Note that since we are interested in scenarios with a low number of labeled examples, *i.e.*, FSFT, and since most of the GLUE datasets contain a relatively large number of examples, we opt to use a reduced version of them. Specifically, for a given dataset, we sample a reduced number N

Method	Num. of trainable params.	% of the model's total params.
Full Fine-tuning	$\approx 124 \times 10^6$	100%
BitFit	$\approx 0.7 \times 10^6$	0.55%
RelBitFit	$\approx 1.8 \times 10^6$	1.5%
RelBitFit with Clip. ($L_{\max} = 64$)	$\approx 1.2 \times 10^6$	1%
RelBitFit with Clip. ($L_{\max} = 32$)	$\approx 0.98 \times 10^6$	0.8%
RelBitFit with PT	$\approx 0.78 \times 10^6$	0.63%
RelBitFit with LR ($d' = 64$)	$\approx 0.76 \times 10^6$	0.8%
RelBitFit with PT+LR+Clip.($L_{\max} = 32$)	$\approx 0.73 \times 10^6$	0.59%

Table 9.3: Number of Trainable Paramters. We list the number of trainable parameters for the baselines and for different variations of the proposed RelBitFit. PT: Parameter Tying. Clip.: Clipping the Relative Distance. LR: Low Rank. Refer to Section 9.4 for details.

Method	QNLI	SST-2	MNLI_m	MNLI_mm	CoLA	MRPC	STS-B	RTE	QQP	Avg.
Size of training set: $N = 2000$										
No parm. red. trick	83.4	**91.7**	73.8	75.1	51.3	91.0	88.8	71.5	79.8	78.5
Clip. ($L_{\max} = 64$)	84.2	91.4	75.0	**75.9**	52.1	91.3	89.1	72.2	80.3	79.1
Clip. ($L_{\max} = 32$)	84.2	91.6	74.7	75.9	52.3	91.5	89.2	72.6	**80.6**	79.2
PT	84.1	**91.7**	74.4	75.4	50.8	91.0	88.8	71.1	80.0	78.6
LR ($d' = 64$)	**84.8**	91.6	**75.1**	76.4	52.9	**92.1**	89.1	**72.9**	80.0	**79.4**
All ($L_{\max} = 64$)	83.9	**91.7**	74.3	75.4	**53.2**	91.4	**89.3**	**72.9**	80.3	79.2
Size of training set: $N = 1000$										
No parm. red. trick	**82.7**	91.3	69.9	70.6	48.6	89.2	87.3	**66.8**	78.8	76.1
Clip. ($L_{\max} = 64$)	82.1	90.5	**71.6**	**72.1**	48.9	88.6	87.6	64.6	78.9	76.1
Clip. ($L_{\max} = 32$)	82.1	**91.5**	70.7	72.0	**49.1**	**90.5**	87.1	64.6	78.7	**76.3**
PT	81.7	90.9	69.1	69.6	48.7	88.5	**87.7**	65.3	**79.0**	75.6
LR ($d' = 64$)	82.6	90.5	69.2	70.0	47.7	89.3	87.3	63.9	78.6	75.5
All ($L_{\max} = 64$)	82.5	91.1	69.9	71.0	47.5	88.2	87.5	64.3	78.6	75.6

Table 9.4: Parameter Reduction. We show the obtained results over the different tasks and for different parameters reduction methods. We train all methods using either $N = 2000$ or $N = 1000$ labeled training examples. PT: Parameter Tying. Clip.: Clipping the Relative Distance. LR: Low Rank. All: PT+Clip.+LR. Refer to Section 9.4 for details.

of training examples from the original set while maintaining the same label distribution as the original set and use it as our new training.

Training Details. In order to adapt our pre-trained $\text{RoBERTa}_{\text{BASE}}$ model on the various tasks of the GLUE benchmark, we follow the standard practice, removing the final classification layer over the vocabulary (*i.e.*, used for pre-training) and attaching a new classification layer that produces C logits for a given input, with C as the desired number of output classes. Since all the GLUE tasks are sentence-level tasks, the classification layer takes as input the d-dimensional representation corresponding to the `[CLS]` token, which encodes the whole input sequence.

For optimization, we use AdamW (Loshchilov et al. 2018) optimizer, and use a fixed learning rate of $1e-4$ and a batch size of 8 for all experiments and for all datasets. For RelBitFit experiment, we ran a limited sweep for each dataset to find the best configuration of the method, mainly the task

Method	QNLI	SST-2	MNLI_m	MNLI_mm	CoLA	MRPC	STS-B	RTE	QQP	Avg.
Size of training set: $N = 2000$										
Full Fine-tuning	50.5	58.9	35.4	35.2	N/A	81.2	58.6	52.7	63.2	54.5
BitFit	83.5	91.6	74.0	75.4	51.1	91.1	88.9	71.8	80.4	78.6
RelBitFit	**84.8**	**91.7**	**75.1**	**76.4**	**55.7**	**92.1**	**89.4**	**74.4**	**80.6**	**80.0**
Size of training set: $N = 1000$										
Full Fine-tuning	50.5	50.9	35.4	35.2	N/A	81.2	83.0	52.7	66.9	57.0
BitFit	81.5	91.3	70.4	71.7	49.3	88.2	86.5	**66.8**	78.4	76.0
RelBitFit	**82.7**	**91.5**	**71.6**	**72.1**	**51.3**	**90.5**	**87.7**	**66.8**	**79.2**	**77.0**

Table 9.5: Comparaison with Relevant Works on Low-label Settings We show the obtained results over the different tasks and for different parameter reduction methods. We train all methods using either $N = 2000$ or $N = 1000$ labeled training examples. For CoLA task, the full fine-tuning was too unstable to report representative results.

Method	QNLI	SST-2	MNLI_m	MNLI_mm	Avg.
Size of training set: $N = 500$					
Full Fine-tuning	52.1	72.6	35.4	35.2	48.8
BitFit	78.4	89.7	63.9	64.5	74.1
RelBitFit	**80.7**	**90.8**	**64.1**	**64.6**	**75.0**
Size of training set: $N = 100$					
Full Fine-tuning	50.5	79.6	35.4	35.2	50.2
BitFit	56.4	84.3	35.3	35.2	52.8
RelBitFit	**64.3**	**84.4**	**36.4**	**36.2**	**55.3**

Table 9.6: A Further Reduction of Labeled Training Data. We show the obtained results over the different tasks and for different parameters reduction methods. We train all methods using either $N = 500$ or $N = 100$ labeled training examples.

appropriate parameter reduction method (*i.e.*, either parameter tying, low rank with $d' = 64$, or both), and the appropriate range of the maximal relative distance $L_{\max} \in \{32, 64\}$.

Evaluation Metrics. For evaluation, we follow the standard practice and report the appropriate metric for each one of the GLUE datasets on their corresponding validation sets. For details about the metrics used for each dataset, see Table 9.2.

9.5.2 Ablation Results

In this ablation section, we investigate the parameter efficiency and effectiveness of the different variations of the proposed RelBitFit method. In Table 9.3, we show the number of training parameters of each of these variations and notice that both BitFit and RelBitFit consume a minimal amount of parameters for a given task of interest. Additionally, we see that the proposed tricks reduce the number of parameters required for RelBitFit, making it on par with BitFit in terms of efficiency, *i.e.*, both consume $\approx 0.05\%$ of total parameters.

In terms of performances, Table 9.5 shows the obtained results for sing different variations of RelBitFit. Overall, we notice that the parameter reduction methods do not impact the performances in any noticeable way and can even help obtain better results on some datasets, indicating that using the full parameter range of RelBitFit might be unnecessary under the FSFT setting and

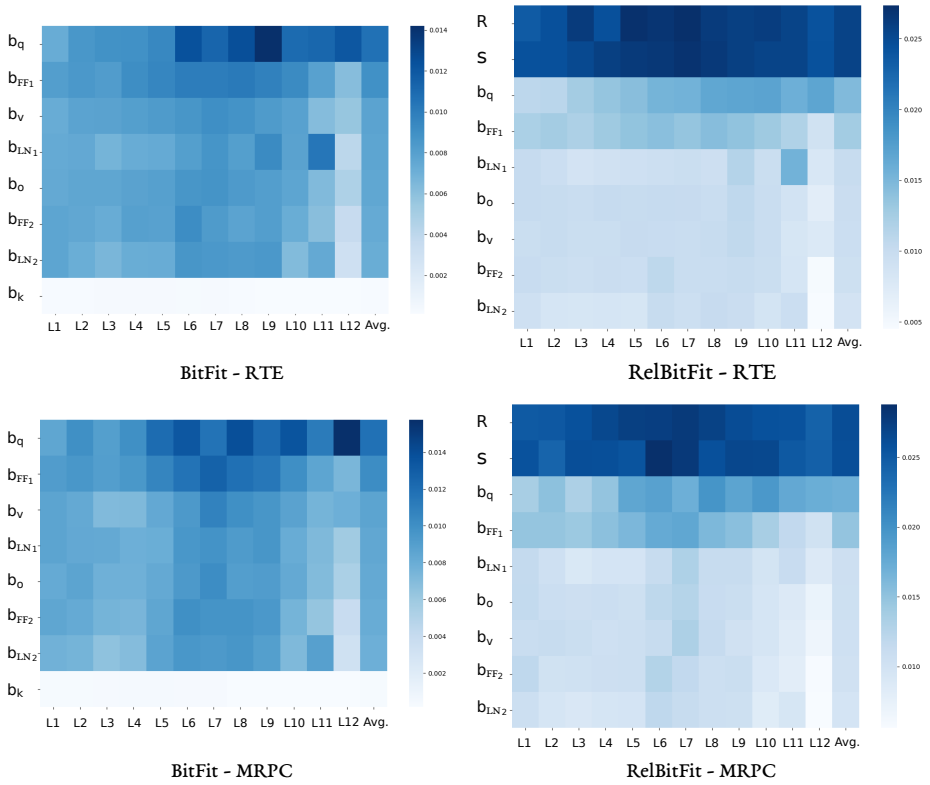


Figure 9.3: Learned Biases. We show the change in the values of the different biases after fine-tuning compared to their initial/pre-trained values. The changes are shown per layer and per bias for both BitFit and RelBitFit when trained on two GLUE tasks: RTE and MRPC.

can help avoid overfitting. For the rest of the results, we use the appropriate parameter reduction method for each one of the GLUE datasets.

9.5.3 Quantitative Results

In this section, we compare RelBitFit with the full fine-tuning baseline (*i.e.*, fine-tuning the whole model), and BitFit, the state-of-the-art parameter-efficient fine-tuning method. The results on the GLUE tasks for $N = 2000$ and $N = 1000$ are shown in Table 9.5. We observe that under the constraint of limited labeled data, our method, RelBitFit, which consists of a simple adjustment of the efficient BitFit method, results in a notable performance boost over both the baseline and BitFit itself.

Additionally, to further reduce the number of labeled examples N used for fine-tuning, and since full fine-tuning becomes unstable in low-label settings, we follow previous FSFT approaches (Gunel et al. 2021) and only use three classification GLUE datasets: QNLI, SST-2, and MNLI. The results obtained $N = 500$ and $N = 100$ are shown in Table 9.6. We observe that the proposed RelBitFil continues to outperform, and sometimes by noticeable margins, when the amount of labeled training data is further reduced. It thus confirms its effectiveness under the constraint of limited labeled data.

9.5.4 Qualitative Results

Finally, following (Zaken et al. 2021), we investigate the fine-tuned biases by computing the amount of change between their initial pre-trained values and their values after fine-tuning. Specifically, for each bias $\mathbf{b}_{(.)}$ we compute the change $\frac{1}{\dim(\mathbf{b})}\left\|\mathbf{b}^0_{(.)} - \mathbf{b}^F_{(.)}\right\|_1$, between the initial $\mathbf{b}^0_{(.)}$ and the final values $\mathbf{b}^F_{(.)}$. Fig. 9.3 shows the results obtained on per bias term and layer basis for both BitFit and RelBitFit trained on two GLUE tasks, RTE and MRPC. First, for BitFit, we observe that the key bias terms $\mathbf{b}_k$ remain unchanged, empirically validating the observation presented in Section 9.3. For RelBitFit, we observe that all the parameters are learned and utilized by the model. And more importantly, the newly added relative biases change the most, indicating that they might play an important role in the final fine-tuned model.

Conclusion

In this chapter, we focused on the textual modality and tackled the setting of FSFT. Specifically, we considered interested in the pre-train the fine-tune parameter and set to develop a more performant method during the fine-tuning stage under the constraint of limited labeled data. To this end, we started with BitFit, and proposed a simple improvement by reformulating the attention computation and injecting learnable relative biases. With detailed experiments, we showed that the proposed approach is highly effective and produces better downstream results.
This chapter culminates the contributions part of this thesis. In the next chapter, we conclude this thesis.

CHAPTER 10

Conclusion & Perspectives

10 Conclusion and perspectives

In this thesis, we tackled the learning problem under limited labeled data. Specifically, we were interested in the set of problems seeking acceptable generalization for a given task of interest in conditions where some human labor supervision is unavailable. All these problems tackle the same general objective: learning a model from some empirically observed data with a limited amount of task-specific labels so that its performances are maximized on a superset inaccessible during training. Tackling this general objective compelled us to consider simpler and more constrained learning paradigms popular within DL tasks. As such, we considered different popular paradigms that explore the learning problem under the constraint of limited labeled data. Such paradigms contemplate either settings with a variable degree of supervision (*i.e.*, SSL, UL, and FSFT), with a distributional shit (*i.e.*, UDA), or a meta-learning problem with many training sub-tasks (*i.e.*, FSL). By tackling them, we built upon their previous methods, proposed incremental improvements to produce better results, and increased the likelihood of adopting such practices in real-world applications. In our work, we also consider different popular vision and language tasks in order to build on existing models and benchmarks and increase their usage prospects. Thus facilitating the training and evaluation process of the proposed methods by utilizing existing models and benchmarks and further increasing their usage prospects.

Revisiting the contributions

Overall, the contributions presented in Chapters 5 to 9 of this thesis focused mostly on introducing a novel set inducting biases, *i.e.*, training objectives, learning procedures, and model architectures, that are more appropriate to the learning paradigms and tasks we addressed. Specifically, after presenting the necessary background knowledge in Chapters 2 to 4, we presented the following contributions:

- **In Chapter 5**, we introduced CCT for semi-supervised image segmentation, which is a novel loss that enforces a consistency of predictions at the pixel level over perturbations injected at the encoder's level. The learning procedure consisted of simultaneously training on the labeled set with a CE loss and the unlabeled set with the CCT loss. The encoder-decoder based model was augmented with auxiliary decoders used over the unlabeled set to compute the proposed CCT objective.
- **In Chapter 6**, we introduced autoregressive segmentation for unsupervised image segmentation, in which we leveraged the structure of autoregressive models to propose a novel view generation method rooted at the pixel level and specific to dense visual tasks such as image segmentation. The learning objective consisted of maximizing the MI between the model's outputs over two views of a given input. These outputs can correspond to either class probabilities (*i.e.*, autoregressive clustering or AC) or feature representations (*i.e.*, autoregressive

representation learning or ARL). The models were built using masked convolution and optional attention blocks to produce these autoregressive views.

- **In Chapter 7**, we introduced TC for the setting of UDA and the tasks of image classification and segmentation. With the TC training objective, we trained explicitly on the target samples by applying a consistency regularization over them, resulting in a more robust and optimal target classifier. Additionally, by coupling TC with a per-class representation alignment (*i.e.*, CLIV objective), we showed further performance improvements. The learning procedure, in this case, incorporates source CE over the labeled source, TC over the unlabeled target, and CLIV over the representations of both sets. For the model, the changes were minimal and consisted of simply using a per-class domain discriminator instead of a single joint one for all classes.
- **In Chapter 8**, we introduced SCL, a contrastive learning-based objective designed to be used during the pre-training stage for TL-based few-show image classification methods. The proposed SCL pushes for similar spatial features for instances of the same classes and for dissimilar features with the rest. When used during pre-training (*i.e.*, meta-training), the learned model exhibits a superior degree of generalization over the unseen meta-testing task. The pre-training learning procedure follows the standard supervised pre-training strategy, but with the SCL objective as an additional auxiliary objective. Regarding the model architecture, the base classification network is augmented with an attention-based spatial alignment module used for spatial alignment and subsequent computation of the SCL objective.
- **In Chapter 9**, we introduce a novel fine-tuning method for large pre-trained LM with a minimal amount of labeled examples available at the fine-tuning stage. The proposed method, dubbed RelBitFit, consisted of reformulating the attention operation by introducing a set of learnable relative biases into the pre-trained model. These new relative biases and the rest of the pre-trained biases are then updated to adapt the model for a given task. It results in a simple, efficient, and effective fine-tuning approach.

Being critical

In this concluding chapter, and to present a set of possible improvements to the proposed methods, viable research directions, and potential future challenges, we first need to reexamine the proposed approaches. But this time, with a critical eye, which will help us layout the groundwork for potential future works and guide us when considering upcoming research objectives. To this end, we highlight the following limitations of the previously presented contributions:

- **Lack of flexible learning paradigms.** In this work, to simulate various settings covering the many possible variations under the constraint of limited labeled data given the generality of the latter, each chapter considered a different paradigm with its pre-existing structures, assumptions, methods, and benchmarks. While the tackled paradigms cover numerous real-world scenarios, being limited to already existing paradigms does ignore a lot of practical cases that do not fit into one of these pre-defined setups. It also opens the door for introducing more general and flexible paradigms in future works, incorporating the different settings we considered into a single and joint one.

- **Lack of appropriate benchmarks.** Throughout this thesis, the training and evaluation process often consisted of converting a dataset, or a benchmark, conceived for the SL setting into a dataset usable for the paradigm of choice under the constraint of limited labeled data. However, such a process results in unrealistic training and evaluation procedures.
  - For SSL, we kept the labels of a subset of the training dataset, discarded the rest of the annotations, and then evaluated the model on the original test/validation set. However, in realistic settings, the unlabeled set might contain noisy and corrupted data or simply data unusable for the desired task. Additionally, we might not have access to large amounts of labels to build test sets big enough to give representative evaluation results.
  - For UL, we have similar issues. During training, we used SL datasets, but without their labels, that contain clean and curated data. Yet, in real-world scenarios, the unlabeled data is often filled with uncurated and unfiltered examples, often resulting in a significant downgrade in performance. For the evaluation, we also used the corresponding test set, making the application of the proposed method unrealistic. Since, in a truly UL setup, we do not have access to such a labeled test set, and even if such a set is available, its size will be very limited.
  - For UDA, the target and source domains were often quite aligned (*e.g.*, both depicting centered image objects or urban scenes from a vehicle's camera) and with a fixed domain gap over all examples. But practically, the domain gap might vary from one example to the other. Some have subtle superfluous variations, while others have more extreme deviations across domains. As for evaluation, we reported the results on the labeled target set, making it an unreasonable setup since, practically, we would only have access to unlabeled target data points.
  - For FSL, we used SL datasets adapted for this setting by simply dividing their classes into seen ones to be used during meta-training and unseen ones to be used during meta-testing. For each subclassification task, a fixed number of classes and a then fixed number of examples per class were sampled randomly. This setup points to many drawbacks, such as the random sampling of the classes of each task. The tasks have limited variability since all classes come from the same original dataset and with a fixed number of classes and examples across all tasks.
  - Finally, for FSFT, in addition to some problems previously mentioned, during training, the model is adapted for a single task at a time with a fixed training set. Nonetheless, in a development setup, we might want to adapt the same model to multiple tasks simultaneously and update it continuously over time with newly obtained data. Both of these cases were considered in the training process.

  All of these limitations point to the need for new benchmarks tailored for the different paradigms that consider more realistic and practical learning settings under the constraint of limited labeled data.
- **Lack of multi-modality.** For each one of the paradigms we considered, we limited ourselves to single-modality tasks and datasets. However, leveraging multiple modalities simul-

taneously to solve a given task might be necessary to equip systems with the desired level of label efficiency.

- **Lack of multi-tasking.** For each one of the paradigms we tackled, we set to solve a single task at a given time. But, multi-tasking or solving multiple tasks concurrently under the same framework might also be instrumental for label-efficient learning. It also might help solve many tasks with scarce data and expensive labels by transferring the learned knowledge from the ones with ample data and/or labels to them.
- **Lack of theoretically sound methods and novel research directions.** If one wishes to describe a common thread between the methods we presented in this thesis, it is that they are heavily intuition-based. We often started with an empirical analysis of the previous methods under a given paradigm which showed their limitations. Based on the intuition gained with such an analysis, we presented a novel and improved method. Additionally, the presented contributions are incremental in nature and were constrained to existing research frameworks. These two critics, however, should motivate our future work to take more risks when exploring upcoming research directions, while also taking more theoretically sound approaches during the development process.
- **Lack of biological relevancy.** When we introduced our work and presented its overarching objective, we set to build label-efficient methods that demonstrated effective learning capabilities similar to that of humans. In this concluding chapter, it is safe to say that our methods did not display such a capability, nor did they contribute to the introduction and advancement of such biologically inspired methods. However, this criticism can often be extended to the most popular DL methods. As the statistician George Box succinctly puts it, *"all models are wrong; some models are useful"*. The presented methods are undoubtedly flawed, but they are, to some degree, useful.
- **Lack of fairness and explainability.** The presented methods lack any analysis of their ethical impact, the fairness of their predictions, or their explainability. In future works, any methods to be developed, especially those to be applied to real-world systems, must consider such questions and incorporate them into the development stage.

A glance into the future

Based on the drawbacks mentioned above and limitations, we propose the following possible research directions that can be considered in future works:

- **A single model, multiple tasks.** Instead of training a model to solve a singular task, and with the recent advances in transformer-based models, a promising research direction is to design massive multi-tasking models that solve many tasks jointly under the same sequence-to-sequence framework without needing task-specific layers or heads (*i.e.*, weights shared across all tasks), *e.g.*, Gato (Reed et al. 2022).
- **A single model, multiple modalities.** Similarly, instead of using one model per modality to solve single or multiple tasks, a better alternative is to design, train and deploy a single-joint model capable of taking as input multiple modalities. Either multi-modal inputs (*e.g.*,

videos, images and their captions, a document and its illustrations) or single-modality inputs (*e.g.*, standalone text and image). In this case, the desired targets correspondingly.

- **A single model, different levels of efficiency.** If such multi-task multi-modality models were to be conceived and developed (*i.e.*, we are starting to see a growing line of work investigating such models *e.g.*, PalM Chowdhery et al. 2022 and *e.g.*, Gato Reed et al. 2022) they must also be designed with different levels of computational efficiency. Depending on the computational environment the model will be deployed on, it might be beneficial to reduce its capabilities to gain some degree of computational efficiency.
- **A single framework, multiple learning paradigms.** Instead of considering particular learning paradigms (*e.g.*, SL, UL, and SSL), either separately or sequentially (*i.e.*, UL for pre-training, then SL for fine-tuning), it might both more helpful and realistic to introduce novel learning settings that study the integration of such paradigms into a single and joint framework. For instance, when training a multi-task model, some tasks might be trained in an unsupervised manner, others in a semi-supervised way, while the rest might follow the supervised approach.
- **A single framework, multiple data sources.** Instead of training on clean and curated training samples originating from a single or a few data distributions, a more realistic training setup might consider different data sources for the same task. Each data source can provide examples from a different data distribution and with different types of noise, both in terms of the inputs and the labels.
- **A single pipeline.** If we consider developing some of the previously mentioned methods, such as multi-task, multi-modal, or flexible learning-based methods, the standard training and evaluation pipeline must also be significantly adjusted and redesigned. Take, for instance, the simple augmentation set during pre-processing. The data augmentations must also be redefined when we have inputs from multiple modalities, by either considering generalized and modality agnostic transformations (*e.g.*, feature level augmentation) or by predefining a set of per-modality augmentations.
- **A single benchmark, multiple settings, and realistic setups.** As we have detailed in the previous section, most of the paradigms outside SL often build their corresponding training and testing sets using fully supervised datasets. However, this results in unrealistic benchmarks that do not reflect real-world scenarios. As such, there is an apparent demand to be filled in terms of DL benchmarks that satisfy the conditions of practicality and realism. For instance, in a UL setting, the training procedure must consider different data sources. The evaluation setup must also be defined based solely on unlabeled data points (*e.g.*, feature inversion or input reconstruction).

Final words

> *Every real story is a never ending story.*
> — Michael Ende, The Neverending Story.

To conclude, while the presented contributions still suffer from many limitations, the road ahead is still long, and a Ph.D. is the start of a research journey in which we will hopefully be able to introduce more constructive, beneficial, and impactful works.

Acronyms

ab-CE	Annealed & Bootstrapped Cross Entropy
AC	Autoregressive Clustering
ARL	Autoregressive Representation Learning
BitFit	BIas-Term FIne-Tuning
CAM	Class Activation Map/Mapping
CCT	Cross-Consistency Training
CE	Cross Entropy
CLIV	Class-Level InVariance
CNN	Convolutional Neural Network
EMA	Exponential Moving Average
FD	Features Distillation
FSFL	Few-Sample Fine-Tuning
FSL	Few-Shot Learning
GAN	Generative Adversarial Network
LM	Language Model
MI	Mutual Information
MLP	Multi-Layer perceptron
MSE	Mean Squared Error
PEFT	Parameter-Efficient Fine-Tuning
RelBitFit	RELative BIas-Term FIne-Tuning
RNN	Recurrent Neural Network
SCL	Spatial Contrastive Learning
SL	Supervised Learning
SSL	Semi-supervised Learning
TC	Target Consistency
UDA	Unsupervised Domain Adaptation
UL	Unsupervised Learning
VAT	Virtual Adversarial Training

Bibliography

Afrasiyabi, A. et al. (2020). "Associative alignment for few-shot image classification". In: *European Conference on Computer Vision*. Springer, pp. 18–35.

Aghajanyan, A. et al. (2021a). "Better fine-tuning by reducing representational collapse". *ICLR*.

Aghajanyan, A. et al. (2021b). "Intrinsic dimensionality explains the effectiveness of language model fine-tuning". *ACL*.

Agrawal, P. et al. (2015). "Learning to see by moving". In: *Proceedings of the IEEE international conference on computer vision*, pp. 37–45.

Ahn, J. et al. (2018). "Learning pixel-level semantic affinity with image-level supervision for weakly supervised semantic segmentation". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4981–4990.

Amodei, D. et al. (2016). "Concrete problems in AI safety". *arXiv preprint arXiv:1606.06565*.

Andrychowicz, M. et al. (2016). "Learning to learn by gradient descent by gradient descent". In: *Advances in neural information processing systems*, pp. 3981–3989.

Arjovsky, M. et al. (2019). "Invariant Risk Minimization". *arXiv:1907.02893*.

Bachman, P. et al. (2019). "Learning representations by maximizing mutual information across views". In: *Advances in Neural Information Processing Systems*, pp. 15509–15519.

Badrinarayanan, V. et al. (2017). "Segnet: A deep convolutional encoder-decoder architecture for image segmentation". *IEEE transactions on pattern analysis and machine intelligence* 39:12, pp. 2481–2495.

Bahdanau, D. et al. (2014). "Neural machine translation by jointly learning to align and translate". *arXiv preprint arXiv:1409.0473*.

Bao, H. et al. (2020). "Unilmv2: Pseudo-masked language models for unified language model pre-training". In: *International Conference on Machine Learning*. PMLR, pp. 642–652.

Baxter, J. (2000). "A model of inductive bias learning". *Journal of artificial intelligence research* 12, pp. 149–198.

Bay, H. et al. (2006). "Surf: Speeded up robust features". In: *European conference on computer vision*. Springer, pp. 404–417.

Bearman, A. et al. (2016). "What's the point: Semantic segmentation with point supervision". In: *European conference on computer vision*. Springer, pp. 549–565.

Becker, S. et al. (1992). "Self-organizing neural network that discovers surfaces in random-dot stereograms". *Nature* 355:6356, pp. 161–163.

Beery, S. et al. (2018). "Recognition in terra incognita". In: *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 456–473.

Bello, I. et al. (2019). "Attention augmented convolutional networks". In: *Proceedings of the IEEE International Conference on Computer Vision*, pp. 3286–3295.

Ben-David, S. et al. (2010). "A theory of learning from different domains". *Machine learning* 79:1-2, pp. 151–175.

Bendale, A. et al. (2016). "Towards open set deep networks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1563–1572.

Bengio, Y. et al. (2013). "Representation learning: A review and new perspectives". *IEEE transactions on pattern analysis and machine intelligence* 35:8, pp. 1798–1828.

Berthelot, D. et al. (2019). "MixMatch: A Holistic Approach to Semi-Supervised Learning". In: *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*. Ed. by H. M. Wallach et al., pp. 5050–5060. URL: https://proceedings.neurips.cc/paper/2019/hash/1cd138d0499a68f4bb72bee04bbec2d7-Abstract.html.

Bertinetto, L. et al. (2019). "Meta-learning with differentiable closed-form solvers". In: *International Conference on Learning Representations*.

Bhushan Damodaran, B. et al. (2018). "Deepjdot: Deep joint distribution optimal transport for unsupervised domain adaptation". In: *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 447–463.

Bielski, A. et al. (2019). "Emergence of object segmentation in perturbed generative models". In: *Advances in Neural Information Processing Systems*, pp. 7256–7266.

Bojar, O. et al. (2016). "Findings of the 2016 conference on machine translation". In: *Proceedings of the First Conference on Machine Translation: Volume 2, Shared Task Papers*, pp. 131–198.

Boudiaf, M. et al. (2020). "Information maximization for few-shot learning". *Advances in Neural Information Processing Systems* 33, pp. 2445–2457.

Bousquet, O. et al. (2003). "Introduction to statistical learning theory". In: *Summer school on machine learning*. Springer, pp. 169–207.

Bouvier, V. et al. (2021). "Robust Domain Adaptation: Representations, Weights and Inductive Bias (Extended Abstract)". In: *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI 2021, Virtual Event / Montreal, Canada, 19-27 August 2021*. Ed. by Z.-H. Zhou. ijcai.org, pp. 4750–4754. DOI: 10.24963/ijcai.2021/644. URL: https://doi.org/10.24963/ijcai.2021/644.

Brostow, G. J. et al. (2009). "Semantic object classes in video: A high-definition ground truth database". *Pattern Recognition Letters* 30:2, pp. 88–97.

Brown, T. et al. (2020). "Language models are few-shot learners". *Advances in neural information processing systems* 33, pp. 1877–1901.

Caesar, H. et al. (2018). "Coco-stuff: Thing and stuff classes in context". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1209–1218.

Cao, K. et al. (2019). "Learning imbalanced datasets with label-distribution-aware margin loss". In: *Advances in Neural Information Processing Systems*, pp. 1567–1578.

Carmon, Y. et al. (2019). "Unlabeled data improves adversarial robustness". In: *Advances in Neural Information Processing Systems*, pp. 11190–11201.

Caron, M. et al. (2018). "Deep clustering for unsupervised learning of visual features". In: *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 132–149.

Carreira, J. et al. (2017). "Quo vadis, action recognition? a new model and the kinetics dataset". In: *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6299–6308.

Caruna, R. (1993). "Multitask learning: A knowledge-based source of inductive bias". In: *Machine learning: Proceedings of the tenth international conference*, pp. 41–48.

Celikyilmaz, A. et al. (2020). "Evaluation of text generation: A survey". *arXiv preprint arXiv:2006.14799*.

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

Chen, L.-C. et al. (2016). "Attention to scale: Scale-aware semantic image segmentation". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3640–3649.

Chen, L.-C. et al. (2017a). "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs". *IEEE transactions on pattern analysis and machine intelligence* 40:4, pp. 834–848.

Chen, L.-C. et al. (2017b). "Rethinking atrous convolution for semantic image segmentation". *arXiv preprint arXiv:1706.05587*.

Chen, M. et al. (2019). "Unsupervised object segmentation by redrawing". In: *Advances in Neural Information Processing Systems*, pp. 12705–12716.

Chen, T. et al. (2020). "A simple framework for contrastive learning of visual representations". *arXiv preprint arXiv:2002.05709*.

Chen, W.-Y. et al. (2019). "A closer look at few-shot classification". In: *International Conference on Learning Representations*.

Chen, X. et al. (2018). "PixelSNAIL: An Improved Autoregressive Generative Model". In: *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Workshop Track Proceedings*. OpenReview.net. URL: https://openreview.net/forum?id=Sk7gI0CUG.

Chen, X. et al. (2019). "Transferability vs. discriminability: Batch spectral penalization for adversarial domain adaptation". In: *International Conference on Machine Learning*, pp. 1081–1090.

Chen, Y. (2015). "Convolutional neural network for sentence classification". MA thesis. University of Waterloo.

Chicco, D. et al. (2020). "The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation". *BMC genomics* 21:1, pp. 1–13.

Child, R. et al. (2019). "Generating Long Sequences with Sparse Transformers". *CoRR* abs/1904.10509. arXiv: 1904.10509. URL: http://arxiv.org/abs/1904.10509.

Chowdhary, K. (2020). *Fundamentals of artificial intelligence*. Springer.

Chowdhery, A. et al. (2022). "Palm: Scaling language modeling with pathways". *arXiv preprint arXiv:2204.02311*.

Chung, J. et al. (2014). "Empirical evaluation of gated recurrent neural networks on sequence modeling". *arXiv preprint arXiv:1412.3555*.

Cicek, S. et al. (2019). "Unsupervised domain adaptation via regularized conditional alignment". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 1416–1425.

Ciresan, D. et al. (2012). "Deep neural networks segment neuronal membranes in electron microscopy images". In: *NeurIPS*, pp. 2843–2851.

Clark, K. et al. (2018). "Semi-supervised sequence modeling with cross-view training". *arXiv preprint arXiv:1809.08370*.

Clark, K. et al. (2019). "What does bert look at? an analysis of bert's attention". *arXiv preprint arXiv:1906.04341*.

Coleman, G. B. et al. (1979). "Image segmentation by clustering". *Proceedings of the IEEE* 67:5, pp. 773–785.

Cordonnier, J.-B. et al. (2020a). "Multi-head attention: Collaborate instead of concatenate". *arXiv preprint arXiv:2006.16362*.

– (2020b). "On the Relationship between Self-Attention and Convolutional Layers". In: *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net. URL: https://openreview.net/forum?id=HJlnC1rKPB.

Cordts, M. et al. (2016a). "The cityscapes dataset for semantic urban scene understanding". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3213–3223.

– (2016b). "The cityscapes dataset for semantic urban scene understanding". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3213–3223.

Cubuk, E. D. et al. (2019a). "Autoaugment: Learning augmentation strategies from data". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 113–123.

Cubuk, E. D. et al. (2019b). "RandAugment: Practical data augmentation with no separate search". *arXiv preprint arXiv:1909.13719*.

Cubuk, E. D. et al. (2019c). "AutoAugment: Learning Augmentation Strategies From Data". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Dai, J. et al. (2015a). "BoxSup: Exploiting Bounding Boxes to Supervise Convolutional Networks for Semantic Segmentation". *2015 IEEE International Conference on Computer Vision (ICCV)*. DOI: 10.1109/iccv.2015.191. URL: http://dx.doi.org/10.1109/ICCV.2015.191.

– (2015b). "Boxsup: Exploiting bounding boxes to supervise convolutional networks for semantic segmentation". In: *Proceedings of the IEEE international conference on computer vision*, pp. 1635–1643.

Dai, Z. et al. (2019). "Transformer-xl: Attentive language models beyond a fixed-length context". *ACL*.

Davies, D. L. et al. (1979). "A cluster separation measure". *IEEE transactions on pattern analysis and machine intelligence* 2, pp. 224–227.

Deng, J. et al. (2009). "Imagenet: A large-scale hierarchical image database". In: *2009 IEEE conference on computer vision and pattern recognition*. Ieee, pp. 248–255.

Devlin, J. et al. (2019). "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*. Ed. by J. Burstein et al. Association for Computational Linguistics, pp. 4171–4186. DOI: 10.18653/v1/n19-1423. URL: https://doi.org/10.18653/v1/n19-1423.

Devries, T. et al. (2017). "Improved Regularization of Convolutional Neural Networks with Cutout". *CoRR* abs/1708.04552. arXiv: 1708.04552. URL: http://arxiv.org/abs/1708.04552.

Dhillon, G. S. et al. (2020). "A baseline for few-shot image classification". In: *International Conference on Learning Representations*.

Dodge, J. et al. (2020). "Fine-Tuning Pretrained Language Models: Weight Initializations, Data Orders, and Early Stopping". *CoRR* abs/2002.06305. arXiv: 2002.06305. URL: https://arxiv.org/abs/2002.06305.

Doersch, C. et al. (2015). "Unsupervised Visual Representation Learning by Context Prediction". In: *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*. IEEE Computer Society, pp. 1422–1430. DOI: 10.1109/ICCV.2015.167. URL: https://doi.org/10.1109/ICCV.2015.167.

Doersch, C. et al. (2020). "CrossTransformers: spatially-aware few-shot transfer". In: *Advances in Neural Information Processing Systems*.

Dosovitskiy, A. et al. (2014). "Discriminative unsupervised feature learning with convolutional neural networks". *Advances in neural information processing systems* 27.

Dvornik, N. et al. (2019). "Diversity with cooperation: Ensemble methods for few-shot classification". In: *IEEE International Conference on Computer Vision*.

Edelman, G. M. (1987). *Neural Darwinism: The theory of neuronal group selection*. Basic books.

Elsayed, G. et al. (2018). "Large margin deep networks for classification". In: *Advances in neural information processing systems*, pp. 842–852.

Ettinger, A. (2020). "What BERT is not: Lessons from a new suite of psycholinguistic diagnostics for language models". *Transactions of the Association for Computational Linguistics* 8, pp. 34–48.
Evans, R. S. (2016). "Electronic health records: then, now, and in the future". *Yearbook of medical informatics* 25:S 01, S48–S61.
Everingham, M. et al. (2010). "The pascal visual object classes (voc) challenge". *International journal of computer vision* 88:2, pp. 303–338.
Fan, J. et al. (2001). "Automatic image segmentation by integrating color-edge extraction and seeded region growing". *IEEE transactions on image processing* 10:10, pp. 1454–1466.
Farabet, C. et al. (2012). "Learning hierarchical features for scene labeling". *IEEE transactions on pattern analysis and machine intelligence* 35:8, pp. 1915–1929.
Fard, M. M. et al. (2020). "Deep *k*-Means: Jointly clustering with *k*-Means and learning representations". *Pattern Recognit. Lett.* 138, pp. 185–192. DOI: 10.1016/j.patrec.2020.07.028. URL: https://doi.org/10.1016/j.patrec.2020.07.028.
Federici, M. et al. (2020). "Learning Robust Representations via Multi-View Information Bottleneck". In: *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net. URL: https://openreview.net/forum?id=B1xwcyHFDr.
Finn, C. et al. (2017). "Model-agnostic meta-learning for fast adaptation of deep networks". In: *International conference on machine learning*. PMLR, pp. 1126–1135.
Finn, C. B. (2018). *Learning to learn with gradients*. University of California, Berkeley.
Foret, P. et al. (2021). "Sharpness-aware Minimization for Efficiently Improving Generalization". In: *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net. URL: https://openreview.net/forum?id=6Tm1mposlrM.
Freitag, M. et al. (2017). "Beam Search Strategies for Neural Machine Translation". In: *Proceedings of the First Workshop on Neural Machine Translation, NMT@ACL 2017, Vancouver, Canada, August 4, 2017*. Ed. by T. Luong et al. Association for Computational Linguistics, pp. 56–60. DOI: 10.18653/v1/w17-3207. URL: https://doi.org/10.18653/v1/w17-3207.
French, G. et al. (2019). "Consistency regularization and CutMix for semi-supervised semantic segmentation". *arXiv preprint arXiv:1906.01916*.
Furlanello, T. et al. (2018). "Born-Again Neural Networks". In: *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*. Ed. by J. G. Dy et al. Vol. 80. Proceedings of Machine Learning Research. PMLR, pp. 1602–1611. URL: http://proceedings.mlr.press/v80/furlanello18a.html.
Ganin, Y. et al. (2014). "N4-Fields: Neural Network Nearest Neighbor Fields for Image Transforms". In: *Asian Conference on Computer Vision*. Springer, pp. 536–551.
– (2015). "Unsupervised domain adaptation by backpropagation". In: *International conference on machine learning*, pp. 1180–1189.
Ganin, Y. et al. (2016). "Domain-adversarial training of neural networks". *The Journal of Machine Learning Research* 17:1, pp. 2096–2030.
Gao, L. et al. (2020). "The pile: An 800gb dataset of diverse text for language modeling". *arXiv preprint arXiv:2101.00027*.
Gao, Y. et al. (2021). "Contrastive Prototype Learning with Augmented Embeddings for Few-Shot Learning". *arXiv preprint arXiv:2101.09499*.
Garcia, V. et al. (2018). "Few-shot learning with graph neural networks". In: *International Conference on Learning Representations*.
Gerke, M. (2014). "Use of the stair vision library within the ISPRS 2D semantic labeling benchmark (Vaihingen)".
Germain, M. et al. (2015). "Made: Masked autoencoder for distribution estimation". In: *International Conference on Machine Learning*, pp. 881–889.
Geva, M. et al. (2019). "Are We Modeling the Task or the Annotator? An Investigation of Annotator Bias in Natural Language Understanding Datasets". *arXiv preprint arXiv:1908.07898*.
Gidaris, S. et al. (2018a). "Dynamic few-shot visual learning without forgetting". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4367–4375.
Gidaris, S. et al. (2018b). "Unsupervised Representation Learning by Predicting Image Rotations". In: *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net. URL: https://openreview.net/forum?id=S1v4N2l0-.
– (2018c). "Unsupervised representation learning by predicting image rotations". *arXiv preprint arXiv:1803.07728*.

Gidaris, S. et al. (2019). "Boosting few-shot visual learning with self-supervision". In: *Proceedings of the IEEE International Conference on Computer Vision*, pp. 8059–8068.
Goodfellow, I. et al. (2016). *Deep learning*. MIT press.
Goodfellow, I. et al. (2020). "Generative adversarial networks". *Communications of the ACM* 63:11, pp. 139–144.
Goyal, A. et al. (2020). "Inductive biases for deep learning of higher-level cognition". *arXiv:2011.15091*.
Grandvalet, Y. et al. (2005). "Semi-supervised learning by entropy minimization". In: *Advances in neural information processing systems*, pp. 529–536.
Grauman, K. et al. (2005). "The pyramid match kernel: Discriminative classification with sets of image features". In: *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*. Vol. 2. IEEE, pp. 1458–1465.
Gretton, A. et al. (2007). "A kernel method for the two-sample-problem". In: *Advances in neural information processing systems*, pp. 513–520.
Gretton, A. et al. (2012). "A kernel two-sample test". *Journal of Machine Learning Research* 13:Mar, pp. 723–773.
Guan, H. et al. (2021). "Domain adaptation for medical image analysis: a survey". *IEEE Transactions on Biomedical Engineering* 69:3, pp. 1173–1185.
Gunel, B. et al. (2021). "Supervised contrastive learning for pre-trained language model fine-tuning". *9th International Conference on Learning Representations*.
Guo, D. et al. (2020). "Parameter-efficient transfer learning with diff pruning". *arXiv preprint arXiv:2012.07463*.
Gutmann, M. et al. (2010). "Noise-contrastive estimation: A new estimation principle for unnormalized statistical models". In: *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pp. 297–304.
Hadsell, R. et al. (2006). "Dimensionality reduction by learning an invariant mapping". In: *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*. Vol. 2. IEEE, pp. 1735–1742.
Haeusser, P. et al. (2018). "Associative deep clustering: Training a classification network with no labels". In: *German Conference on Pattern Recognition*. Springer, pp. 18–32.
Handa, A. et al. (2016). "Understanding real world indoor scenes with synthetic data". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4077–4085.
Hariharan, B. et al. (2011). "Semantic contours from inverse detectors". In: *2011 International Conference on Computer Vision*. IEEE, pp. 991–998.
Hartigan, J. A. (1972). "Direct clustering of a data matrix". *Journal of the american statistical association* 67:337, pp. 123–129.
He, K. et al. (2016). "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778.
He, K. et al. (2020). "Momentum contrast for unsupervised visual representation learning". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9729–9738.
He, Z. et al. (2008). "k-ANMI: A mutual information based clustering algorithm for categorical data". *Information Fusion* 9:2, pp. 223–233.
Henaff, O. (2020). "Data-Efficient Image Recognition with Contrastive Predictive Coding". In: *Proceedings of the 37th International Conference on Machine Learning*. Vol. 119. Proceedings of Machine Learning Research. PMLR, pp. 4182–4192.
Hendrycks, D. et al. (2016). "Gaussian error linear units (gelus)". *arXiv preprint arXiv:1606.08415*.
Hendrycks, D. et al. (2020). "AugMix: A Simple Data Processing Method to Improve Robustness and Uncertainty". In: *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net. URL: https://openreview.net/forum?id=S1gmrxHFvB.
Hestness, J. et al. (2017). "Deep learning scaling is predictable, empirically". *arXiv preprint arXiv:1712.00409*.
Hinton, G. et al. (2015). "Distilling the knowledge in a neural network". *arXiv:1503.02531*.
Hjelm, R. D. et al. (2018a). "Learning deep representations by mutual information estimation and maximization". *arXiv preprint arXiv:1808.06670*.
– (2018b). "Learning deep representations by mutual information estimation and maximization". In: *International Conference on Learning Representations*.
Houlsby, N. et al. (2019). "Parameter-efficient transfer learning for NLP". In: *International Conference on Machine Learning*. PMLR, pp. 2790–2799.
Hu, E. J. et al. (2022). "LoRA: Low-Rank Adaptation of Large Language Models". In: *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net. URL: https://openreview.net/forum?id=nZeVKeeFYf9.

Hu, S. X. et al. (2020). "Empirical bayes transductive meta-learning with synthetic gradients". *arXiv preprint arXiv:2004.12696*.
Hu, W. et al. (2017). "Learning discrete representations via information maximizing self-augmented training". In: *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, pp. 1558–1567.
Huang, J. et al. (2006). "Correcting sample selection bias by unlabeled data". *Advances in neural information processing systems* 19.
Huang, P. et al. (2014). "Deep embedding network for clustering". In: *2014 22nd International conference on pattern recognition*. IEEE, pp. 1532–1537.
Huang, Z. et al. (2018). "Weakly-supervised semantic segmentation network with deep seeded region growing". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7014–7023.
Hung, W.-C. et al. (2018). "Adversarial learning for semi-supervised semantic segmentation". *British Machine Vision Conference 2018, BMVC 2018, Newcastle, UK, September 3-6, 2018*.
Hwang, J.-J. et al. (2019). "SegSort: Segmentation by Discriminative Sorting of Segments". In: *Proceedings of the IEEE International Conference on Computer Vision*, pp. 7334–7344.
Ioffe, S. et al. (2015). "Batch normalization: Accelerating deep network training by reducing internal covariate shift". *arXiv preprint arXiv:1502.03167*.
Iscen, A. et al. (2019). "Label propagation for deep semi-supervised learning". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5070–5079.
Isola, P. et al. (2015). "Learning visual groups from co-occurrences in space and time". *arXiv preprint arXiv:1511.06811*.
James, G. et al. (2013). *An introduction to statistical learning*. Vol. 112. Springer.
Ji, X. et al. (2019). "Invariant information clustering for unsupervised image classification and segmentation". In: *Proceedings of the IEEE International Conference on Computer Vision*, pp. 9865–9874.
Jiang, B. et al. (2019). "Semi-supervised learning with graph learning-convolutional networks". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 11313–11320.
Jiang, J. et al. (2007). "Instance weighting for domain adaptation in NLP". In: ACL.
Johansson, F. et al. (2019). "Support and Invertibility in Domain-Invariant Representations". In: *The 22nd International Conference on Artificial Intelligence and Statistics*, pp. 527–536.
Johnson, J. et al. (2017). "Billion-scale similarity search with GPUs". *arXiv:1702.08734*.
Johnson, J. M. et al. (2019). "Survey on deep learning with class imbalance". *Journal of Big Data* 6:1, pp. 1–54.
Joskowicz, L. et al. (2019). "Inter-observer variability of manual contour delineation of structures in CT". *European radiology* 29:3, pp. 1391–1399.
Kalchbrenner, N. et al. (2014). "A convolutional neural network for modelling sentences". *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*.
Kalluri, T. et al. (2019). "Universal semi-supervised semantic segmentation". *019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*.
Kamnitsas, K. et al. (2018). "Semi-Supervised Learning via Compact Latent Space Clustering". In: *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*. Ed. by J. G. Dy et al. Vol. 80. Proceedings of Machine Learning Research. PMLR, pp. 2464–2473. URL: http://proceedings.mlr.press/v80/kamnitsas18a.html.
Kannan, H. et al. (2018). "Adversarial Logit Pairing". *CoRR* abs/1803.06373. arXiv: 1803.06373. URL: http://arxiv.org/abs/1803.06373.
Khosla, P. et al. (2020). "Supervised contrastive learning". In: *Advances in Neural Information Processing Systems*.
Kingma, D. P. et al. (2014a). "Adam: A method for stochastic optimization". *rd International Conference on Learning Representations*.
Kingma, D. P. et al. (2014b). "Semi-supervised learning with deep generative models". *Advances in neural information processing systems* 27.
Kipf, T. N. et al. (2016). "Semi-supervised classification with graph convolutional networks". *5th International Conference on Learning Representations*.
Kirillov, A. et al. (2019). "Panoptic segmentation". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 9404–9413.
Krähenbühl, P. et al. (2011). "Efficient inference in fully connected crfs with gaussian edge potentials". In: *Advances in neural information processing systems*, pp. 109–117.
Krause, J. et al. (2013). "3d object representations for fine-grained categorization". In: *Proceedings of the IEEE international conference on computer vision workshops*, pp. 554–561.

Krizhevsky, A. et al. (2010). "Cifar-10 (canadian institute for advanced research)". *URL http://www. cs. toronto. edu/kriz/cifar. html* 5.

Krizhevsky, A. et al. (2012). "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems*, pp. 1097–1105.

Kuhn, H. W. (1955). "The Hungarian method for the assignment problem". *Naval research logistics quarterly* 2:1-2, pp. 83–97.

Kumar, A. et al. (2011). "A co-training approach for multi-view spectral clustering". In: *Proceedings of the 28th international conference on machine learning (ICML-11)*, pp. 393–400.

Kumar, A. et al. (2017). "Semi-supervised learning with gans: Manifold invariance with improved inference". In: *Advances in Neural Information Processing Systems*, pp. 5534–5544.

Kumar, S. et al. (2007). "Glaucoma screening: analysis of conventional and telemedicine-friendly devices". *Clinical & experimental ophthalmology* 35:3, pp. 237–243.

Lafferty, J. et al. (2001). "Conditional random fields: Probabilistic models for segmenting and labeling sequence data".

Laine, S. et al. (2016). "Temporal ensembling for semi-supervised learning". *arXiv preprint arXiv:1610.02242*.

Lake, B. M. et al. (2017). "Building machines that learn and think like people". *Behavioral and brain sciences* 40.

Larochelle, H. et al. (2011). "The neural autoregressive distribution estimator". In: *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pp. 29–37.

Larsson, G. et al. (2016). "Learning representations for automatic colorization". In: *European conference on computer vision*. Springer, pp. 577–593.

Lazebnik, S. et al. (2006). "Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories". In: *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*. Vol. 2. IEEE, pp. 2169–2178.

LeCun, Y. et al. (1995). "Convolutional networks for images, speech, and time series". *The handbook of brain theory and neural networks* 3361:10, p. 1995.

LeCun, Y. et al. (1998). "Gradient-based learning applied to document recognition". *Proceedings of the IEEE* 86:11, pp. 2278–2324.

Lee, D.-H. (2013). "Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks". In: *Workshop on challenges in representation learning, ICML*. Vol. 3, p. 2.

Lee, J. et al. (2019). "FickleNet: Weakly and Semi-Supervised Semantic Image Segmentation Using Stochastic Inference". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5267–5276.

Lee, K. et al. (2019). "Meta-learning with differentiable convex optimization". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 10657–10665.

Lewis, M. et al. (2019). "Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension". *arXiv preprint arXiv:1910.13461*.

Li, K. et al. (2018). "Tell Me Where to Look: Guided Attention Inference Network". *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. DOI: 10.1109/cvpr.2018.00960. URL: http://dx.doi.org/10.1109/CVPR.2018.00960.

Lim, S. et al. (2019). "Fast autoaugment". In: *Advances in Neural Information Processing Systems*, pp. 6662–6672.

Lin, B. Y. et al. (2019). "CommonGen: A constrained text generation challenge for generative commonsense reasoning". *arXiv preprint arXiv:1911.03705*.

Lin, D. et al. (2016). "Scribblesup: Scribble-supervised convolutional networks for semantic segmentation". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3159–3167.

Lin, T.-Y. et al. (2014). "Microsoft COCO: Common Objects in Context". *Lecture Notes in Computer Science*, pp. 740–755. ISSN: 1611-3349. DOI: 10.1007/978-3-319-10602-1_48. URL: http://dx.doi.org/10.1007/978-3-319-10602-1_48.

Liu, B. et al. (2019). "Deep metric transfer for label propagation with limited annotated data". *2019 IEEE/CVF International Conference on Computer Vision Workshops*.

Liu, H. et al. (2019). "Transferable Adversarial Training: A General Approach to Adapting Deep Classifiers". In: *International Conference on Machine Learning*, pp. 4013–4022.

Liu, P. et al. (2021). "Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing". *arXiv preprint arXiv:2107.13586*.

Liu, Y. et al. (2018). "Learning to propagate labels: Transductive propagation network for few-shot learning". *arXiv preprint arXiv:1805.10002*.

Liu, Y. et al. (2020). "An ensemble of epoch-wise empirical bayes for few-shot learning". In: *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XVI 16*. Springer, pp. 404–421.

Liu, Y. et al. (2019). "Roberta: A robustly optimized bert pretraining approach". *arXiv preprint arXiv:1907.11692*.

Long, J. et al. (2015). "Fully convolutional networks for semantic segmentation". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3431–3440.

Long, M. et al. (2015). "Learning transferable features with deep adaptation networks". In: *International conference on machine learning*. PMLR, pp. 97–105.

Long, M. et al. (2017). "Deep transfer learning with joint adaptation networks". In: *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, pp. 2208–2217.

Long, M. et al. (2018). "Conditional adversarial domain adaptation". In: *Advances in Neural Information Processing Systems*, pp. 1640–1650.

Loshchilov, I. et al. (2018). "Fixing weight decay regularization in adam".

Lowe, D. G. (2004). "Distinctive image features from scale-invariant keypoints". *International journal of computer vision* 60:2, pp. 91–110.

MacCartney, B. (2009). *Natural language inference*. Stanford University.

Mahabadi, R. K. et al. (2021). "Parameter-efficient multi-task fine-tuning for transformers via shared hypernetworks". *arXiv preprint arXiv:2106.04489*.

Marcus, G. (2020). "The Next Decade in AI: Four Steps Towards Robust Artificial Intelligence". *arXiv:2002.06177*.

McClosky, D. et al. (2006). "Effective self-training for parsing". In: *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pp. 152–159.

McCormac, J. et al. (2017). "Scenenet rgb-d: Can 5m synthetic images beat generic imagenet pre-training on indoor segmentation?" In: *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2678–2687.

Medina, C. et al. (2020). "Self-supervised prototypical transfer learning for few-shot classification". *arXiv preprint arXiv:2006.11325*.

Mintz, M. et al. (2009). "Distant supervision for relation extraction without labeled data". In: *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pp. 1003–1011.

Mishra, N. et al. (2018). "A Simple Neural Attentive Meta-Learner". In: *International Conference on Learning Representations*.

Miyato, T. et al. (2018). "Virtual adversarial training: a regularization method for supervised and semi-supervised learning". *IEEE transactions on pattern analysis and machine intelligence*.

Mosbach, M. et al. (2020). "On the stability of fine-tuning bert: Misconceptions, explanations, and strong baselines". *arXiv preprint arXiv:2006.04884*.

Müller, R. et al. (2019). "When does label smoothing help?" In: *Advances in Neural Information Processing Systems*, pp. 4694–4703.

Naik, D. K. et al. (1992). "Meta-neural networks that learn by learning". In: *[Proceedings 1992] IJCNN International Joint Conference on Neural Networks*. Vol. 1. IEEE, pp. 437–442.

Nair, V. et al. (2010). "Rectified linear units improve restricted boltzmann machines". In: *Icml*.

Nar, K. et al. (2019). "Cross-entropy loss and low-rank features have responsibility for adversarial examples". *arXiv preprint arXiv:1901.08360*.

Nguyen, X. et al. (2010). "Estimating divergence functionals and the likelihood ratio by convex risk minimization". *IEEE Transactions on Information Theory* 56:11, pp. 5847–5861.

Niven, T. et al. (2019). "Probing neural network comprehension of natural language arguments". *arXiv preprint arXiv:1907.07355*.

Noroozi, M. et al. (2016). "Unsupervised learning of visual representations by solving jigsaw puzzles". In: *European conference on computer vision*. Springer, pp. 69–84.

Novak, R. et al. (2018). "Sensitivity and Generalization in Neural Networks: an Empirical Study". In: *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net. URL: https://openreview.net/forum?id=HJC2SzZCW.

Och, F. J. et al. (2004). "A smorgasbord of features for statistical machine translation". In: *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics: HLT-NAACL 2004*, pp. 161–168.

Odena, A. (2016). "Semi-supervised learning with generative adversarial networks". *arXiv preprint arXiv:1606.01583*.

Oliva, A. et al. (2007). "The role of context in object recognition". *Trends in cognitive sciences* 11:12, pp. 520–527.

Oord, A. van den et al. (2016). "Pixel Recurrent Neural Networks". In: *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*. Ed. by M.-F. Balcan et al. Vol. 48. JMLR Workshop and Conference Proceedings. JMLR.org, pp. 1747–1756. URL: http://proceedings.mlr.press/v48/oord16.html.

Oord, A. v. d. et al. (2018a). "Representation learning with contrastive predictive coding". *arXiv preprint arXiv:1807.03748*.

– (2018b). "Representation learning with contrastive predictive coding". *arXiv preprint arXiv:1807.03748*.

Oreshkin, B. et al. (2018). "Tadam: Task dependent adaptive metric for improved few-shot learning". In: *Advances in Neural Information Processing Systems*, pp. 721–731.

Ouali, Y. et al. (2020a). "An overview of deep semi-supervised learning". *arXiv preprint arXiv:2006.05278*.

– (2020b). "Autoregressive unsupervised image segmentation". In: *European Conference on Computer Vision*. Springer, pp. 142–158.

– (2020c). "Semi-supervised semantic segmentation with cross-consistency training". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12674–12684.

Ouali, Y. et al. (2020d). "Target Consistency for Domain Adaptation: when Robustness meets Transferability". *arXiv preprint arXiv:2006.14263*.

Ouali, Y. et al. (2021). "Spatial contrastive learning for few-shot classification". In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, pp. 671–686.

Ouyang, X. et al. (2020). "ERNIE-M: Enhanced multilingual representation by aligning cross-lingual semantics with monolingual corpora". *arXiv preprint arXiv:2012.15674*.

Pal, S. K. et al. (1992). "Multilayer perceptron, fuzzy sets, classifiaction".

Pan, S. J. et al. (2009). "A survey on transfer learning". *IEEE Transactions on knowledge and data engineering* 22:10, pp. 1345–1359.

Papandreou, G. et al. (2015a). *Weakly- and Semi-Supervised Learning of a DCNN for Semantic Image Segmentation*. arXiv: 1502.02734 [cs.CV].

Papandreou, G. et al. (2015b). "Weakly-and semi-supervised learning of a deep convolutional network for semantic image segmentation". In: *Proceedings of the IEEE international conference on computer vision*, pp. 1742–1750.

Parmar, N. et al. (2018). "Image transformer". *arXiv preprint arXiv:1802.05751*.

Paszke, A. et al. (2019). "PyTorch: An imperative style, high-performance deep learning library". In: *Advances in Neural Information Processing Systems*, pp. 8024–8035.

Pathak, D. et al. (2016). "Context encoders: Feature learning by inpainting". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2536–2544.

Paullada, A. et al. (2021). "Data and its (dis) contents: A survey of dataset development and use in machine learning research". *Patterns* 2:11, p. 100336.

Pedregosa, F. et al. (2011a). "Scikit-learn: Machine learning in Python". *Journal of machine learning research* 12:Oct, pp. 2825–2830.

– (2011b). "Scikit-learn: Machine learning in Python". *the Journal of machine Learning research* 12, pp. 2825–2830.

Pei, Z. et al. (2018). "Multi-adversarial domain adaptation". In: *Thirty-Second AAAI Conference on Artificial Intelligence*.

Peng, X. et al. (2017). "Visda: The visual domain adaptation challenge". *arXiv preprint arXiv:1710.06924*.

Petroni, F. et al. (2019). "Language models as knowledge bases?" *arXiv preprint arXiv:1909.01066*.

Pham, H. et al. (2021). "Meta pseudo labels". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11557–11568.

Piaget, J. (1976). "Piaget's theory". In: *Piaget and his school*. Springer, pp. 11–23.

Pinheiro, P. et al. (2014). "Recurrent convolutional neural networks for scene labeling". In: *International conference on machine learning*, pp. 82–90.

Plank, B. et al. (2011). "Effective measures of domain similarity for parsing". In: *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pp. 1566–1576.

Pohlen, T. et al. (2017). "Full-resolution residual networks for semantic segmentation in street scenes". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4151–4160.

Qi, H. et al. (2018). "Low-shot learning with imprinted weights". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5822–5830.

Qiao, L. et al. (2019). "Transductive Episodic-Wise Adaptive Metric for Few-Shot Learning". In: *IEEE International Conference on Computer Vision*.

Qiu, X. et al. (2020). "Pre-trained models for natural language processing: A survey". *Science China Technological Sciences* 63:10, pp. 1872–1897.
Quionero-Candela, J. et al. (2009). *Dataset shift in machine learning*. The MIT Press.
Radford, A. et al. (2015). "Unsupervised representation learning with deep convolutional generative adversarial networks". *arXiv preprint arXiv:1511.06434*.
Radford, A. et al. (2018). "Improving language understanding by generative pre-training".
Radford, A. et al. (2019). "Language models are unsupervised multitask learners". *OpenAI blog* 1:8, p. 9.
Raffel, C. et al. (2020). "Exploring the limits of transfer learning with a unified text-to-text transformer." *J. Mach. Learn. Res.* 21:140, pp. 1–67.
Raghu, A. et al. (2019). "Rapid learning or feature reuse? towards understanding the effectiveness of maml". In: *International Conference on Learning Representations*.
Rasmus, A. et al. (2015). "Semi-supervised learning with ladder networks". In: *Advances in neural information processing systems*, pp. 3546–3554.
Ravi, S. et al. (2017). "Optimization as a model for few-shot learning". In: *International Conference on Learning Representations*.
Ravichandran, A. et al. (2019). "Few-shot learning with embedded class models and shot-free meta training". In: *Proceedings of the IEEE International Conference on Computer Vision*, pp. 331–339.
Rawat, W. et al. (2017). "Deep convolutional neural networks for image classification: A comprehensive review". *Neural computation* 29:9, pp. 2352–2449.
Rebuffi, S.-A. et al. (2017). "Learning multiple visual domains with residual adapters". *Advances in neural information processing systems* 30.
Reed, S. et al. (2022). "A generalist agent". *arXiv preprint arXiv:2205.06175*.
Remus, R. (2012). "Domain adaptation using domain similarity-and domain complexity-based instance selection for cross-domain sentiment analysis". In: *2012 IEEE 12th international conference on data mining workshops*. IEEE, pp. 717–723.
Ren, M. et al. (2018). "Meta-Learning for Semi-Supervised Few-Shot Classification". In: *International Conference on Learning Representations*.
Richter, S. R. et al. (2016a). "Playing for data: Ground truth from computer games". In: *European conference on computer vision*. Springer, pp. 102–118.
– (2016b). "Playing for data: Ground truth from computer games". In: *European conference on computer vision*. Springer, pp. 102–118.
Riloff, E. (1996). "Automatically generating extraction patterns from untagged text". In: *Proceedings of the national conference on artificial intelligence*, pp. 1044–1049.
Rizve, M. N. et al. (2021). "In defense of pseudo-labeling: An uncertainty-aware pseudo-label selection framework for semi-supervised learning". *arXiv preprint arXiv:2101.06329*.
Rogers, A. et al. (2020). "A primer in bertology: What we know about how bert works". *Transactions of the Association for Computational Linguistics* 8, pp. 842–866.
Ronneberger, O. et al. (2015). "U-net: Convolutional networks for biomedical image segmentation". In: *International Conference on Medical image computing and computer-assisted intervention*. Springer, pp. 234–241.
Rublee, E. et al. (2011). "ORB: An efficient alternative to SIFT or SURF". In: *2011 International conference on computer vision*. Ieee, pp. 2564–2571.
Ruder, S. (2019). "Neural transfer learning for natural language processing". PhD thesis. NUI Galway.
Rumelhart, D. E. et al. (1986). "Learning representations by back-propagating errors". *nature* 323:6088, pp. 533–536.
Russakovsky, O. et al. (2015). "Imagenet large scale visual recognition challenge". *International journal of computer vision* 115:3, pp. 211–252.
Rusu, A. A. et al. (2019). "Meta-Learning with Latent Embedding Optimization". In: *International Conference on Learning Representations*.
Saenko, K. et al. (2010). "Adapting visual category models to new domains". In: *European conference on computer vision*. Springer, pp. 213–226.
Salakhutdinov, R. et al. (2007). "Learning a nonlinear embedding by preserving class neighbourhood structure". In: *Artificial Intelligence and Statistics*, pp. 412–419.
Salimans, T. et al. (2017). "PixelCNN++: Improving the PixelCNN with Discretized Logistic Mixture Likelihood and Other Modifications". In: *5th International Conference on Learning Representations, ICLR 2017, Toulon, France,*

April 24-26, 2017, Conference Track Proceedings. OpenReview.net. URL: https://openreview.net/forum?id=BJrFC6ceg.

Schaefer, J. et al. (2020). "The use of machine learning in rare diseases: a scoping review". *Orphanet journal of rare diseases* 15:1, pp. 1–10.

Schaekermann, M. et al. (2019). "Understanding expert disagreement in medical data analysis through structured adjudication". *Proceedings of the ACM on Human-Computer Interaction* 3:CSCW, pp. 1–23.

Scott, T. et al. (2018). "Adapted deep embeddings: A synthesis of methods for k-shot inductive transfer learning". In: *Advances in Neural Information Processing Systems*, pp. 76–85.

Selvaraju, R. R. et al. (2017). "Grad-cam: Visual explanations from deep networks via gradient-based localization". In: *Proceedings of the IEEE international conference on computer vision*, pp. 618–626.

Sennrich, R. et al. (2015). "Neural machine translation of rare words with subword units". *arXiv preprint arXiv:1508.07909*.

Shankar, S. et al. (2017). "No classification without representation: Assessing geodiversity issues in open data sets for the developing world". *arXiv preprint arXiv:1711.08536*.

Shaw, P. et al. (2018). "Self-attention with relative position representations". *ACL*.

Shi, W. et al. (2018). "Transductive semi-supervised deep learning using min-max features". In: *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 299–315.

Shi, W. et al. (2016). "Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1874–1883.

Shih, F. Y. et al. (2005). "Automatic seeded region growing for color image segmentation". *Image and vision computing* 23:10, pp. 877–886.

Shu, R. et al. (2018). "A DIRT-T Approach to Unsupervised Domain Adaptation". In: *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net. URL: https://openreview.net/forum?id=H1q-TM-AW.

Simonyan, K. et al. (2014). "Very deep convolutional networks for large-scale image recognition". *arXiv preprint arXiv:1409.1556*.

Smeulders, A. W. et al. (2000). "Content-based image retrieval at the end of the early years". *IEEE Transactions on pattern analysis and machine intelligence* 22:12, pp. 1349–1380.

Smith, L. et al. (2005). "The development of embodied cognition: Six lessons from babies". *Artificial life* 11:1-2, pp. 13–29.

Smith, N. A. (2011). "Linguistic structure prediction". *Synthesis lectures on human language technologies* 4:2, pp. 1–274.

Snell, J. et al. (2017). "Prototypical networks for few-shot learning". In: *Advances in neural information processing systems*, pp. 4077–4087.

Sohn, K. et al. (2020). "Fixmatch: Simplifying semi-supervised learning with consistency and confidence". *arXiv preprint arXiv:2001.07685*.

Song, C. et al. (2019). *Box-driven Class-wise Region Masking and Filling Rate Guided Loss for Weakly Supervised Semantic Segmentation*. arXiv: 1904.11693 [cs.CV].

Song, K. et al. (2019). "Mass: Masked sequence to sequence pre-training for language generation". *arXiv preprint arXiv:1905.02450*.

Song, S. et al. (2015). "Sun rgb-d: A rgb-d scene understanding benchmark suite". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 567–576.

Souly, N. et al. (2017). "Semi supervised semantic segmentation using generative adversarial network". In: *Proceedings of the IEEE International Conference on Computer Vision*, pp. 5688–5696.

Steed, R. et al. (2021). "Image representations learned with unsupervised pre-training contain human-like biases". In: *Proceedings of the 2021 ACM conference on fairness, accountability, and transparency*, pp. 701–713.

Su, J.-C. et al. (2020). "When does self-supervision improve few-shot learning?" In: *European Conference on Computer Vision*. Springer, pp. 645–666.

Subramanya, A. et al. (2014). "Graph-based semi-supervised learning". *Synthesis Lectures on Artificial Intelligence and Machine Learning* 8:4, pp. 1–125.

Sugiyama, M. et al. (2007). "Covariate shift adaptation by importance weighted cross validation". *Journal of Machine Learning Research* 8:May, pp. 985–1005.

Sugiyama, M. et al. (2008). "Direct importance estimation with model selection and its application to covariate shift adaptation". In: *Advances in neural information processing systems*, pp. 1433–1440.

Sukhbaatar, S. et al. (2014). "Training convolutional networks with noisy labels". *arXiv preprint arXiv:1406.2080*.

Sun, B. et al. (2015). "Return of frustratingly easy domain adaptation". *arXiv preprint arXiv:1511.05547*.

Sun, Q. et al. (2019). "Meta-transfer learning for few-shot learning". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 403–412.
Sung, F. et al. (2018). "Learning to compare: Relation network for few-shot learning". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1199–1208.
Sung, Y.-L. et al. (2021). "Training neural networks with fixed sparse masks". *Advances in Neural Information Processing Systems* 34, pp. 24193–24205.
Sutskever, I. et al. (2013). "On the importance of initialization and momentum in deep learning". In: *International conference on machine learning*. PMLR, pp. 1139–1147.
Szegedy, C. et al. (2016). "Rethinking the inception architecture for computer vision". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2818–2826.
Tan, K. C. et al. (2019). "The herbarium challenge 2019 dataset". *arXiv preprint arXiv:1906.05372*.
Tarvainen, A. et al. (2017). "Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results". In: *Advances in neural information processing systems*, pp. 1195–1204.
Thoma, M. (2016). "A survey of semantic segmentation". *arXiv preprint arXiv:1602.06541*.
Thrun, S. (1998). "Lifelong learning algorithms". In: *Learning to learn*. Springer, pp. 181–209.
Tian, Y. et al. (2019). "Contrastive multiview coding". In: *International Conference on Learning Representations*.
Tian, Y. et al. (2020a). "Rethinking Few-Shot Image Classification: a Good Embedding Is All You Need?" In: *European Conference on Computer Vision*. Springer.
Tian, Y. et al. (2020b). "What makes for good views for contrastive learning". In: *Advances in Neural Information Processing Systems*.
Tompson, J. et al. (2015). "Efficient object localization using convolutional networks". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 648–656.
Torrey, L. et al. (2010). "Transfer learning". In: *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques*. IGI Global, pp. 242–264.
Tsai, Y.-H. et al. (2018). "Learning to adapt structured output space for semantic segmentation". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7472–7481.
Tschannen, M. et al. (2019). "On mutual information maximization for representation learning". *arXiv preprint arXiv:1907.13625*.
Tseng, H.-Y. et al. (2020). "Cross-Domain Few-Shot Classification via Learned Feature-Wise Transformation". In: *International Conference on Learning Representations*.
Tsuboi, Y. et al. (2009). "Direct density ratio estimation for large-scale covariate shift adaptation". *Journal of Information Processing* 17, pp. 138–155.
Van den Oord, A. et al. (2016). "Conditional image generation with pixelcnn decoders". In: *Advances in neural information processing systems*, pp. 4790–4798.
Van Horn, G. et al. (2018). "The inaturalist species classification and detection dataset". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 8769–8778.
Vapnik, V. N. (1999). "An overview of statistical learning theory". *IEEE transactions on neural networks* 10:5, pp. 988–999.
Vaswani, A. et al. (2017). "Attention is all you need". *Advances in neural information processing systems* 30.
Venkateswara, H. et al. (2017). "Deep hashing network for unsupervised domain adaptation". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5018–5027.
Verma, V. et al. (2022). "Interpolation consistency training for semi-supervised learning". *Neural Networks* 145, pp. 90–106. DOI: 10.1016/j.neunet.2021.10.008. URL: https://doi.org/10.1016/j.neunet.2021.10.008.
Vinyals, O. et al. (2016). "Matching networks for one shot learning". In: *Advances in neural information processing systems*, pp. 3630–3638.
Vu, T.-H. et al. (2019). "Advent: Adversarial entropy minimization for domain adaptation in semantic segmentation". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2517–2526.
Wallace, E. et al. (2019). "Do NLP Models Know Numbers? Probing Numeracy in Embeddings". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*. Ed. by K. Inui et al. Association for Computational Linguistics, pp. 5306–5314. DOI: 10.18653/v1/D19-1534. URL: https://doi.org/10.18653/v1/D19-1534.
Wang, A. et al. (2018). "GLUE: A multi-task benchmark and analysis platform for natural language understanding". *arXiv preprint arXiv:1804.07461*.

Wang, T. et al. (2020). "Understanding Contrastive Representation Learning through Alignment and Uniformity on the Hypersphere". In: *Proceedings of the 37th International Conference on Machine Learning*.

Wang, X. et al. (2018). "Non-local neural networks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7794–7803.

Wang, X. et al. (2019). "Transferable Normalization: Towards Improving Transferability of Deep Neural Networks". In: *Advances in Neural Information Processing Systems*, pp. 1951–1961.

Wang, Y.-X. et al. (2016). "Learning to learn: Model regression networks for easy small sample learning". In: *European Conference on Computer Vision*. Springer, pp. 616–634.

Wang, Z. et al. (2019). "Characterizing and avoiding negative transfer". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 11293–11302.

Wei, Y. et al. (2017). "Object region mining with adversarial erasing: A simple classification to semantic segmentation approach". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1568–1576.

Wei, Y. et al. (2018). "Revisiting Dilated Convolution: A Simple Approach for Weakly- and Semi-Supervised Semantic Segmentation". *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. DOI: 10.1109/cvpr.2018.00759. URL: http://dx.doi.org/10.1109/cvpr.2018.00759.

Welinder, P. et al. (2010). "Caltech-UCSD birds 200".

Wolf, T. et al. (2020). "Transformers: State-of-the-art natural language processing". In: *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations*, pp. 38–45.

Wolpert, D. H. et al. (1995). *No free lunch theorems for search*. Technical report. Technical Report SFI-TR-95-02-010, Santa Fe Institute.

Wu, Y. et al. (2016). "Google's neural machine translation system: Bridging the gap between human and machine translation". *arXiv preprint arXiv:1609.08144*.

Wu, Z. et al. (2018a). "Improving generalization via scalable neighborhood component analysis". In: *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 685–701.

Wu, Z. et al. (2018b). "Unsupervised feature learning via non-parametric instance discrimination". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3733–3742.

Xie, Q. et al. (2019). "Unsupervised data augmentation". *arXiv preprint arXiv:1904.12848*.

Xie, Q. et al. (2020). "Unsupervised data augmentation for consistency training". *Advances in Neural Information Processing Systems* 33, pp. 6256–6268.

Xu, J. et al. (2019). "Understanding and Improving Layer Normalization". In: *Advances in Neural Information Processing Systems*. Ed. by H. Wallach et al. Vol. 32. Curran Associates, Inc. URL: https://proceedings.neurips.cc/paper/2019/file/2f4fe03d77724a7217006e5d16728874-Paper.pdf.

Yang, B. et al. (2017). "Towards k-means-friendly spaces: Simultaneous deep learning and clustering". In: *international conference on machine learning*, pp. 3861–3870.

Yang, Z. et al. (2016). "Revisiting semi-supervised learning with graph embeddings". In: *International conference on machine learning*. PMLR, pp. 40–48.

Yin, D. et al. (2019). "A fourier perspective on model robustness in computer vision". In: *Advances in Neural Information Processing Systems*, pp. 13255–13265.

You, A. et al. (2019). *TorchCV: A PyTorch-Based Framework for Deep Learning in Computer Vision*. https://github.com/donnyyou/torchcv.

Yu, F. et al. (2015). "Multi-scale context aggregation by dilated convolutions". *4th International Conference on Learning Representations*.

Yun, S. et al. (2019). "Cutmix: Regularization strategy to train strong classifiers with localizable features". In: *Proceedings of the IEEE International Conference on Computer Vision*, pp. 6023–6032.

Zaken, E. B. et al. (2021). "Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models". *arXiv preprint arXiv:2106.10199*.

Zeiler, M. D. et al. (2011). "Adaptive deconvolutional networks for mid and high level feature learning". In: *2011 International Conference on Computer Vision*. IEEE, pp. 2018–2025.

Zhang, A. et al. (2021). "Dive into deep learning". *arXiv preprint arXiv:2106.11342*.

Zhang, H. et al. (2018a). "Self-attention generative adversarial networks". *arXiv preprint arXiv:1805.08318*.

Zhang, H. et al. (2018b). "mixup: Beyond Empirical Risk Minimization". In: *International Conference on Learning Representations*.

Zhang, R. et al. (2016). "Colorful image colorization". In: *European conference on computer vision*. Springer, pp. 649–666.

Zhang, T. et al. (2020). "Bertscore: Evaluating text generation with bert". *8th International Conference on Learning Representations*.

Zhang, T. et al. (2021). "Revisiting few-sample BERT fine-tuning". *9th International Conference on Learning Representations*.

Zhang, Y. et al. (2018). "An overview of multi-task learning". *National Science Review* 5:1, pp. 30–43.

Zhang, Y. et al. (2011). "Transition-based dependency parsing with rich non-local features". In: *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies*, pp. 188–193.

Zhang, Z. et al. (2018). "Generalized cross entropy loss for training deep neural networks with noisy labels". In: *Advances in neural information processing systems*, pp. 8778–8788.

Zhao, H. et al. (2019). "On learning invariant representation for domain adaptation". *Proceedings of the 36th International Conference on Machine Learning*.

Zhao, H. et al. (2017). "Pyramid scene parsing network". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2881–2890.

Zhao, J. et al. (2017). "Multi-view learning overview: Recent progress and new challenges". *Information Fusion* 38, pp. 43–54.

Zheng, S. et al. (2016). "Improving the robustness of deep neural networks via stability training". In: *Proceedings of the ieee conference on computer vision and pattern recognition*, pp. 4480–4488.

Zhou, B. et al. (2016). "Learning Deep Features for Discriminative Localization". *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. DOI: 10.1109/cvpr.2016.319. URL: http://dx.doi.org/10.1109/CVPR.2016.319.

Zhou, B. et al. (2017). "Places: A 10 million image database for scene recognition". *IEEE transactions on pattern analysis and machine intelligence* 40:6, pp. 1452–1464.

Zhu, X. J. (2005). *Semi-supervised learning literature survey*. Technical report. University of Wisconsin-Madison Department of Computer Sciences.