UNIVERSITY OF GENOVA

PHD PROGRAM IN BIOENGINEERING AND ROBOTICS

Knowledge Transferability for Data-Efficient Deep Learning

by

Andrea Maracani

Thesis submitted for the degree of *Doctor of Philosophy* (36th cycle)

December 2023

Dibris

Department of Informatics, Bioengineering, Robotics and Systems Engineering

I would like to dedicate this thesis to all the people who supported me and to the joy of learning. May it inspire others to keep exploring and discovering new things.

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements.

Andrea Maracani
April 2024

Acknowledgements

I extend my deepest gratitude to my supervisors, Lorenzo Natale and Lorenzo Rosasco, for their invaluable guidance and wisdom throughout my PhD journey. Their insights and advice have been essential in shaping my research and academic growth.

Special thanks go to Elisa and Raffaello, whose consistent support and collaboration were instrumental in our discussions, experimental planning, publication writing, and in providing insights that have significantly contributed to my development as a researcher.

I am also grateful to Vito for our numerous brainstorming sessions and collaborative work, which have sparked many innovative project ideas.

My appreciation also goes to Gabriele, Federico C. and V., Nicola and all my colleagues in the HSP group at IIT and the Malga group at the university. The collaborations, the time spent together, filled with engaging conversations and shared experiences, has been both enjoyable and enriching.

I must also express my heartfelt thanks to Davide, my friend and flatmate, for the endless and stimulating discussions on new ideas and research topics that have crossed our paths.

Last but certainly not least, I owe a profound debt of gratitude to my family, whose support and encouragement have been very important for my journey. Their belief in my endeavors has been a constant source of strength and motivation.

Abstract

Deep Learning has demonstrated remarkable progress in various Computer Vision tasks. However, its effectiveness often relies on the availability of large, well-annotated datasets. In many practical scenarios, there is often limited availability of such data. Furthermore, collecting or labeling more samples that align with the post-deployment environment can sometimes be challenging or impossible, resulting in reduced and sub-optimal performance. Transfer Learning (TL) emerges as a promising solution in these contexts, offering methods to leverage the knowledge acquired by a Deep Neural Network (DNN) on one task to enhance its performance on another, especially when data is scarce, or labels are absent. The main objective of this project is to study and analyze TL methodologies in depth, providing novel perspectives, suggesting new directions both for real-world applications and future research in data-efficient Deep Learning. To achieve this, we present a comprehensive analysis of prevalent TL pipelines, particularly focusing on the fine-tuning of DNNs weights from pre-trained models in real-world scenarios. We also conduct an extensive experimental study on Domain Adaptation techniques, assessing numerous datasets, a wide range of DNN architectures and pre-training strategies to determine which are the most important design choices for a successful transferability. Furthermore, we develop new algorithms aimed at enhancing data-efficiency and adaptability (like Object Detection in robotics). We also introduce a new Domain Adaptation technique for Image Classification, characterized by its straightforward design, which favorably competes with state-of-the-art approaches. Our exploration extends to validate Domain Adaptation methodologies in a Sim-to-Real application for robotics, focusing on surface recognition using vision-based tactile sensors. The result of this work is a comprehensive overview of Transfer Learning, emphasizing data and label efficiency, with new solutions to many challenges, addressing both specific real-world applications and more general algorithmic paradigms. From a broader perspective, the algorithms, analyses, and discoveries presented in this research study have substantial implications for the expansive field of data-efficient Deep Learning. In applications such as robotics, autonomous vehicles, industrial automation and healthcare, the refined TL techniques that we introduce can play a crucial role. Enhancing data efficiency allows for reductions in computational costs and training times, facilitating faster and more precise model deployments. Our methodologies emphasize adaptability, offering great value in situations where models face varied and dynamic environments. Furthermore, our findings serve as a robust foundation for subsequent research, allowing for exploration into deeper domain-specific adaptations or integration with emerging AI methodologies.

Table of contents

List of figures

List of tables

Part I

Introduction and Background

Chapter 1

Introduction

1.1 Overview

Humans possess the distinctive feature of efficiently acquiring new skills and seamlessly applying their existing knowledge to improve various competencies and comprehend diverse concepts across different contexts. This capacity to deeply understand concepts, relationships, and create abstractions is a distinctive feature of human intelligence, which remains challenging, if not impossible, to replicate artificially.

Recent advancements in the development of *intelligent machines* are predominantly driven by the field of Deep Learning (DL) [1]. DL represents a subset of Machine Learning that aims to extract valuable latent representations from data, employing complex models known as Deep (Artificial) Neural Networks (DNNs). Despite the theoretical complexities surrounding DL, it has demonstrated remarkable empirical success across numerous Computer Vision (CV) [2] and Natural Language Processing (NLP) [3] tasks. These tasks include Image Classification, Image Segmentation, Object Detection, Action Recognition, Pose Estimation, Monocular Depth Estimation, Image-to-Text Generation, Image Captioning, Text Generation, Human-like Text-based Conversations, Sentiment Analysis and many others.

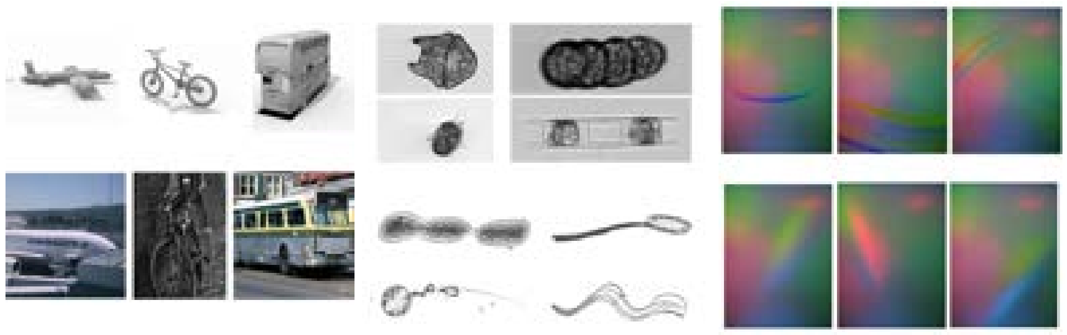
The progress in DL is supported by the continuous research and development of novel architectures and algorithms. However, undeniably, two crucial factors have driven its achievements: the introduction of high-performance hardware accelerators (such as GPUs, TPUs and many others [4]), and the collection of extensive, large-scale datasets (e.g., ImageNet [5]). These two elements have significantly influenced the effectiveness of DNNs, as increased data quantity and the availability of hardware specific accelerators have enabled the expansion of model sizes, resulting in increasingly improved task performance.

Numerous studies, such as [6, 7, 8, 9], have rigorously examined the scaling characteristics of DNNs by focusing on the number of parameters within the model, the size of the dataset utilized, and the extent of computational resources employed. A significant finding from these investigations is the emergence of *power laws* that appear to dictate the scaling behavior of DNNs. For instance, when considering a model with substantial parameters and adequate computational resources, the final error or loss ($\mathcal{E}$) on a dataset comprising N samples can be approximated by $\mathcal{E} \approx N^{-\gamma}$ [10]. The exponent γ, typically close to zero, is suggested to be influenced by the dimension of the data manifold from which training samples are uniformly extracted, as indicated by some theoretical studies [11, 12].

In practical applications, achieving noticeable enhancements in model performance often requires exponentially more data points. Furthermore, it is crucial to recognize that the aforementioned data power law does not apply if there is no scaling in both the model size and computational resources that can lead to sub-optimal performance outcomes, deviating from the optimal Pareto frontier (refer to Fig. 2 in [7]). These observations underscore the inherent limitations within DL methodologies, despite their remarkable achievements in various fields: GPUs (and most of other accelerators) are cost-intensive and consume substantial energy, with consequential environmental implications. Moreover, gathering and annotating data is an arduous, impractical, and error-prone task. For instance, in Semantic Segmentation, the manual pixel-level annotation of images of some tasks can consume hours per image and necessitate professional expertise [13]. This challenge is further worsened when considering domains such as medicine or biology, where data acquisition is constrained by practical limitations.

Another noteworthy shortcoming of DNNs is their task specificity. Unlike humans, when optimally trained for one task, DNNs lack the innate capacity to transfer their knowledge seamlessly to other tasks [14]. Minor discrepancies between the training data and the real-world data encountered during inference can result in substantial performance degradation. Practical applications often face the unavailability of task-specific training data, device variations, environmental fluctuations, and dataset biases, all of which can cause well-trained DNNs to relevant performance degradation.

In this dissertation, we will focus on CV tasks and investigate techniques aimed at enhancing the data efficiency of DNNs through knowledge transferability. Specifically, we will rigorously examine a highly successful and widely adopted technique known as fine-tuning [16]. This involves training a DNN on a large-scale dataset and subsequently adjusting its weights using a smaller dataset specific to the desired task. This approach simplifies data requirements for the target task, allowing a *general-purpose* large-scale dataset to serve as



(a) VisDA Dataset (b) Plankton Images (c) Vision-based tactile sensor

Figure 1.1 Three examples of images from different domains. **(a)** Images from VisDA [15] dataset: synthetic (top) and real (bottom). **(b)** Plankton images acquired with different microscopes. **(c)** Simulated (top) and real (bottom) images of a vision-based tactile sensor.

pre-training for diverse and unrelated tasks. Additionally, we will explore Domain Adaptation (DA), where two domains exist: a labeled source domain with abundant data and a target domain with potentially limited data and absent labels. While both domains correspond to the same task, stylistic differences may exist in the images (some examples are reported in Figure 1.1). DA algorithms aim to leverage information from the source domain to enhance performance and achieve robust results in the target domain.

In summary, this dissertation comprehensively addresses some relevant aspects of DL in the context of CV tasks, offering insights into techniques to enhance the data efficiency of DNNs through knowledge transferability and reusability.

1.2 Contributions

The scope of this thesis is centered around two primary areas of interest. Firstly, we undertake a comprehensive analysis and exploration of Transfer Learning through a series of controlled empirical experiments. Our objective is to gain valuable insights that will benefit both future researchers in the field and practitioners tasked with making informed decisions regarding algorithm implementations. Secondly, we present specific applications and practical solutions for real-world CV problems related to Transfer Learning.

1.2.1 Analysis of Transfer Learning

In our research, a significant Transfer Learning setting is Unsupervised Domain Adaptation (UDA) [17], where two domains are considered: the source domain and the target domain, each characterized by a distinct data distribution. In the context of this study, we assume that images from these domains exhibit stylistic differences while sharing identical semantic information. This divergence between the domains is commonly referred to as *domain shift* and it proves to be a prevalent issue in practical applications. For instance, when images are captured using different cameras, microscopes, or devices, they often exhibit significant variations. Additionally, data collection frequently occurs in controlled environments that differ substantially from the real-world conditions the model will encounter during inference. In some cases, simulated images are used (or incorporated) during training to build the source dataset (e.g., [15]) even if these synthetic images typically exhibit a substantial domain shift compared to real data.

The UDA framework assumes the availability of labeled images from the source domain and unlabeled images from the target domain during training, with the goal of achieving high performance on the target domain. The absence of labels in the target domain makes this setting both challenging and valuable.

Taking a step beyond standard UDA, we explore a relatively recent variant known as Source-Free Unsupervised Domain Adaptation (SF-UDA). In SF-UDA, an additional constraint is introduced: the source data is not accessible during adaptation. Consequently, the training process for the model is divided into two phases: (i) utilizing labeled source data to update the model's weights and (ii) leveraging unlabeled target data (without access to source data) to adapt the model to the new domain in an unsupervised manner. We focus on this specific setting due to its recognized utility in various applications, particularly in scenarios where privacy concerns, intellectual property restrictions, or memory/computational constrains limit the access to source data, leaving only a source-trained model at disposal.

The sequential nature of this setting enables us to analyze the effects of each phase during model training in detail, evaluating the efficacy of pre-training, fine-tuning on the source domain, and the adaptation algorithm on the target domain. Chapter 3 of this thesis presents an **extensive empirical analysis** that thoroughly examines each component.

Furthermore, given that many state-of-the-art (SOTA) SF-UDA algorithms tend to introduce complexity to enhance target performance on specific benchmark datasets and using specific DNN architectures, we introduce a novel, simple yet highly effective SF-UDA algorithm, referred to as **Trust And Balance** (TAB) Adaptation, in Chapter 4. Chapter 5 introduces an additional analysis on the role of architecture, pre-training, and fine-tuning in the context of image classification. We use a real-world problem involving **plankton image classification** to conduct controlled experiments, highlighting the importance and advantages of fine-tuning. Interestingly, we also demonstrate that in certain scenarios (such as the one considered), pre-training DNN on a large-scale natural images dataset (e.g., ImageNet), characterized by a substantial number of classes and high variability, outperforms pre-training on a large-scale In-Domain dataset containing the same type of images (plankton images in our study) that the model will encounter during fine-tuning.

1.2.2 Applications of Transfer Learning

We explore two real-world robotic applications where knowledge transferability and data-efficient pipelines are of great importance. Firstly, we address the challenge of **surface classification** using vision-based tactile sensors in Chapter 6. Here, we leverage a simulator to generate images corresponding to various surface types. In particular, first we sample a point cloud from some 3D objects, then we use a simulator to obtain a synthetic image of sensor contact for each point. Subsequently, we devise an algorithm that, based on point cloud information, can automatically categorize synthetic images into different surface types. We finally introduce a dedicated sim-to-real transfer learning pipeline to bridge the gap between the two domains, which incorporates real sensor-acquired images (without any label) to train a classifier capable of accurately predicting surface types on real sensors.

In Chapter 7, we present a **Weakly-Supervised approach to Transfer Learning for Online Object Detection**. Our method employs a simple policy for dynamic, unsupervised model adaptation, supplemented by a small set of annotations provided by the user as needed, within an Active Learning framework [18]. Specifically, we focus on adapting a model from a domain where objects are physically held by the user (enabling easy automated labeling) to a domain where objects are situated freely on a surface, such as a tabletop.

1.3 Publications

In the following section, we provide a detailed account of the publications that we discuss in this dissertation, along with the works currently under review.

- **Andrea Maracani**, Raffaello Camoriano, Elisa Maiettini, Davide Talon, Lorenzo Rosasco, and Lorenzo Natale. *Key Design Choices in SF-UDA: An In-depth Empirical Analysis.* Under review at IJCV (Chapter 3).

- **Andrea Maracani**, Lorenzo Rosasco and Lorenzo Natale. *Trust And Balance: Few Trusted Samples Pseudo-Labeling and Temperature Scaled Loss for Effective Source-Free Unsupervised Domain Adaptation.* Under review at ECCV 2024 (Chapter 4).

- **Andrea Maracani***, Vito Paolo Pastore*, Lorenzo Natale, Lorenzo Rosasco, and Francesca Odone. *In-domain versus out-of-domain transfer learning in plankton image classification.* Scientific Reports 2023 (Chapter 5).
- Gabriele M. Caddeo*, **Andrea Maracani***, Paolo D. Alfano*, Nicola A. Piga, Lorenzo Rosasco, and Lorenzo Natale. *Sim2Real Bilevel Adaptation for Object Surface Classification using Vision-Based Tactile Sensors.* ICRA 2024 (Chapter 6).
- Elisa Maiettini*, **Andrea Maracani***, Raffaello Camoriano, Giulia Pasquale, Vadim Tikhanoff, Lorenzo Rosasco, and Lorenzo Natale. *From Handheld to Unconstrained Object Detection: a Weakly-supervised On-line Learning Approach.* RO-MAN 2022 (Chapter 7).

1.4 Organization of the dissertation

This dissertation is organized to clearly explain our research. The first two chapters (including this) introduce the motivations and basic ideas. After that, we discuss our specific research contributions in detail. The dissertation ends with a chapter that talks about the overall importance of our work and suggests ideas for future research.

Overview and Background

In Chapter 2, we establish the foundational concepts and review pertinent literature in the field. This chapter serves to connect the broader topics with our specific contributions, setting the stage for the in-depth discussions that follow.

Detailed Contributions

The core of the dissertation is divided into distinct chapters, each dedicated to a unique contribution. These contributions are primarily based on our published works.

- In Chapter 3, we conduct a comprehensive empirical study on key design choices for SF-UDA, providing insights into their impact, efficacy and limitations.
- Chapter 4 introduces *Trust and Balance Adaptation*, our novel approach for SF-UDA in image classification.

- In Chapter 5, we analyze the pre-training and fine-tuning processes of DNNs, focusing on the real-world application of plankton recognition.
- Chapter 6 describes the development of an UDA pipeline aimed at bridging the gap between simulated and real-world data in vision-based tactile sensing for robotics.
- In Chapter 7, we propose a weakly supervised algorithm that integrates Active Learning for domain adaptation in online object detection.

Conclusions and Future Work

Chapter 8 presents the concluding remarks of this dissertation. Here, we discuss the limitations of our work and propose potential directions for future research and investigation.

Note for the reader. The next chapter provides a high-level overview of the key topics and a general discussion of related works. Each contribution chapter, from Chapter 3 to Chapter 7, is designed to be self-contained, mirroring the structure of the original publications. They include specific motivations, introductions, and reviews of related literature. Furthermore, each chapter begins with a *Context* section, placing the contribution within the broader scope of this work. This structure allows readers the flexibility to read each contribution in any order, without losing the overall context and coherence of the dissertation.

Chapter 2

Background

2.1 Supervised Learning

In this section, we delineate the framework of supervised learning, specifically focusing on image classification. Let $\mathcal{X} \in \mathbb{R}^{H \times W \times 3}$ denote the space of RGB images, where H and W represent the height and width of the images, respectively. The label set $\mathcal{Y} = \{c\}_{c=1}^{C}$ encompasses all possible $C \in \mathbb{N}$ categories.

We postulate an underlying probability distribution $\mathcal{D}$ over the data space $\mathcal{V} = \mathcal{X} \times \mathcal{Y}$ that cannot be observed directly. Instead, we are provided with a dataset comprising N i.i.d. samples from this distribution, denoted as $D_{\text{train}} = \{\mathbf{v}_i = (\mathbf{x}_i, y_i)\}_{i=1}^{N} \sim \mathcal{D}^N$. In image classification, we typically focus a specific complex nonlinear function, defined by a DNN architecture. This network is parameterized by weights θ and it is designed to map an input image to a corresponding prediction from the label set: $f_\theta : \mathcal{X} \to \mathcal{Y}$.

We define the *loss* as a measurable function that quantifies the prediction error for a given data sample $(\mathbf{x}, y) = \mathbf{v} \in \mathcal{V}$, in relation to the function's parameters θ:

$$\ell : \mathcal{V} \times \Theta \to \mathbb{R}^{+} \tag{2.1}$$

$$(\mathbf{v}, \theta) \mapsto \ell(\mathbf{v}, \theta) = \ell'(f_\theta(\mathbf{x}), y) \tag{2.2}$$

Here, Θ signifies the parameter space, and ℓ' is a function measuring the discrepancy between the predicted output $f_\theta(\mathbf{x})$ and the actual label y for the input sample $(\mathbf{x}, y)$. The objective is to identify parameters θ^* that minimize the expected error over the data distribution. In other words, we seek parameters such that when new data points are sampled from

$\mathcal{D}$, the function f_{θ^*} demonstrates minimal error. Assuming that a unique solution exists, this is formally expressed as:

$$\theta^* = \arg\min_{\theta\in\Theta} \mathbb{E}_{\mathbf{v}\sim\mathcal{D}}[\ell(\mathbf{v},\theta)] \tag{2.3}$$

where $\mathbb{E}$ denotes the expected value. Due to the unavailability of direct access to $\mathcal{D}$, we address a proxy problem in practice: minimizing the loss function over the training dataset D_{train} through Empirical Risk Minimization [19], aspiring to approximate a solution $\hat{\theta}$ with comparable performance to θ^*:

$$\hat{\theta} = \text{ERM}(D_{\text{train}}) = \arg\min_{\theta\in\Theta} \frac{1}{N}\sum_{i=1}^{N} \ell(\mathbf{v}_i,\theta) \tag{2.4}$$

Considering the intricate nature of DNNs, their over-parametrization, and the non-convex optimization problem presented in Equation 2.4, there currently exists a disparity between theoretical understanding (both in terms of optimal solution identification on training data and generalization on new data from $\mathcal{D}$) and empirical observations in practice [20].

Given the current boundaries of DL theory, the design of the DNN architectures, the optimization procedures [21] and the training strategies (e.g., using regularization [22], data augmentations [23], etc.) are mostly lead by heuristics and practical experimentation. In particular it is common practice to partition the dataset into two subsets: the training set D_{train} and the test set D_{test}. The training set is exclusively used for optimizing the loss function and determining the parameters of the DNN, whereas the test set is reserved for evaluating the model's performance and capabilities with statistical validity, under the assumption that the distribution of future data remains consistent. We refer to the model's ability to perform well on unseen data as **(In-Domain) Generalization (IDG)**. For a DNN with parameters $\hat{\theta}$, this is quantified as:

$$\text{IDG} = \mathbb{E}_{\mathbf{v}\sim\mathcal{D}}[\ell(\mathbf{v},\hat{\theta})] \tag{2.5}$$

Empirically, we estimate this generalization using the test set:

$$\widehat{\text{IDG}} = \frac{1}{|D_{\text{test}}|}\sum_{\mathbf{v}\in D_{\text{test}}} \ell(\mathbf{v},\hat{\theta}) \tag{2.6}$$

Notably, the loss function ℓ is tailored to achieve the desired task, but it may not directly reflect the specific metric of interest. In image classification, the focus often lies on classifi-

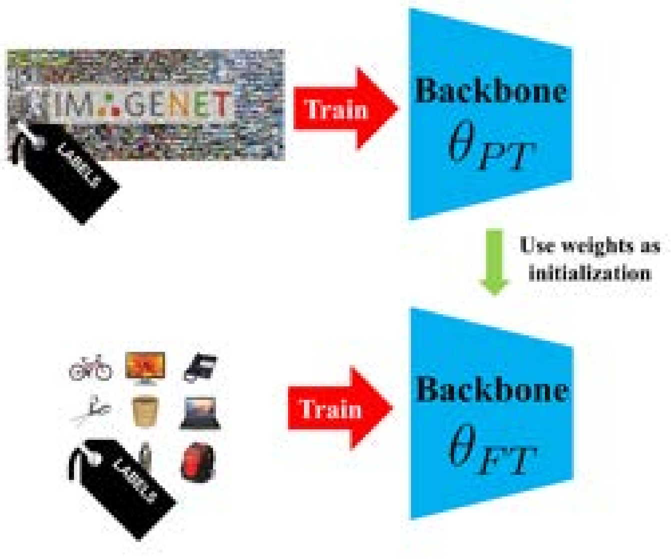


Figure 2.1 After a DNN backbone has been chosen, it is (pre-) trained on a large scale dataset, such as ImageNet. Then the weights are used as initialization for a subsequent training (fine-tuning) on specific task of interest.

cation accuracy. Therefore, in such scenarios, we might redefine IDG in terms of average accuracy on the test set, even if it is a slight deviation from the standard notation:

$$\widehat{\text{IDG}}_{(\text{accuracy})} = \frac{1}{|D_{\text{test}}|} \sum_{(\mathbf{x},y)\in D_{\text{test}}} \mathbb{1}(f_{\hat{\theta}}(\mathbf{x}) = y) \tag{2.7}$$

where $\mathbb{1}(\cdot)$ represents the indicator function.
Another crucial aspect is the model's performance when the underlying data distribution $\mathcal{D}$ shifts. Suppose we introduce an additional distribution $\mathcal{D}'$ that differs from $\mathcal{D}$ but retains some similarities. For instance, both distributions might contain semantically identical images but vary in style. We define **Out-Of-Distribution Generalization (ODG)** [24] as the model's effectiveness when trained on data from $\mathcal{D}$ and evaluated on data from $\mathcal{D}'$. Practically, this is assessed using a test set from $\mathcal{D}'$, applying equations 2.6 and 2.7 to measure the model's performance.

2.2 Fine-tuning

The significant progress in DL has been greatly aided by the availability of large datasets. However, when confronted with a limited dataset, training a DNN from scratch often yields sub-optimal results. Moreover, in scenarios with scarce data, large models may not provide an advantage over smaller ones due to the risk of overfitting. In response to these challenges, **fine-tuning** has emerged as a prevalent and effective strategy. Fine-tuning refers to the process of training a DNN on a specific dataset, beginning not with randomly initialized weights, but rather with weights that have been pre-trained on a different dataset (and/or task). This process typically involves the following steps (summarized in Figure 2.1):

1. Selection of a DNN architecture and the initialization of its weights with random values (f_θ).

2. Training of the DNN using a large-scale dataset, such as ImageNet [5], to acquire pre-trained (PT) weights ($\theta \rightarrow \theta_{PT}$).

3. Refinement of the pre-trained weights using the dataset pertinent to the target task to obtain the fine-tuned (FT) weights ($\theta_{PT} \rightarrow \theta_{FT}$).

This approach offers two key advantages. Firstly, even with limited task-specific data, fine-tuning can still yield robust performance. Secondly, the resource-intensive pre-training phase, which equips the model with the capability to recognize relevant features in images, needs to be conducted only once. The acquired pre-trained weights can then serve as a starting point for various fine-tuning tasks. Typically, pre-training is executed in the context of image classification, followed by fine-tuning for different classification tasks or other applications like object detection or semantic segmentation.

2.2.1 Related Works and Contributions

Fine-tuning

The importance of the fine-tuning technique in DL has been explored in numerous studies. These works typically focus on proposing new heuristics and guidelines to enhance the adjustment of model weights or to identify optimal hyperparameters. Additionally, they examine the fine-tuning process, highlighting its benefits and limitations in specific contexts.

Kolesnikov et al. [25] present an efficient method for fine-tuning DNNs for visual tasks. Their work underscores the advantages of extensive pre-training and a refined fine-tuning process that adapts effectively to various dataset sizes. The approach significantly boosts the adaptability and performance of DNNs in diverse visual applications.

Huh et al. [26] explore the critical factors that contribute to the effectiveness of ImageNet in transfer learning. The research explores whether the dataset's size, the number of classes, or a balance of these elements most significantly impacts the successful transfer of learned features to new tasks. Insights into optimizing datasets for effective pre-training in DL are provided, with a focus on balancing dataset size, class diversity, and granularity.

Studer et al. [27] assess the impact of pre-training DNNs on ImageNet for historical document analysis. Their study evaluates the effectiveness of this technique across various tasks, such as character recognition, style classification, and manuscript dating. Results indicate that while ImageNet pre-training generally enhances performance in classification and image retrieval, its influence on semantic segmentation is mixed, suggesting a need for task-specific considerations in transfer learning.

Ding et al. [28] investigate efficient strategies for fine-tuning large pre-trained language models. Their study introduces *delta-tuning*, a technique that updates only a small subset of the model's parameters, thereby notably reducing computational and storage demands. The paper examines a range of delta-tuning methods, demonstrating their effectiveness and practicality in adapting large pre-trained language models to a variety of applications.

Yosinski et al. [29] examine the transferability of features in DNNs, particularly focusing on which layers transfer effectively and the influencing factors. Their research identifies that transferability decreases with increasing differences between tasks and highlights two main challenges: higher-layer specialization to specific tasks and optimization difficulties when separating co-adapted layers.

Chu et al. [30] concentrate on fine-tuning visual classifiers for new domains. They systematically explore and recommend strategies based on the visual similarity to the pre-training dataset and the quantity of available training data. Their study concludes that utilizing as many layers as possible from a pre-trained network and then fine-tuning based on the visual proximity to the source yields the best results.

Pre-training

The current trend of scaling up DNNs for improved performance, along with the power scaling laws identified in recent research, has given interest in even larger datasets than ImageNet, such as JFT-300M [31]. Furthermore, the introduction of attention-based [32] architectures in Computer Vision, notably Vision Transformers (ViTs) [33], has further underscored the need for large-scale datasets for pre-training. Given the time and the computational power required to collect data and pre-train models on such extensive datasets, alternative pre-training techniques beyond standard supervised learning have been proposed.

For example, Steiner et al. [34] conduct an in-depth study on training Vision Transformers. Their work emphasizes the interplay between data augmentation, model regularization, and training data size in optimizing ViTs. A key finding is that careful application of data augmentation and model regularization can offset the limitations of smaller datasets, proving particularly effective for medium-sized datasets like ImageNet. The study also investigates how dataset size affects model generalizability, showing that models trained on larger datasets such as ImageNet21k exhibit more versatile features, beneficial for various tasks.

Ridnik et al. [35] introduce a novel pre-training approach using the ImageNet21k dataset. This method involves a pre-processing technique that transforms ImageNet21k into an efficient and high-quality pre-training dataset. The study contrasts single-label and multi-label pre-training methods, developing a new *semantic softmax* scheme that takes advantage of ImageNet21k's hierarchical structure. The approach is shown to enhance performance on various downstream tasks across a range of architectures, including those oriented towards mobile devices.

In another direction, **Self-Supervised Learning (SSL)** is emerging as a powerful strategy to train DNNs in the absence of labeled data. SSL algorithms leverage large amounts of unlabeled data by designing pretext learning tasks in which the data inherently offers its own supervision. In particular, these approaches extract a supervisory signal from the intrinsic structure of the data enabling models to learn useful features in an unsupervised fashion. Gui et al. [36] provide a comprehensive overview of SSL's development, its diverse algorithms, and its applications in fields like Computer Vision and Natural Language Processing. SSL is particularly notable for its potential in contexts where labeled data is scarce or costly, offering a viable alternative to conventional supervised learning methods. Relevant methods include MOCO v1 [37], MOCO v2 [38], Masked Autoencoders [39], BeIT [40], DINO [41], and DINO v2 [42].

Our contributions

In Chapter 3, we present an analysis across numerous datasets and DNN architectures. We compare the performance of classifiers trained on features from pre-trained, fixed models against those fine-tuned on the same datasets. This comprehensive comparison underscores the effectiveness of fine-tuning in enhancing model performance.

Moreover, in Chapter 5, our research shows the impact of pre-training on datasets like ImageNet and ImageNet21k for enhancing image classification tasks, particularly in a specialized domain. We focus on classifying microscopic plankton images which are very different from ImageNet's natural images. Our results reveal that pre-training on ImageNet, coupled with fine-tuning, outperforms training exclusively on extensive plankton datasets. This finding emphasizes the value of broad pre-training encompassing various classes and diversities.

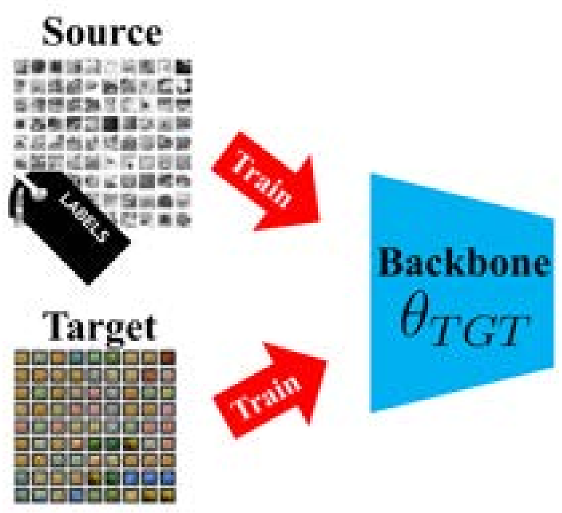


Figure 2.2 In UDA, a labeled source domain and an unlabeled target domain are used concurrently with the goal of training a DNN backbone and obtaining enhanced performance on the target domain.

2.3 Unsupervised Domain Adaptation

In Section 2.1 we introduced ODG, where a model is trained on data from one distribution and tested on data sampled from another. We now introduce **Unsupervised Domain Adaptation (UDA)**, presenting approaches to specifically bridge the gap between training and test. In this context (as in ODG) we consider two different distributions within the data space $\mathcal{V}$: the source domain $\mathcal{D}_S$ and the target domain $\mathcal{D}_T$. These domains are usually related to similar problems but show some differences, such as varying image styles. The source domain typically is more accessible and a (potentially large scale) labeled dataset sampled from it is available. For instance, simulators might be used to generate and label images automatically. On the other hand, the target domain is our main area of interest where we aim to achieve good performance. However, in the UDA setting, we only have access to unlabeled images from this domain.

The rationale behind UDA is that labeling images can be expensive and time-consuming, while acquiring unlabeled images is generally easier. Therefore, UDA algorithms use data from the source and unlabeled data from the target to improve performance on the target domain.

In general, in UDA two separate datasets are available during the training process (see Figure 2.2):

- **Source dataset**: $D_S \sim \mathcal{D}_S^N, N \in \mathbb{N}$
- **Target dataset**: $D_T \sim \mathcal{D}_T(\mathcal{X})^M, M \in \mathbb{N}$

Here, $\mathcal{D}_T(\mathcal{X})$ is the marginal distribution of the target domain over the input space, as we do not have access to the labels in this domain. UDA algorithms use both datasets to adjust the weights of a DNN with the goal of obtaining improved performance on the target domain. While using source data is straightforward (as labels are available for model tuning), the key challenge is to use unlabeled target data effectively during training. This setting is similar to semi-supervised learning [43], where algorithms try to improve DNN performance using both labeled and unlabeled data. However, UDA is distinct as it uses unlabeled data from a different domain. Various UDA algorithms combine a supervised objective (calculated with source domain labels) with an unsupervised objective (based on the unlabeled images of the target domain).

Evaluation. UDA methods can be evaluated based on two different methods: the **inductive** setting and the **transductive** setting. In the inductive setting, there are two datasets from the target domain: one unlabeled for training and another labeled only for testing. The transductive setting evaluates the algorithm using the labels of the same target images seen during training. This second approach is more common in benchmarks and studies related to image classification.

2.3.1 Related Works and Contributions

Foundation Theory

The fundamental principles of UDA have been extensively examined, presenting a combination of algorithmic strategies and theoretical guarantees. This substantial research has formed the foundation for UDA, influencing numerous modern and innovative DL adaptation methods. These advancements are primarily derived from theoretical knowledge.

Mansour et al. [44] present a detailed theoretical and algorithmic examination of domain adaptation. This work introduces the *discrepancy distance* concept, a novel metric for comparing domain distributions across various loss functions. The authors establish Rademacher complexity bounds for this measure and propose new generalization bounds for domain adaptation. They extend their analysis to regularization-based algorithms like SVMs and kernel ridge regression, emphasizing empirical discrepancy. Their study concludes with the development of discrepancy minimization algorithms and initial experimental results, underscoring their utility in domain adaptation scenarios.

In their work, Ben-David et al. [45] tackle fundamental questions in UDA. They propose a framework to predict when a classifier trained on a source domain will be effective in a target domain. A key innovation is their *H-divergence* measure, derived from unlabeled data, which, alongside empirical source error, predicts classifier performance on the target domain. Their research also investigates the combination of labeled source data with limited labeled target data, offering a learning bound for such situations. This work balances theoretical insights with practical applications in domain adaptation.

For a comprehensive overview of UDA theory, we reference the survey by Redko et al. [46]. This survey offers an extensive theoretical analysis of domain adaptation, encompassing learning bounds, divergence-based learning bounds, hardness results, and bounds involving integral probability metrics. It also considers PAC-Bayesian theory and algorithm-centric domain adaptation, presenting a thorough summary of theoretical progress in this field.

Algorithms for UDA in Computer Vision

In Computer Vision, numerous algorithms have been proposed to mitigate domain shifts utilizing DNNs. We highlight a few notable contributions.

Ganin et al. [47] present *Domain Adversarial Training of Neural Networks* (DANN). DANN's core principle is to minimize the ability of a domain classifier to differentiate between source and target domain, thereby promoting domain-agnostic features. This is achieved via an architecture with a gradient reversal layer that aligns feature distributions during training: the proposed procedure involves an adversarial game between the feature extractor and the domain classifier. The method's effectiveness is shown in various tasks, including sentiment analysis and image classification, demonstrating its potential in enhancing domain adaptation in DL.

Kang et al. [48] introduce *Contrastive Adaptation Network* (CAN), which tackles domain discrepancy in UDA. CAN optimizes a novel metric that accounts for both intra-class and inter-class domain discrepancies. Through an alternating update strategy, CAN achieves improved performance on several image classification benchmarks.

Zhang et al. [49] propose an approach that merges theoretical and algorithmic aspects of domain adaptation. They introduce *Margin Disparity Discrepancy* (MDD), a new divergence concept that incorporates margin theory to establish margin-aware generalization bounds. This leads to an adversarial learning algorithm that effectively unites theoretical concepts with practical applications. Their empirical studies demonstrate its superiority in many domain adaptation tasks.

Jaemin Na et al. [50] develop *FixBi*, an UDA method addressing large domain shifts. FixBi generates multiple intermediate domains between source and target using a fixed ratio-based Mixup [51], and trains complementary source-dominant and target-dominant models. Employing confidence-based learning strategies, including bidirectional matching and self-penalization, FixBi efficiently transfers domain knowledge. Its effectiveness is confirmed through comprehensive experiments and ablation studies on standard benchmarks.

For an exhaustive review of UDA methods, we refer to the survey by Wilson et al. [17]. UDA has also been extensively explored in other CV tasks like Object Detection [52] and Semantic Segmentation [53].

Our Contributions

In Chapter 6, we develop a pipeline leveraging Denoising Diffusion Probabilistic (DDP) [54] models and DANN to bridge the gap between simulated and real images of vision-based tactile sensors. Our goal is to recognize surface types upon contact. We introduce a bilevel adaptation process, initially training a state-of-the-art DDP model for denoising real images. Simulated images are then injected with controlled Gaussian noise, partially altering their style while preserving essential information. These images, post-DDP denoising, resemble real images, retaining the original simulated contact information. Automatic annotations are applied based on the contact surface type, derived from simulator point cloud data. These images, along with unlabeled real images, are then used to train a classifier with DANN to recognize surface types.

In Chapter 7, we explore a novel approach for object detection in robotics, focusing on weakly-supervised learning (WSL) to enhance models initially trained on handheld objects. Our research addresses domain adaptation challenges when transitioning from constrained (handheld) to unconstrained (tabletop) environments. We present a comprehensive analysis of various Active Learning (AL) and Self-Supervised Learning techniques, highlighting their efficacy in reducing manual annotation needs while maintaining high detection accuracy.

2.4 Source-Free Unsupervised Domain Adaptation

Similar to UDA, **Source-Free Unsupervised Domain Adaptation** (SF-UDA) involves a labeled source domain and an unlabeled target domain. The objective is to achieve strong performance on the target domain. However, SF-UDA introduces the additional constraint of not utilizing the source domain data during adaptation. This constraint is crucial in situations where intellectual property, privacy, or memory limitations restrict the use of source data, but access to a source-trained model is available. SF-UDA training consists of two main sequential stages:

1. **Source Training:** a DNN, often pre-trained on a large-scale dataset, is further trained with labeled data from the source domain. Subsequently, the final model is retained, but access to the source data is no longer allowed.
2. **Target Adaptation:** the DNN model from the first phase, along with the target domain dataset, are used for unsupervised adaptation. The aim is to enhance the model's performance on the target domain.

While standard UDA algorithms typically focus on reducing discrepancy measures between the two domains, this method is not applicable in SF-UDA due to the absence of source domain data during the adaptation phase. This constraint poses a greater challenge compared to UDA but it makes the setting appealing since, in real-world scenarios, one might only have access to a model and not the data on which it was trained. In such cases, SF-UDA becomes particularly relevant. Additionally, SF-UDA offers an efficiency advantage: the requirement of only target domain data for adaptation often results in SF-UDA algorithms being faster than their UDA counterparts.

2.4.1 Related Works and Contributions

A pioneering theoretical work of SF-UDA has been presented by Ilja Kuzborskij and Francesco Orabona [55], where they analyze the theoretical aspects of Hypothesis Transfer Learning (HTL). HTL leverages source hypotheses trained on a source domain, rather than directly utilizing source data during the adaptation process. The paper emphasizes the role of algorithmic stability in the context of Regularized Least Squares with biased regularization and addresses the challenge of negative transfer in unrelated domains. It ensures that performance is at least as good as algorithms that do not use source domain information, which is crucial in situations with limited training data. This research contributes significantly to the understanding of incorporating prior knowledge into algorithmic stability and its implications for HTL, offering a new perspective on domain adaptation strategies. Later, the field of SF-UDA have gained popularity with Source HypOthesis Transfer (SHOT) [56] that introduced a simple and effective algorithm using an unsupervised objective (the Information

Maximization loss), together with a pseudo-labeling procedure in order to adapt DNNs in the task of image classification. The paper has been followed by a multitude of novel algorithms and techniques for SF-UDA in CV like AAD [57], NRC [58], A^{2}Net [59], 3C-GAN [60] and many others. For a comprehensive survey we refer to [61] that provides an in-depth analysis of SF-UDA. In particular, it categorizes SF-UDA methods into two primary types: data-based and model-based. The data-based methods focus on reconstructing domains or extracting information from images, while the model-based methods leverage self-training techniques such as pseudo-labeling, entropy minimization, and contrastive learning. The survey comprehensively compares many SF-UDA methods and explores their applications, offering insights into potential future research directions in this field.

Our Contributions

In Chapter 3 we provide a large scale empirical study on SF-UDA. We analyze the main design choices including pre-training, how to deal with the source fine-tuning phase and the adaptation method to be used. We highlight both some strengths and weaknesses of SF-UDA approaches and we systematically study their robustness. In Chapter 4 we present a novel SF-UDA adaptation method for image classification with a simple and effective design. We incorporate some findings from our study in the design of this algorithm that, despite being simpler than other methods, it favourably compare with state-of-the-art methods on different benchmarks.

Part II

Contributions

Chapter 3

Key Design Choices in SF-UDA: An In-depth Empirical Analysis

3.1 Context

In this contribution we provide a comprehensive benchmark framework for Source-Free Unsupervised Domain Adaptation (SF-UDA) in image classification, aiming to clarify the complex relationships among various design factors within SF-UDA. The study rigorously evaluates pre-training datasets and strategies, particularly focusing on both Supervised and Self-Supervised methods, as well as the impact of source fine-tuning. It further examines diverse SF-UDA adaptation techniques, assessing their consistency across datasets, dependence on specific hyperparameters, and applicability across different architectural types. The study highlights gaps in existing benchmark practises, guiding SF-UDA research towards more effective and detailed approaches. It emphasizes the importance of backbone architecture and pre-training dataset selection on SF-UDA performance, serving as an essential reference and providing key insights for future research in this field.

3.2 Introduction

Deep Learning has established itself as the leading approach to tackle most computer vision tasks. However, the performance of DNNs is largely dependent on the availability of large-scale annotated datasets, which may be costly to acquire and also challenging for specialized tasks. To address this issue, the prevalent strategy is to initially pre-train model weights on extensive datasets and then *fine-tune* them for specific tasks using a smaller set of labeled

examples [26, 29, 30]. This approach adheres to the transfer learning [62] paradigm: expertise acquired on one task can subsequently enhance learning performance in related, yet distinct downstream tasks.
Additionally, DNNs exhibit strong performance only when the training and test datasets are drawn from the same distribution. However, in practical applications several factors, such as environmental variations or dataset biases, lead to a domain shift between training (*source domain*) and test data (*target domain*), potentially causing a significant performance degradation.

To bridge the gap between source and target domains, **Unsupervised Domain Adaptation** (UDA) jointly employs labeled data from the source domain and unlabeled data from the target domain to improve model adaptation [17]. In particular, in this study we focus on the more challenging **Source-Free Unsupervised Domain Adaptation** (SF-UDA) setting [56]. SF-UDA presents more constraints than UDA, involving a two-stage training, here referred as *double transfer*: (i) a pre-trained model is first fine-tuned on a labeled source domain, and (ii) it is subsequently adapted by employing only the unlabeled data from the target domain with no further access to any source data. This setting is especially useful in applications where access to source data is constrained due to privacy, communication, and storage issues. [63] conducted a rigorous study on strategies commonly adopted in UDA. In particular design choices like architecture and pre-training strategy have been explored in the context of UDA. Their findings indicate that some recent methodologies may be tailored to excel on specific benchmarks: when certain modifications are applied (e.g., the architecture is changed), such methods under-perform compared to earlier approaches. Motivated by these analyses for UDA, our research centers on the SF-UDA setting, with a focus on understanding the contribution of each phase in its characteristic *double transfer*. Our analysis extends beyond standard benchmark assessments, providing an in-depth characterization of multiple design components, ranging from pre-training methods and adaptation strategies to hyper-parameter sensitivity, and evaluating the effectiveness of fine-tuning on the source domain. Through our analysis, we provide a novel perspective into the strengths and limitations of SF-UDA pipelines, setting a foundation for future progress in the field. The main contributions of this work are the following:

- We propose a benchmark framework to evaluate SF-UDA methods focusing on their general applicability in different experimental settings, their reproducibility, robustness, and failure rates.

- We analyze the influence of the backbone and pre-training dataset choices on the final SF-UDA performance. Our results show a high correlation between the ImageNet top-1 accuracy, Out-of-Distribution Generalization (ODG), and SF-UDA performance across hundreds of architectures, including Convolutional Neural Networks and Vision Transformers. Furthermore, we analyze the performance of Self-Supervised pre-training strategies, which are gaining momentum in the current literature.

- While fine-tuning the backbone on the source domain is common practice in SF-UDA, we show that in some scenarios this leads to severe performance degradation. Additionally, we highlight the marked effect of normalization layer selection on SF-UDA perfomance. Architectures with Layer Normalization consistently outpace those employing Batch Normalization (Sec. 3.8).

The following sections discuss related work (Sec. 3.3), detail our benchmark approach (Sec. 3.4 and 3.5), present experimental results (Sec. 3.6, 3.7 and 3.8), and conclude with insights on findings and directions for future research (Sec. 3.9 and 3.10).

3.3 Related Work

Unsupervised Domain Adaptation (UDA). DNNs and other Machine Learning models typically operate under the assumption that their training and test datasets are drawn from the same distribution [17]. This assumption, however, often does not hold in practical applications, leading to significant performance drops. To address these challenges, Unsupervised Domain Adaptation (UDA) has been proposed. UDA leverages a labeled source domain alongside an unlabeled target domain, aiming to optimize model performance on the latter. Early theoretical works [64, 45, 44] formed the foundation for a variety of UDA algorithms, applicable across fields such as time series data analysis [65], Natural Language Processing [66], Sentiment Analysis [67], and Computer Vision tasks such as video analysis [68], image classification [69], object detection [52], and semantic segmentation [53].
Specifically, UDA has been extensively studied in image classification. Proposed techniques include Adversarial Training [47], Maximum Mean Discrepancy Minimization [48], Bi-directional Matching [50], Margin Disparity Discrepancy [49] and Contrastive Learning [70]. These methods proved successful at enabling DNNs to bridge the gap between source and target domains, thereby enhancing their adaptability and accuracy in diverse and changing environments.

Source-Free Unsupervised Domain Adaptation (SF-UDA). Our study specifically focuses on SF-UDA, a more constrained subset of UDA where there is no granted access to source data during adaptation. SF-UDA emerges as a suitable setting in scenarios where, due to concerns such as intellectual property, privacy, or memory limitations, the labeled source domain is not available during the adaptation phase.

The foundation of SF-UDA is Hypothesis Transfer Learning [55], which inspired the first SF-UDA methods to adapt DNNs in image classification tasks, such as SHOT [56]. These early methods showed competitive performance compared to UDA approaches. Further works introduced a variety of SF-UDA algorithms. These include generative modeling [60], techniques focusing on entropy minimization, transferring Batch Normalization statistics [71], creating surrogate source domains during adaptation [72], employing self-distillation techniques [73], and leveraging the latent structure of source-trained models for adaptation [57, 58]. Contrastive learning has also been employed in this context [74]. A comprehensive review of contemporary SF-UDA methods is available in [75], and the specific methods explored in our study are detailed in Section 3.4.2.

Experimental Studies. The challenges surrounding DNNs training, given their computational, time, and data-intensive requirements, have prompted significant research efforts. Their main goals have been to extract representations suitable for effective transfer to new tasks and to identify the key components enabling more efficient architectures and training methods. Past efforts have explored many aspects of Transfer Learning, as shown in works by Chu et al. [30], Huh et al. [26], and Kornblith et al. [76]. In the UDA setting, [77] explores model selection based on ImageNet performance, while [63] analyzes various pre-training techniques for Domain Generalization [78] and UDA.

Conversely, this work focuses on the potentials and limitations of SF-UDA, a setting gaining traction for its ability to rival traditional UDA in performance while enforcing stricter data constraints. Our comprehensive analysis decouples and studies in depth the properties of each phase in the double transfer process: transitioning from pre-training to the source domain and, then, from the source to the target domain. This level of *modular* decomposition of the SF-UFA pipeline is inherently unfeasible in traditional UDA where the adaptation is performed using the source and target domains, concurrently, in a single phase. Finally, along with our results, we release the experimental framework code, which is a valuable to facilitate future research. In particular, the code [1] not only enables the replication of our results, but it also serves as a foundation for further empirical investigation, as it can be effortlessly expanded with novel methods, architectures, and datasets.

[1]The code will be released at https://www.github.com/andreamaracani/SFUDA_KDC/

3.4 Methods

This section describes the high-level SF-UDA pipeline and introduces the SF-UDA approaches included in our study.

3.4.1 SF-UDA pipeline

In SF-UDA, two data distributions (domains) are defined: the **source domain** and the **target domain**. Under the Closed-set assumption, images from different domains may differ in style, yet share the same label set.

Model training is divided into different phases based on the data availability. In particular, the full process can be summarized in **three main stages**:

1. A model (a DNN backbone) is initially **pre-trained** on a large dataset, e.g., ImageNet or Imagenet21k [5];

2. **First transfer**: the model weights can potentially be fine-tuned (FT) using the labeled source dataset;

3. **Second transfer**: unlabeled images from the target domain are employed for model adaptation to the new domain. Note that no access to the source data is available at this stage.

The outlined modular pipeline is illustrated in Fig. 3.1. This structure facilitates a methodical evaluation of its individual components: the strategy for unsupervised adaptation using target images (Sec. 3.6), the criteria for selecting the pre-training dataset and methodology (Sec. 3.7), and the effectiveness and issues related to fine-tuning the model's weights on the source domain (Sec. 3.8). This work includes a sequence of controlled experiments designed to isolate and evaluate the impact of each individual component on the final outcomes.

3.4.2 SF-UDA methods

To quantitatively assess the performance of SF-UDA techniques across a wide range of configurations, we conduct a detailed software re-implementation guided by the original research papers and their associated code. Importantly, we add the options to employ many different backbones and perform distributed training. In particular, for our analyses we select a set of representative SF-UDA methods: SCA (Sec. 3.4.2), SHOT [56] (Sec. 3.4.2), NRC [58] (Sec. 3.4.2), AAD [57] (Sec. 3.4.2) and PCSR [79] (Sec. 3.4.2).

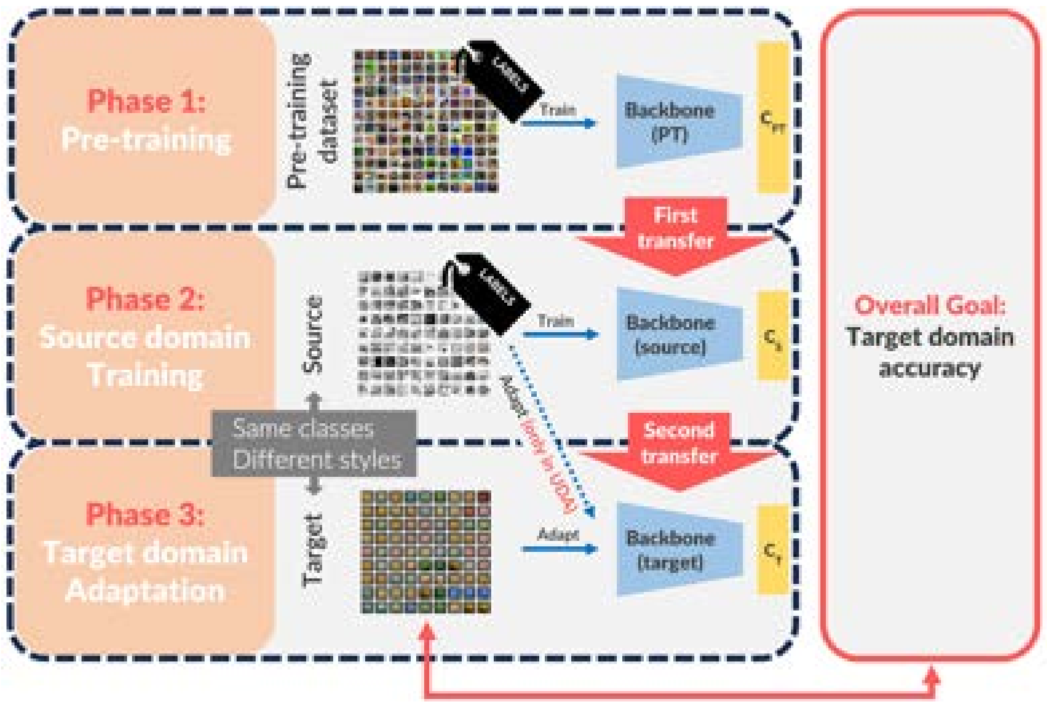


Figure 3.1 **SF-UDA pipeline.** In this work, we meticulously analyze the SF-UDA pipeline. The process begins **pre-training** a backbone (along with its classifier, C_{PT}) on a large dataset, e.g., ImageNet (see Sec. 3.7). This is followed by the **first transfer** phase, where labeled source data is used to refine the backbone and, possibly, train a classifier for the new task (see Sec. 3.8). Then, the **second transfer** phase happens, leveraging unlabeled target data to adapt the model to the target domain (see Sec. 3.6).

Several other SF-UDA approaches exist in the literature [75]. Still, we focus on the aforementioned methods for the following reasons:

- Despite our meticulous implementation process, it was not possible to replicate the results of some methods as presented in their original papers, indicating a sensitivity to particular experimental configurations.
- Some methods were excluded because their official source code was unavailable (e.g., [59, 80]) or their corresponding papers lacked comprehensive training or implementation details.
- We omitted certain methods designed for specific architectures (e.g., [71, 81]) or those requiring unique architectural modifications or additions (e.g., [82, 60]) or external data/resources (e.g., methods designed for multi-source SF-UDA as [83]). These methods were not pertinent to our wide-ranging architectural evaluation.

In the next paragraphs, we briefly describe each SF-UDA method used in our study.

Simple Class Alignment

Simple Class Alignment (**SCA**) is a simple yet effective method, often employed as a part of other domain adaptation algorithms, such as CAN [48] and SHOT [56]. However, its use as a standalone method has not been explored in the literature, and its specific role within multistage algorithms still requires close examination. Furthermore, despite its simplicity, we observe that in some settings it reaches state-of-the-art results (see Sec. 3.6.5). For these reasons, we included it in our study. The main algorithmic steps of SCA are:

1. **Prototype Creation**: generate a representative vector (prototype) for each class in the source domain (for example by averaging the feature vectors of samples in that class).
2. **K-Means Initialization**: use the source prototypes as starting centroids for K-Means clustering on the unlabeled data from the target domain.
3. **Prototype Adaptation and Classification**: apply spherical K-Means [84] to fit prototypes to target domain feature vectors. The final prototypes are then employed by a 1-Nearest Neighbor classifier: each target sample is assigned the class of the closest prototype, employing the cosine similarity as metric.

SCA's distinguishing feature is its capability to provide a classifier for the target domain by aligning source prototypes with the target latent vectors distribution. This enables its use with feature representations from a pre-trained model with no additional refinement. Hence, SCA emerges as a resource-efficient SF-UDA method, rivaling even the most advanced ones in terms of performance.

Source HypOthesis Transfer

Source HypOthesis Transfer [56] (**SHOT**) has become a prominent reference in the field of SF-UDA. Its efficacy has set a baseline for newer methodologies in the field. The adaptation in SHOT is characterized by a sequence of epochs, each consisting of two phases:

1. **Generation of pseudo-labels** for the target samples through source-trained model.
2. Tuning of the feature extractor weights optimizing an objective composed of two terms: the unsupervised **Information Maximization loss** and a standard **Cross-Entropy loss** based on the previously computed pseudo-labels. Notably, the classifier weights remain unaltered in this phase.

For pseudo-labels computation, SHOT employs a modified version of SCA. Utilizing the feature extractor $f(\cdot)$ and the fixed source classifier, the algorithm determines class output probabilities for each target sample $\mathbf{x}^{(i)} \in \mathcal{X}_{\text{TGT}}$ as $\mathbf{p}^{(i)} = (p_1^{(i)}, \ldots, p_C^{(i)})$, where C is the total number of classes. The initial prototype for a generic class $c \in \{1, \ldots, C\}$, i.e., $\mathbf{k}_c$, is obtained as a weighted average of target domain features, according to the following equation:

$$\mathbf{k}_c = \frac{\sum_i p_c^{(i)} \cdot f(\mathbf{x}^{(i)})}{\sum_i p_c^{(i)}}, \tag{3.1}$$

Where i indexes all the samples of the target domain $\mathcal{X}_{\text{TGT}}$. The obtained prototypes $\{\mathbf{k}_c\}_{c=1}^{C}$ serve as the initialization for the spherical K-Means steps of SCA. This leads to a 1-NN classifier generating the pseudo-labels for the target dataset.

Neighborhood Reciprocity Clustering

Neighborhood Reciprocity Clustering [58] (**NRC**) leverages the intrinsic latent structure of target data by focusing on neighborhood information. Specifically, the training aims to bring similar samples closer while distancing dissimilar ones. The **Neighborhood Affinity** is a pivotal concept, distinguishing neighbors by their potential use for accurate supervision. Neighbors are categorized as either reciprocal (RNN) or non-reciprocal (nRNN). RNNs are neighbors that mutually identify each other as nearest neighbors. NRC assumes, supported by empirical evidence, that RNNs offer greater potential for providing valuable adaptation information. Consequently, they are assigned more importance in the overall objective. This perspective is further extended by the **Expanded Neighborhood Affinity**, which considers second-order neighbors (i.e., neighbors of neighbors). NRC devises an objective by integrating these components, ensuring a balanced consideration of both immediate and extended neighborhoods. This is complemented by a **Self-Regularization** term, which employs current predictions as a guide to mitigate the influence of potentially unreliable neighbors.

Attracting and Dispersing

Attracting and Dispersing [57] (**AAD**) bears similarities to NRC, but introduces a simpler methodology and objective. For every target feature vector, two distinct sets are determined: the **Close Neighbor Set**, which encompasses the nearest or most similar representations, and the **Background Set**, consisting of feature vectors unrelated or distant from the current one (often from different classes). The method employs the principles of **Attracting** and

Dispersing. In *Attracting*, the Close Neighbor Set's representations are guided to produce consistent predictions. Conversely, in *Dispersing*, predictions from the Background Set are kept as distant as possible from the ones obtained with the current representations. The model's parameters are optimized with a simple objective to achieve a balance between attraction and dispersion, iteratively refining the process until the model adapts to the target domain.

Polycentric Clustering and Structural Regularization

Polycentric Clustering and Structural Regularization [79] (**PCSR**) is similar to SHOT since it employs the Information Maximization loss and a pseudo-labeling procedure. However, PCSR refines the pseudo-labeling through **Polycentric Clustering**. Leveraging on the K-Means algorithm, it computes multiple centers for each class, enhancing pseudo-label accuracy. PCSR introduces also the use of **MixUp** [85, 86, 87] to provide additional regularization during the adaptation. The result is a composite loss function integrating all the aforementioned objectives for the unsupervised model optimization on the target domain.

3.5 Benchmarking Framework

A significant contribution of this work is the introduction of an open-source benchmarking framework enabling the execution of a systematic large-scale experimental analysis of SF-UDA. Our framework facilitates the construction, training, and testing of SF-UDA methods. Thanks to the integration with state-of-the-art libraries, e.g., `timm` [88], it enables comparisons between more than 500 backbones. Furthermore, it allows to test SF-UDA methods on several datasets, ranging from widely used benchmarks, such as *Office 31* [89] and *Office-Home* [90], to other more recent datasets, such as *DomainNet* [91].
Importantly, the proposed framework also allows to modify and assess individual components of distinct methods in a flexible way. This enables three primary outcomes:

1. A deeper understanding of the SF-UDA setting, which is currently understudied;
2. The establishment of a systematic protocol for the comparison of different SF-UDA methods;
3. Guidance in making key choices during the design of SF-UDA approaches.

In the following, we present the datasets (Sec. 3.5.1) and the backbones (Sec. 3.5.2) considered in our study. Furthermore, we give details on the benchmark protocol (Sec. 3.5.3) and on the experimental setup (Sec. 3.5.4).

3.5.1 Datasets

We now introduce all datasets considered in our study, distinguishing between those used for pre-training and for domain adaptation.

Pre-training datasets. For the *first transfer*, we consider two datasets: ImageNet, composed of 1.2 M images for 1 000 mutually exclusive classes, and the superset ImageNet21k [5], composed of 14 M images for 21 841 not mutually exclusive classes. Specifically, we either consider models pre-trained (i) on ImageNet (*IN*), or (ii) on ImageNet21k and consequently fine-tuned on the 1 000 classes of ImageNet (*IN21k*). It is important to note that the additional fine-tuning step on ImageNet for models pre-trained on ImageNet21k does not significantly impact their performance on downstream tasks. This fine-tuning process has become a widely accepted standard in pre-training protocols, with numerous models being readily available following this method. Models solely pre-trained on ImageNet21k were not included in our analysis due to their limited availability. However, we did evaluate a select few to verify that their performance closely aligns with that of their counterparts that underwent fine-tuning, reinforcing the consistency of our pre-training approach.

Domain adaptation image datasets. These image classification datasets include two or more subsets corresponding to different domains with distinct visual styles and sharing the same classes. In our experiments, we consider: *DomainNet* [91], *ImageClef-DA* [92], *Office-31* [89], *Modern Office-31* [93], *Visda-2017* [15], *Office-Home* [90], and *Adaptiope* [93]. An overview of all datasets is illustrated in Tab. 3.1.

Datasets choice rationale. For pre-training, we include ImageNet and ImageNet21k, as they are open datasets frequently employed in the literature. Consequently, many DNNs pre-trained on these datasets are available. Instead, for domain adaptation, we select 7 datasets. These include popular benchmark datasets (Office31, Office-Home, and Visda) and also other datasets less often used in SF-UDA research (like DomainNet, Adaptiope, Modern Office-31, and ImageClef-DA). The goal of this diverse selection is to ensure a comprehensive evaluation across different scenarios, supporting our findings regarding SF-UDA. To evaluate different SF-UDA methods (Sec. 3.6), we select Office31 and Office-Home, as they are

Table 3.1 **Domain Adaptation datasets** considered in this study. Checkmarks in the final two columns show which datasets are used in the SF-UDA methods evaluation and which in the pre-training and fine-tuning (PT/FT) experiments.

Dataset	**Images**	**Domains**	**Classes**	**SF-UDA Eval**	**PT/FT Exp.**
DomainNet	596 006	6	345		✓
Visda-2017	280 000	2	12		✓
Adaptiope	36 900	3	123	✓	✓
Office-Home	15 588	4	65	✓	✓
Modern Office-31	7 210	4	31		✓
Office-31	4 110	3	31	✓	
ImageClef-DA	2 400	4	12	✓	✓

used as benchmarks in the original papers, and Adaptiope and ImageClef-DA, which are not considered in those studies. This allows to determine if the methods can generalize to new datasets or if their performance strongly depends on the dataset choice. For the experiments on pre-training and fine-tuning (Sec. 3.7 and 3.8), we employ all the domain adaptation datasets mentioned above with the exception of Office31 to avoid possible biases due to its similarity to Modern Office-31. This results in a total of 23 domains and 74 domain pairs (source-target). Notably, for Visda-2017, unlike the usual benchmark, we consider both synthetic-to-real and real-to-synthetic experiments. We consider *domain-pair-averages* across this work as summary metric: we run an experiment for each domain pair (a source and target domain) and we average the accuracy across all the experiments. The same conclusions can be drawn even if *dataset-averages* are considered instead (i.e., by computing the averages for each dataset individually first and, then, across datasets).

3.5.2 Backbone selection and integration

Before any SF-UDA pipeline, the backbone should be chosen (see Sec. 3.4.1): in most of our experiments we rely on the *PyTorch Image Models* Python library (`timm`) [88] that provides access to a remarkably large number of different architectures and pre-trained weights. Specifically, to analyze the pre-training phase, in Sec. 3.7 we evaluate SCA and the baselines (presented in Sec. 3.5.3) on more than 500 models selected among more than 25 different families of architectures (e.g., VGG [94], ResNet [95], EfficientNet [96], ConvNext [97], ViT [33], SWIN [98], Deit [99] and XCiT [100]). Moreover, to evaluate the impact of the fine-tuning step on the source domain (Sec. 3.8 and some experiments of Sec. 3.7), we sample a subset of 59 models, taken from more than 12 families of architectures. Instead, to benchmark SF-UDA methods and test their robustness (Sec. 3.6), we consider two commonly used backbones, namely the convolution-based Resnet50 [95] (that we initialized with TorchVision[2] weights for consistency with original papers) and the Attention-based Vision Transformer Large (ViT-Large) [33]. Finally, for experiments involving Self-Supervised pre-trained networks (Sec. 3.7.2), we use the weights released by the authors of different methods if not available on TorchVision Hub [101]. Notably, our analysis comprises both modern Vision Transfomers and CNNs (like ConvNext) and more traditional architectures.

3.5.3 Evaluation protocol

Our experimental framework is designed to serve as a toolkit for the in-depth analysis of SF-UDA and it enables the evaluation of each individual component of the SF-UDA pipeline. Additionally, our evaluation protocol is standardized across datasets and architectures in order to provide reliable and reproducible assessments.

SF-UDA protocol. For every pre-trained backbone, the final linear layer (i.e., the ImageNet classifier) is removed. Then, we introduce a newly initialized bottleneck, followed by a final linear classification layer tailored to the number of classes of the given task. The bottleneck includes a linear layer which projects the backbone's features to 256 dimensions followed by a normalization layer and a ReLU (or GELU) activation function. Notably, the type of normalization adopted for the bottleneck is contingent on the backbone's architecture, opting for Batch Normalization if present in the backbone or Layer Normalization otherwise (and the same holds for the type of activation function). If the training on the *source domain* is performed with fine-tuning (FT), backbone weights are adjusted and the bottleneck and classifier weights are learnt from scratch. Instead, experiments without FT keep the backbone weights fixed and train the bottleneck and the classifier only. Finally, the adaptation on the *target domain* is performed with the selected SF-UDA technique.

Experimental baselines. We also implement four baselines serving as upper and lower reference bounds for SF-UDA methods. For these baselines, we consider the following two *data settings*:

- **In-Domain Generalization** (**IDG**). This represents the standard "generalization" typically considered in supervised learning. In this context, the network is trained and tested on a single labeled domain that, for consistency, we call *target*. Images from

[2]https://pytorch.org/hub/

this domain are randomly partitioned into a training set (80%) and a test set (20%). We regard IDG as the upper bound for SF-UDA.

- **Out-of-Domain Generalization** (**ODG**). In this setting, the pre-trained network is trained on the (labeled) *source domain* and tested on the *target domain*. Given that no adaptation takes place, we consider this as the lower bound for SF-UDA.

Moreover, we consider the following two baseline *training strategies*.

- **Linear Probing** (**LP**). The pre-trained backbone weights are kept frozen and only a linear classifier is trained.

- **Fine-Tuning** (**FT**). We stack a bottleneck and a new linear classifier on top of the backbone and we fine-tune end-to-end the entire model.

As a result, we end up with four baselines: **LP-IDG** and **LP-ODG** for in-domain and out-of-domain generalization with linear probing and **FT-IDG** and **FT-ODG** for in-domain and out-of-domain generalization with fine-tuning. See Tab. 3.2 for a comparative summary of the baselines and the considered SF-UDA settings.

In our study, we adhere to the transductive setting [102]: the final target accuracy is evaluated using the same images that were used during the unsupervised adaptation process, aligning with standard practices in UDA and SF-UDA. We evaluate the performance of various methodologies based on classification accuracy and failure rate on the target domain. In particular, we consider a SF-UDA method to fail an experiment if the final target accuracy is lower than the LP-ODG baseline. Detailed insights on accuracy variations of the considered methods can be found in Sec. 3.8.

3.5.4 Experimental setup

The accompanying code of this benchmark is developed following the modular sequential pipeline of Fig. 3.1, and each component can be changed and individually evaluated. To ensure fast and consistent reproducibility, the codebase features instructions for downloading and formatting datasets, a large variety of supported model architectures and many adaptation algorithms with distributed training implemented. Such features enable a large-scale and in-depth investigation of the SF-UDA setting. For all experiments, we used Python with the PyTorch framework. Based on the computational requirements of different experiments, we allocate from 1 to 16 GPUs; in particular, we empoyed Nvidia V100 16GB and Nvidia A100 80GB GPUs.

Table 3.2 **Evaluation baselines and SF-UDA settings** considered in this work. In the first section on the left the data of supervised training is specified together with the training strategy: optimize only the weight of a classifier or fine-tune also the whole model. The following columns indicate the adaptation data available for the adaptation (second transfer) and the domain employed to test the performance. IDG baselines are considered generalization upper-bounds for the given domain, while ODG baselines are considered lower-bounds fore the given domain pair. *SF-UDA* can be any of the adaptation methods considered.

Task	**First Transfer (supervised)**		**Second Transfer**	**Test**
	Domain	**Optimization**	**Domain**	**Domain**
LP-IDG	Target (w/ labels)	Classifier only	None	Target
FT-IDG	Target (w/ labels)	Backbone+Classifier	None	Target
LP-ODG	Source (w/ labels)	Classifier only	None	Target
FT-ODG	Source (w/ labels)	Backbone+Classifier	None	Target
SF-UDA	Source (w/ labels)	Classifier only	Target (w/o labels)	Target
FT-SF-UDA	Source (w/ labels)	Backbone+Classifier	Target (w/o labels)	Target

3.6 Analysis of SF-UDA methods

In this section we experiment with SCA, SHOT, NRC, AAD, and PCSR, as motivated in section 3.4.2, highlighting both the strengths and weaknesses of each method across different settings. Our evaluation centers on the following research questions.

Reproducibility. Can the selected SF-UDA methods be consistently reproduced in more general experimental settings with respect to those presented in their original evaluations?

Distributed training. SF-UDA methods are often tested on small and medium size datasets using backbones such as ResNet50 (not computationally demanding). However, it is important to study their ability to distribute computation across GPUs, especially when large-scale datasets are used or when modern, more computationally demanding backbones (e.g., Vision Transformers and ConvNext) are adopted.

Generalization across datasets. Can SF-UDA methods generalize to datasets not originally mentioned in their respective papers? This aspect helps identifying potential biases or "dataset cherry-picking" in original reports.

Backbone independence. While the original papers show how these methods can be beneficial on ResNet50 (or ResNet101), can they be used to provide performance gains also to more recent and better performing architectures (like Vision Transformers)?

Hyperparameter sensitivity. How sensitive are these methods to hyperparameters? As, in SF-UDA, target domain labels are absent during adaptation, hyperparameter selection becomes challenging. This may lead to fundamental issues like data leakage (selecting hyperparameters based on target test accuracy after trial and error), which can misrepresent performance metrics. Assessing robustness to hyperparameter changes can shed light on a method's real-world applicability.

3.6.1 Reproducibility

In this work, we look for methods maintaining consistent performance across diverse settings and robust to variations in the random seed and software version. To verify this, we set up a unique testing environment, distinct from those used in the original papers code. We conduct the experiments using five different random seeds, a departure from the prevalent practice in UDA and SF-UDA literature, which typically reports results from a single run. As evidenced in Tab. 3.3, most methods demonstrate robustness to our testing environment, though AAD and NRC, on average, perform marginally below the cited results. Importantly, the standard deviation from our findings casts doubt on the widespread practise of reporting single run performance on *benchmark tables*. Using these results to rank methods is questionable, as the disparities between methods are often smaller than the standard deviations we observed. Thus, relying solely on results from a single run fails to provide a significant evaluation. SCA is not included in the table, since we are the first presenting and studying it as a stand-alone SF-UDA method.

3.6.2 Distributed Training

While distributed training enables computational scalability, its potential remains largely unexplored in SF-UDA research. This can be attributed to the prevalent practice of benchmarking methods using architectures such as ResNet50 and ResNet101, which are not computational demanding according to current standards. However, as we discuss in section 3.7, the choice of architecture and pre-training strategy can profoundly influence outcomes. In fact, moving from ResNet50 to recent architectures, such as ViT or ConvNext, might offer

Table 3.3 Accuracy comparison between paper results and our findings on *Office31* and *Office-Home* datasets. We report the mean accuracy and the standard deviation of 5 runs.

	OFFICE31		**OFFICE-HOME**	
	Reported	**Ours**	**Reported**	**Ours**
AAD	89.9	88.8±0.5	72.7	71.4±0.3
NRC	89.4	88.4±0.6	72.2	70.8±0.1
SHOT	88.6	88.5±0.3	71.8	72.1±0.2
PCSR	89.5	89.4±0.5	72.8	72.7±0.3

significant enhancements to target accuracy. This underscores the importance of effective distributed training.

SCA is not affected by the distribution of the computation. Indeed, adapting the classifier only with K-Means (i.e., no Network optimization) facilitates the distribution. Feature extraction can be done with a batch size of one, while the K-Means stages can be optimized across multiple GPUs. Instead, most SF-UDA methods (including all the the others considered in this work) incorporate a diversity term [57] in their objective, whose computation may require the entirety of the target dataset at once [56]. In practice, the diversity term is often approximated using the current batch. When this batch is split across multiple GPUs and the gradients are averaged (a common practice in distributed training), it often results in a less accurate approximation than if the entire batch was processed in a centralized way. Hence, a performance degradation may happen when SF-UDA methods training is distributed.

We empirically analyze this aspect in Tab. 3.4, showing the methods performance with the standard global batch size of 64 distributed across 1, 2, 4, 8, and 16 GPUs. The tables show different results for different methods. Specifically, the diversity terms in SHOT and PCSR exhibit robustness, allowing for efficient parallelization. Conversely, the diversity terms in AAD and NRC are more sensitive, necessitating larger batch sizes for effective performance. Smaller batch sizes can, at times, entirely undermine their efficacy, especially with ResNet50 where the performance drop is up to 55%. We observe the same behaviour across all datasets.

3.6.3 Dataset and architecture independence

Datasets frequently employed in SF-UDA experimental analyses typically fall within small to medium sizes or they contain just few domains. This can lead to involuntary data leakage and unreliable performance evaluations of the algorithms. For instance, iterative design

Table 3.4 Evaluation of SF-UDA methods with ResNet50 and ViT-Large on **Office31**, **Office-Home**, **Adaptiope** and **Image-CLEF** datasets (for each experiment mean and standard deviation are evaluated over 5 runs). The *global batch size* is fixed to 64. Different rows represent different distributed setting (#GPUs × local batch size).

		ResNet50					ViT-Large				
		SCA	**AAD**	**NRC**	**SHOT**	**PCSR**	**SCA**	**AAD**	**NRC**	**SHOT**	**PCSR**
OFFICE 31	**1x64**		88.8 ± 0.5	88.4 ± 0.6	88.5 ± 0.3	89.4 ± 0.5		94.4 ± 0.5	94.9 ± 0.2	94.3 ± 0.4	94.1 ± 1.8
	2x32		86.8 ± 0.7	87.4 ± 0.5	88.5 ± 0.6	89.4 ± 0.5		94.2 ± 0.5	93.4 ± 1.2	94.2 ± 0.3	94.3 ± 0.7
	4x16	84.5 ± 0.4	83.0 ± 0.9	84.4 ± 0.4	88.4 ± 0.4	89.3 ± 0.4	94.9 ± 0.2	93.0 ± 0.9	90.1 ± 3.0	94.4 ± 0.2	94.7 ± 0.3
	8x8		76.8 ± 1.1	77.1 ± 0.3	87.6 ± 0.2	89.1 ± 0.5		91.8 ± 0.5	75.5 ± 0.8	94.5 ± 0.3	93.9 ± 1.7
	16x4		57.9 ± 3.0	62.4 ± 1.5	86.8 ± 0.4	88.1 ± 0.7		86.4 ± 5.6	52.9 ± 1.3	94.4 ± 0.2	94.0 ± 1.7
OFFICE-HOME	**1x64**		71.4 ± 0.3	70.8 ± 0.1	72.1 ± 0.2	72.7 ± 0.3		80.9 ± 4.4	81.0 ± 2.4	88.0 ± 0.9	85.1 ± 3.9
	2x32		67.4 ± 0.3	67.7 ± 0.2	71.9 ± 0.2	72.3 ± 0.6		80.8 ± 5.3	77.8 ± 5.4	86.5 ± 3.1	84.6 ± 2.6
	4x16	66.1 ± 0.1	60.5 ± 0.5	51.9 ± 0.4	71.7 ± 0.2	72.6 ± 0.2	87.2 ± 0.2	75.0 ± 7.2	68.7 ± 2.2	87.7 ± 0.7	85.3 ± 3.2
	8x8		49.1 ± 0.4	22.4 ± 0.5	71.1 ± 0.2	72.0 ± 0.2		76.0 ± 5.6	56.4 ± 2.4	86.3 ± 3.6	81.9 ± 5.8
	16x4		22.8 ± 0.7	31.7 ± 0.6	68.9 ± 0.3	70.2 ± 0.1		70.2 ± 12.2	46.9 ± 1.6	86.5 ± 1.3	85.9 ± 3.8
ADAPTIOPE	**1x64**		59.9 ± 1.5	64.9 ± 0.6	65.0 ± 0.9	71.8 ± 0.9		49.2 ± 6.1	72.6 ± 9.1	84.6 ± 8.0	89.2 ± 6.8
	2x32		46.6 ± 1.2	26.6 ± 0.3	62.2 ± 1.0	70.8 ± 0.9		49.2 ± 4.0	65.6 ± 1.8	86.3 ± 6.7	92.3 ± 0.7
	4x16	42.1 ± 0.7	32.3 ± 0.8	10.1 ± 0.6	59.5 ± 1.0	69.6 ± 0.9	86.4 ± 0.2	38.6 ± 4.0	55.9 ± 3.8	88.3 ± 1.4	87.3 ± 6.9
	8x8		21.6 ± 1.2	10.2 ± 0.5	55.9 ± 1.6	67.9 ± 0.9		35.0 ± 4.1	51.5 ± 3.4	87.1 ± 1.4	76.4 ± 21.1
	16x4		4.9 ± 1.2	18.9 ± 0.7	51.9 ± 2.3	66.0 ± 1.2		24.2 ± 9.1	42.8 ± 7.4	84.9 ± 6.9	82.4 ± 11.0
IMAGECLEF	**1x64**		83.2 ± 0.2	83.3 ± 0.1	82.9 ± 0.2	83.0 ± 0.3		87.8 ± 1.1	88.4 ± 0.2	87.8 ± 0.3	87.7 ± 0.2
	2x32		82.8 ± 0.2	82.9 ± 0.2	82.7 ± 0.2	82.8 ± 0.3		88.2 ± 0.2	88.4 ± 0.1	87.6 ± 0.5	86.2 ± 3.2
	4x16	81.1 ± 0.2	82.5 ± 0.4	82.7 ± 0.2	82.7 ± 0.1	82.7 ± 0.1	87.5 ± 0.0	87.1 ± 1.7	88.0 ± 0.2	87.6 ± 0.4	87.4 ± 0.6
	8x8		79.8 ± 1.6	81.4 ± 0.2	82.7 ± 0.2	82.5 ± 0.1		87.3 ± 0.8	87.2 ± 0.2	87.5 ± 0.3	86.6 ± 1.9
	16x4		67.3 ± 2.0	77.1 ± 0.7	81.8 ± 0.1	81.5 ± 0.2		86.2 ± 0.9	73.8 ± 2.1	87.4 ± 0.2	87.2 ± 0.4

choices and hyperparameter tuning, guided by observations of target accuracy (trial and error procedure), can inadvertently cause an algorithm to specialize excessively to a particular benchmark dataset with the given backbone. This can mislead researchers into perceiving an algorithm as universally effective when it might be excessively tailored (“overfitted") to specific datasets. Furthermore the architecture choice can influence SF-UDA performance and, even though ResNets are the standard choice in experimental evaluations, it is crucial to discern whether the methods are robust to backbone changes.

To address these questions, our experimental framework includes two datasets not commonly featured in SF-UDA papers (i.e. Adaptiope and Image-CLEF) together with two datasets that, instead, are normally considered (i.e., Office31 and Office-Home). Moreover, we use ViT-Large to evaluate the methods beyond their well-studied experimental settings with the ResNet backbones. The selection of ViT-Large is driven by these factors: it offers a distinct inductive bias compared to ResNets, characterized by its structure as a Vision Transformer incorporating Self-Attention and Layer Normalization, unlike the Convolutional Neural Network with Batch Normalization Layers seen in ResNets. Additionally, ViT has demonstrated enhanced generalization capabilities in ImageNet and various other Computer Vision tasks. Note that, the aim of this experiment is to test the robustness of the different SF-UDA methods while a comprehensive study on the impact of the backbone and pre-training choices is reported in Sec. 3.7.

We report our results in Tab. 3.4 and Tab 3.8. In summary, SHOT and PCSR demonstrate robustness across different architectures and datasets. NRC exhibits competitive performance with different datasets when ResNet50 is employed but it shows sub-optimal performance with ViT. Moreover, AAD performance is negatively affected with changes in either the dataset, architecture, or both. It is worth noting that while a higher accuracy is attainable with ViT, all methods display larger standard deviations, due to possible failures or sub-optimal adaptation. Finally, SCA shows strong performance when ViT is used, while it obtains sub-optimal performance with ResNet50. As we will see in Sec. 3.8 this is due to the failure in some domain-pairs caused by Batch Normalization Layers: since SCA aligns just the classifier without optimizing the backbone weights, large differences in the BN statistics can cause the algorithm to fail in some scenarios.

Remark. The under-performance of NRC and AAD on ViT-Large, even falling below the FT-ODG baseline (i.e., fine-tuning without any adaptation), despite ViT’s superior inductive bias and stronger out-of-distribution generalization (as indicated in Table 3.8), suggests that while these methods might be effective on architectures different from ResNets, they likely require modifications to be compatible with more advanced architectures like Vision Transformers.

Table 3.5 Hyperparamter test for AAD with ResNet50 (mean accuracy on Office31, Office Home, Adaptiope and Image-Clef datasets).

AAD	K=3	K=5
$\beta = 0$	75.70	73.16
$\beta = 0.75$	76.34	73.28
$\beta = 1$	76.20	73.03
$\beta = 2$	75.36	71.83
$\beta = 5$	73.28	69.52

Table 3.6 Hyperparamter test for NRC on ResNet50 (mean accuracy on Office31, Office Home, Adaptiope and Image-Clef datasets).

NRC	K=2	K=3	K=4	K=5
KK=2	75.37	77.03	77.39	77.28
KK=3	76.87	77.45	77.19	76.56
KK=4	77.31	77.31	76.47	75.52
KK=5	77.40	76.78	75.57	74.33

3.6.4 Hyperparameter Sensitivity

The focus of this work is on hyperparameter-free SF-UDA methods. The reason is that in practical situations, determining the optimal hyperparameters can be challenging without data leakage. While unsupervised hyperparameter selection techniques might represent a solution [103], their practical effectiveness is yet to be studied, especially concerning their resilience against variations in architecture and dataset. Parameters such as the number of training epochs and the learning rate are considered non-problematic, as they can be adjusted observing how the unsupervised objective is optimized during adaptation. Indeed, most SF-UDA methods adopt by default the same learning rate, learning rate schedule, weight decay, optimizer, etc.

While SCA is hyperparameter-free, methods like SHOT and PCSR present some hyperparameters, described in their original papers, that are never changed and we treat them as universal constants. Even if these parameters have been probably derived for specific datasets, their successful use in different settings confirms the low sensitivity of those methods to hyperparameters variations. In contrast, AAD and NRC come with dataset-specific hyperparameters. Our analysis focuses on K and β for AAD (the number of considered neighbours and a loss scheduling parameter, respectively) and K and KK for NRC (the number of neighbours and the number of second order neighbours, respectively). For details on these hyperparameters, we recommend consulting the original publications and code.

Tab. 3.5 and 3.6 present average accuracy values across four datasets (i.e., Office31, Office-Home, Adaptiope and Image-CLEF). These tables consider a range of hyperparameters as specified in their respective papers. We observe that AAD's average performance lags behind that of NRC, which exhibits performance more in line with SHOT and PCSR (with average accuracy of 76.34 and 77.45, respectively, on ResNet50). A notable challenge with NRC arises when considering its high accuracy on the VisDA dataset with ResNet101 (achieving a macro-averaged accuracy of 85.9) using $K = 5$ and $KK = 5$. This setting, as demonstrated in our tables, do not yield good performance on the datasets included in our experiments.

Table 3.7 Results summary of the methods evaluation performed in Sec. 3.6 based on Office31, Office-Home, Adaptiope and Image-CLEF datasets and ResNet50 and ViT-Large Backbones.

Methods	**SCA**	**SHOT**	**NRC**	**AAD**	**PCSR**
Reprod.	✓	✓	✓	✓	✓
Parallelism	✓	✓	✗	✗	✓
Dataset Flex.	✓	✓	✓	✗	✓
Backbone Flex.	✗	✓	✗	✗	✓
Hparam-free	✓	✓	✗	✗	✓

Table 3.8 Average accuracy of different methods on Office31, Office-Home, Image-Clef, and Adaptiope. Each experiment (defined by a domain-pair of a dataset) is given equal importance in the average computation. The first row includes the baseline of Fine-Tuning without any adaptation. For each architecture best results are in **bold**, while other high and comparable results are underlined.

	ResNet50	**ViT-Large**
FT-ODG	65.52	85.95
SHOT	<u>77.26</u>	<u>88.40</u>
NRC	<u>77.45</u>	84.28
AAD	76.34	80.16
PCSR	**78.77**	<u>88.12</u>
SCA	71.74	**88.42**

3.6.5 Final Remarks

Results discussed in previous sections brought insights on the reproducibility, possibility of distributed training, dataset, backbone and hyperparameter variations robustness on the

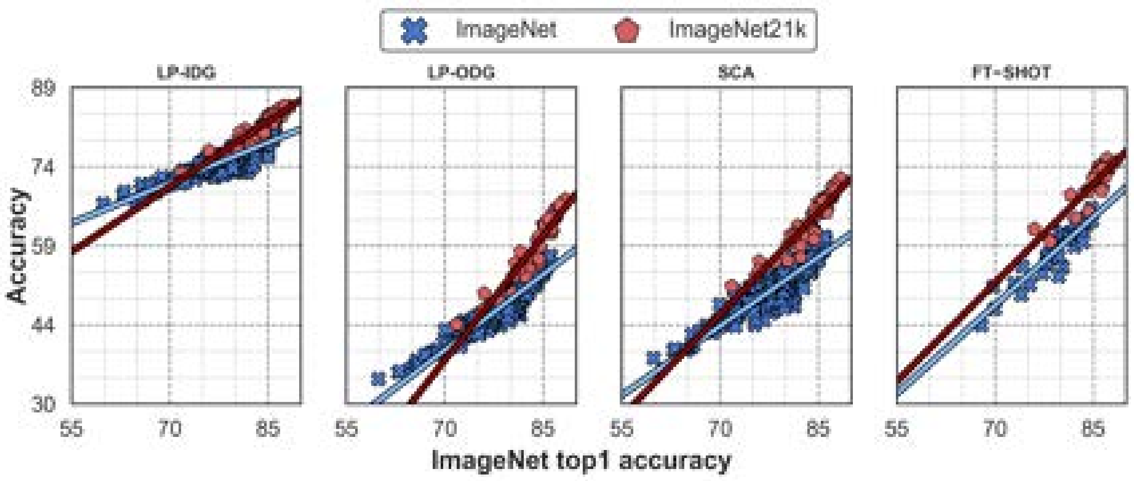


Figure 3.2 Left to right: LP-IDG accuracy (upper bound) averaged over 23 domains, LP-ODG (lower bound), SCA, and FT-SHOT accuracy (averaged over 74 domain pairs). Each marker indicates an architecture, with the x-axis denoting the ImageNet top1 accuracy. Markers color and shape signify the respective pre-training datasets.

considered SF-UDA methods. We summarize our findings in Tab. 3.7 and 3.8. In the next sections we chose to focus on SCA and SHOT. We exclude AAD and NRC due to previously discussed limitations. Although PCSR performs similarly to SHOT and shares many design choices, we exclude it. Despite some accuracy improvements in specific benchmarks, we gave importance to the intrinsic simplicity of SHOT for our studies. Finally, we also consider the complementary of SHOT and SCA: while SHOT, during adaptation, modifies the weights of the backbone keeping the classifier fixed, SCA only adapts the classifier without tuning the backbone weights. The issues of SCA with ResNet50 are discussed and analyzed in Sec. 3.8, where we will see that they are caused by BN layers.

3.7 Impact of pre-training and backbone selection

This section aims to analyze the impact of strategic choices during pre-training on the final SF-UDA accuracy. To this purpose, we start with a comprehensive study considering over 500 backbones and the ImageNet and ImageNet21K datasets, to evaluate the importance of the backbone and dataset choices during the pre-training (Sec. 3.7.1). Note that, we narrow down the analysis to a subset of 59 models for more computationally intensive evaluations, i.e., when fine-tuning (FT) is involved. Finally, we study some Self-Supervised methods for pre-training and their effects on the SF-UDA (Sec. 3.7.2).

3.7.1 Statistical analysis

Fig. 3.2 reports the found correlation between the ImageNet top-1 accuracy and the final accuracy on the task at hand of the two considered SF-UDA methods (SCA and FT-SHOT) and of their upper and lower bounds (LP-IDG and LP-ODG, respectively). Each different point represents a different backbone and we compare models trained on ImageNet (blue) with those trained on ImageNet21K (red). The data shows high correlation among the variables: models that achieve higher top-1 accuracy on ImageNet tend to generalize better across other tasks. While this connection had been previously noted for UDA [76], our findings further stress its significance in the context of ODG and SF-UDA. However, ImageNet top-1 accuracy, though indicative, is not the sole predictor of performance. A distinct phenomenon was observed with models pre-trained on the larger ImageNet21k dataset and subsequently fine-tuned on ImageNet. These models not only enhanced their ImageNet performance but also exhibited an additional improvement in tasks like SF-UDA, not solely explained by their top-1 ImageNet accuracy gain.

To quantify the significance of observed correlations, we present a statistical analysis, identifying two predominant variables that can affect the final performance: the ImageNet top-1 accuracy and the choice of pre-training dataset. If we ignore the effect of the pre-training, the relationship for each backbone $\mathcal{B}$ can be formalized as:

$$\text{accuracy}(\mathcal{B}) = m \cdot \text{top1}(\mathcal{B}) + q + \varepsilon, \tag{3.2}$$

where m and q represent task-specific coefficients, and ε is the residual variance not captured by the model. The choice of pre-training dataset is an important additional explanatory variable. In our experiments, the pre-training dataset, denoted as $\text{pretrain}(\mathcal{B})$, could be either 0 for ImageNet or 1 for ImageNet21k. To account for the influence of the pre-training dataset in our model, we propose a multi-linear model:

$$\text{accuracy}(\mathcal{B}) = [m + \Delta m \cdot \text{pretrain}(\mathcal{B})] \cdot \text{top1}(\mathcal{B}) + q + \Delta q \cdot \text{pretrain}(\mathcal{B}) + \varepsilon, \tag{3.3}$$

where Δm and Δq are the additional coefficients. A comparison of both statistical models is illustrated in Tab. 3.9, where it is possible to see a large increase of adjusted $\bar{R}^2$ values when considering multi-linear models. Finally, Tab. 3.9 shows that our findings are valid in all considered settings, with and without the fine-tuning.

Table 3.9 Comparison of adjusted $\bar{R}^2$ values between linear (considering top-1 accuracy only) and multi-linear models (with both top-1 accuracy and pre-training). Results are shown for the upper bounds (LP-IDG and FT-IDG), the lower bounds (LP-ODG and FT-ODG) and for the considered SF-UDA methods, SCA and SHOT, with and without fine-tuning (FT).

	LP-IDG	**LP-ODG**	**SCA**	**SHOT**
Linear	0.736	0.810	0.731	0.792
Multi-Linear	0.851	0.935	0.902	0.890

	FT-IDG	**FT-ODG**	**FT-SCA**	**FT-SHOT**
Linear	0.803	0.698	0.668	0.838
Multi-Linear	0.878	0.822	0.792	0.932

Table 3.10 ImageNet vs ImageNet21k pre-training for different backbone sizes. The accuracy reported are the average of 74 domain shifts.

	LP-ODG		**SCA**		**FT-SHOT**	
Backbone (params)	**IN**	**IN21k**	**IN**	**IN21k**	**IN**	**IN21k**
VGG19 (143.7M)	45.2	**47.6**	49.4	**53.1**	**56.0**	55.7
ResNet50 (25.6M)	47.2	**51.3**	49.8	**58.2**	55.9	**62.0**
W-ResNet50 (68.9M)	50.5	**52.5**	53.3	**58.2**	62.1	**64.0**
DenseNet161 (28.7M)	48.0	**52.2**	52.7	**58.2**	61.6	**65.3**
ConvNext B (88.6M)	54.4	**65.2**	58.4	**68.5**	65.1	**72.7**

Influence of backbones size. Some studies, like [25], suggest that larger models, due to their expansive parameter set, would primarily benefit from the pre-training on larger datasets like ImageNet21k, while small models can obtain no or negative benefit from very large scale pre-trainings. However, our investigation challenges this evidence in the context of SF-UDA. In Tab. 3.10, we demonstrate that not only larger architectures, but even small and medium size backbones can significantly benefit from an ImageNet21k pre-training.

3.7.2 Self-Supervised pre-training

Recently, self-supervised pre-training has gained popularity due to its demonstrated effectiveness in many studies [36]. In this section, we examine some state-of-the-art Self-Supervised Learning (SSL) pre-training methods within the SF-UDA context, comparing them to traditional supervised pre-training. Our focus is on two widely-used architectures: ResNet50 and

ViT-Base. We choose these architectures since they are frequently evaluated in SSL studies, and their pre-trained weights are available online.

SSL with ResNet50. For the analysis with ResNet50, we evaluate DINO [104], MOCO v1 [37], and MOCO v2 [38] approaches. As shown in Tab. 3.11, when considering FT-ODG and FT-SCA, both MOCO v2 and supervised pre-training on ImageNet (1k) exhibit similar results, while other methods perform worse. However, when applying SHOT after the fine-tuning (FT-SHOT), supervised pre-training clearly outperforms all SSL pre-training methods, including MOCO v2. A detailed analysis of why FT-SCA does not perform as well as FT-SHOT in this context is presented in Sec. 3.8. Thus, for ResNet50, supervised pre-training is more effective than the considered SSL methods.

Table 3.11 Target accuracy (%) using different SSL initialization weights of ResNet50 after performing the following experiments: fine-tuning on the source domain (FT-ODG), FT-SCA and FT-SHOT.

Exp.	**Pre-train**	MO31	Visda	O.Home	Adapt.	I-CLEF	D.Net	**Avg**
FT-ODG	DINO	42.7	45.6	45.7	30.9	71.3	19.7	42.7
	MOCO v1	38.5	44.6	41.7	28.6	66.2	20.1	40.0
	MOCO v2	54.7	**54.2**	55.1	39.5	**76.9**	**25.5**	**51.0**
	Supervised (IN1k)	**56.4**	49.6	**55.5**	**43.3**	75.9	21.6	50.4
FT-SCA	DINO	50.8	54.5	49.8	35.9	75.0	21.9	48.0
	MOCO v1	49.5	52.3	43.9	34.3	70.4	23.9	45.7
	MOCO v2	**63.4**	**56.9**	55.8	45.3	**79.5**	**28.0**	54.8
	Supervised (IN1k)	62.7	55.5	**59.4**	**49.6**	79.3	23.6	**55.0**
FT-SHOT	DINO	71.8	67.3	58.0	55.4	77.5	24.1	59.0
	MOCO v1	56.6	58.3	52.9	37.5	71.7	19.2	49.4
	MOCO v2	73.2	65.7	66.9	51.9	81.0	26.2	60.8
	Supervised (IN1k)	**80.6**	**71.6**	**68.9**	**66.6**	**81.6**	**27.3**	**66.1**

SSL with ViT. For ViT-Base backbone we consider Masked Autoencoder (MAE) [39], DINO [104] and DINO v2 [42] as SSL pre-training strategies. The results are presented in Tab. 3.12.

Despite recognizing the effectiveness of MAE as SSL pre-training technique, our results show that, without a subsequent supervised ImageNet fine-tuning, it does not provide a strong foundation for SF-UDA and ODG tasks. Nevertheless, DINO shows stronger performance

Table 3.12 Target accuracy (%) for VIT-Base we report results for the lower bounds (LP-ODG and FT-ODG) and for the considered SF-UDA methods (SCA and SHOT) with and without fine-tuning (FT). DINO v2 and MAE need distinct FT strategies (indicated with [†]) and their backbones were frozen during SHOT adaptation (indicated with an [*]). Best results for each dataset and setting are highlighted in **bold**, with the top overall results underlined.

Experiment	**Pre-train**	O31	MO31	Visda	O.Home
LP-ODG	MAE	44.2	30.0	45.6	37.8
	DINO	81.5	71.6	65.7	59.8
	DINO v2	**89.4**	**89.8**	**72.6**	**79.3**
	Supervised (IN21k)	**89.4**	85.1	69.4	78.2
FT-ODG	MAE†	52.9	39.0	53.4	51.0
	DINO	80.7	72.4	65.4	63.6
	DINO v2†	**89.8**	**87.7**	58.5	76.8
	Supervised (IN21k)	89.7	86.9	**77.9**	**79.9**
SCA	MAE	16.4	15.6	17.6	5.4
	DINO	86.6	84.4	74.2	61.1
	DINO v2	**94.1**	<u>**96.5**</u>	**81.7**	**83.0**
	Supervised (IN21k)	92.7	91.8	69.1	78.5
FT-SCA	MAE†	34.9	28.7	49.8	32.0
	DINO	85.2	81.1	73.6	67.9
	DINO v2†	**93.8**	**95.0**	62.8	79.8
	Supervised (IN21k)	92.7	91.9	**82.6**	**82.5**
SHOT	MAE*	8.4	12.4	62.2	18.1
	DINO	89.4	86.5	79.2	68.3
	DINO v2*	94.0	96.8	80.6	84.7
	Supervised (IN21k)	<u>**95.3**</u>	**93.9**	<u>**88.5**</u>	<u>**86.5**</u>
FT-SHOT	MAE†*	38.0	39.0	67.1	55.7
	DINO	87.5	75.8	56.9	74.2
	DINO v2†*	93.3	95.5	68.4	81.2
	Supervised (IN21k)	**93.7**	**93.9**	**87.5**	**84.4**

than MAE on all the examined tasks, even if it is still outperformed by supervised pre-training in every setting.

Finally, for DINO v2 evaluation, we faced two main challenges. The first is related to its fine-tuning on the source domain, caused by gradients of high magnitude in the initial training phase that hardly alter the weights of the model reducing the inherent benefits of DINO v2 initialization. To mitigate this, we proposed to adopt a two-phase fine-tuning strategy: (i) we freeze the backbone, training only the newly introduced bottleneck and classifier on the source domain and (ii) we continue the whole network fine-tuning using gradient clipping.

The second challenge is related to the model adaptation to the target domain with SHOT. SHOT adaptation fails with DINO v2 when the whole network is optimized with the unsupervised objective. In our pipeline, to solve this problem we completely freeze DINO v2 weights and we optimize just the bottleneck with SHOT objective. Applying FT-SHOT on DINO v2 without our approach leads to bad performance on all datasets. For instance, on Office31, the accuracy is 5.6% without our method, while it is 93.3% when we apply it. Analogous patterns are evident on Modern Office31 (improving from 7.0% to 95.5%), VisDA (from 14.9% to 68.4%), and Office-Home (from 3.5% to 81.4%). In general, if our approach is used on DINO v2, it obtains similar performance to supervised pre-training, even surpassing it in some settings.

Note that, we applied the same strategies presented for DINO v2 to MAE. This yielded marginal improvements but even with these enhancements, the performance remains sub-optimal, rendering MAE less competitive than the other methods.

3.8 Analysis of the impact of fine-tuning

In SF-UDA, the backbones are typically fine-tuned on the source domain during the first transfer. Given the importance of this step, we analyze its impact in the overall SF-UDA performance. For this, we study the accuracy variation happening when fine-tuning (FT) is applied with respect to cases without it. We report the results of this analysis in Fig. 3.3. Each marker represents a backbone, sorted on the X-axis by their top-1 accuracy on ImageNet (just a subset of backbones considered is shown for visualization constrains). The arrows represent the accuracy variation happening for that model. In the top row, we report the accuracy difference between LP-IDG and FT-IDG (left) and LP-ODG and FT-ODG (right), being our upper and lower bounds, respectively. The plots in the other rows represent the accuracy difference between the lower bound (LP-ODG) and the considered SF-UDA methods (i.e.,

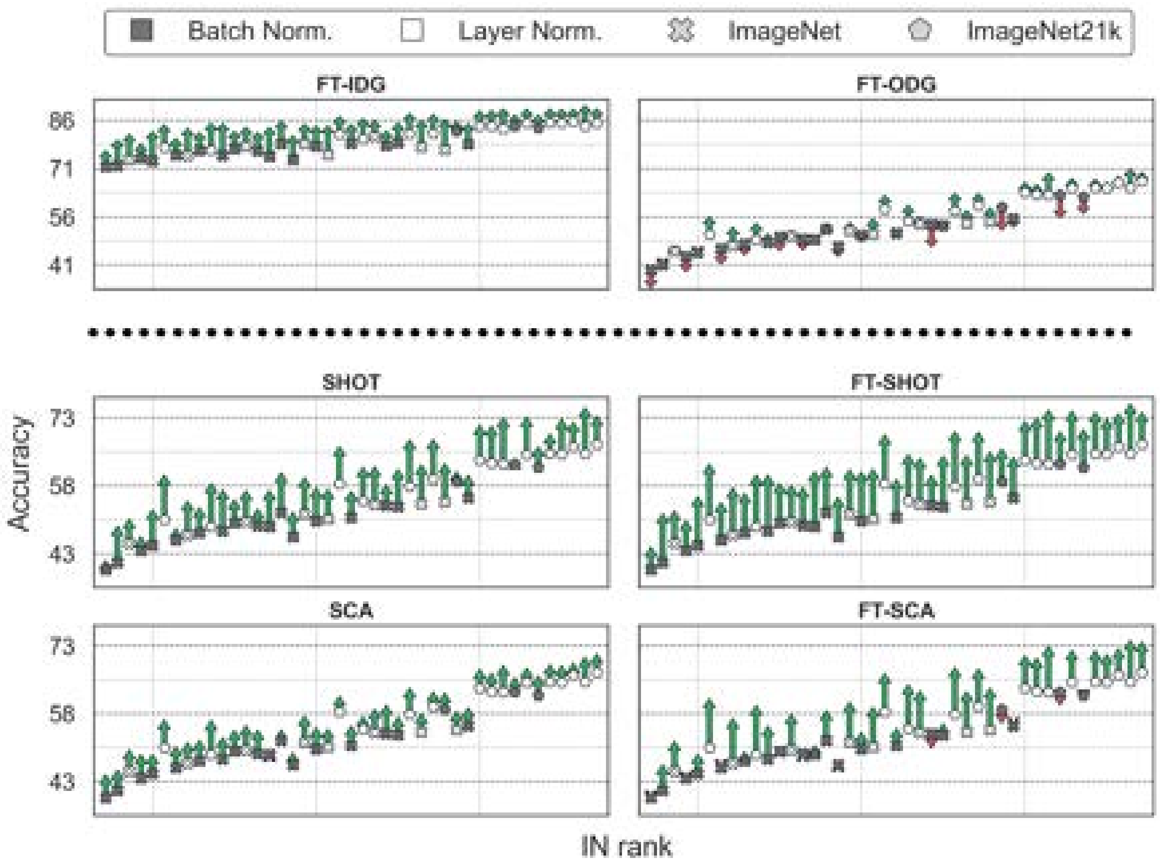


Figure 3.3 Impact on final accuracy of source fine-tuning. Each marker is a backbone. Variations on the final accuracy is reported as arrows. In the top row, we report the accuracy difference between LP-IDG and FT-IDG (left) and LP-ODG and FT-ODG (right), being our upper and lower bounds, respectively. All other plots represent accuracy difference between LP-ODG with the considered SF-UDA method (i.e., SHOT, FT-SHOT, SCA and FT-SCA). Models with BN and LN are represented with dark and white markers, respectively.

Table 3.13 Average accuracy variation and failure rates with respect to LP-ODG (lower bound). These are averaged over 74 domain shifts. Each column represents a normalization layer type. The parenthesis shows the number of considered backbones.

	Δ Accuracy			**Failure Rate**		
Tasks	**All**	**BN (32)**	**LN (27)**	**All**	**BN (32)**	**LN (27)**
FT-ODG	0.46 ±3.71	−2.21 ±2.75	3.62 ±1.57	40.82 ±16.14	51.52 ±11.42	28.13 ±10.85
SCA	4.21 ±1.39	4.08 ±1.55	4.36 ±1.19	21.05 ±11.92	27.24 ±11.64	13.71 ±7.25
SHOT	6.97 ±2.24	6.21 ±2.39	7.88 ±1.68	11.18 ±8.75	14.82 ±9.32	6.86 ±5.61
FT-SCA	4.90 ±3.86	2.16 ±2.94	8.14 ±1.66	19.08 ±14.57	29.77 ±11.05	6.41 ±4.52
FT-SHOT	9.66 ±1.83	9.68 ±1.96	9.63 ±1.69	4.47 ±3.61	5.83 ±3.77	2.85 ±2.69

SHOT, FT-SHOT, SCA and FT-SCA). Thus, comparing the two columns in the second and third rows, we can visualize the impact of the fine-tuning for the SF-UDA methods.
As shown in Fig. 3.3 and reported in Table 3.13 (left side), SHOT combined with fine-tuning (FT-SHOT) consistently enhances performance across different backbones. Instead, for SCA, the benefit of the fine-tuning (FT-SCA) is particularly evident for LN models with an average increase of 8.14%, while the boost for BN models is modest (2.16%). A similar pattern is identified for the lower bound FT-ODG, where the accuracy increases of 0.46% across all architectures for all the experiments. However, backbones with Batch Normalization (BN) and with Layer Normalization (LN) show different behaviors. For the first ones, the final accuracy declines of 2.21%, while for the second ones, it improves of 3.62%. Finally, the upper bound (FT-IDG) does not show the same pattern. This is due to the fact that the FT impact in the same domain is beneficial for both BN and LN models, as already well-established [76].

3.8.1 Layer or Batch Normalization?

In this section, we analyze the failure rate for each backbone after the fine-tuning on the source. We define a failure to be a performance degradation, with respect to the LP-ODG lower bound, when a method is applied.
Starting from the insights previously presented, in this section we compare backbones with LN and backbones with BN. We report our results in Tab. 3.13. In particular, from the right side of the table, it is evident that models with BN consistently show higher failure rates compared to those with LN. For instance, for FT-SCA, models with BN have a failure rate of 29.77% while models with LN have 6.41% .

Interestingly, even without source fine-tuning, SHOT, which adjusts BN statistics and backbone's weights during adaptation, registers a BN failure rate of 14.82%. In contrast, SCA achieves a failure rate of 27.24% for BN models. However, LN models failure rates significantly decrease for both SHOT and SCA, registering 6.86% and 13.71%, respectively. This large improvement underscores BN's instability in the presence of domain shifts, in contrast with LN models robustness.

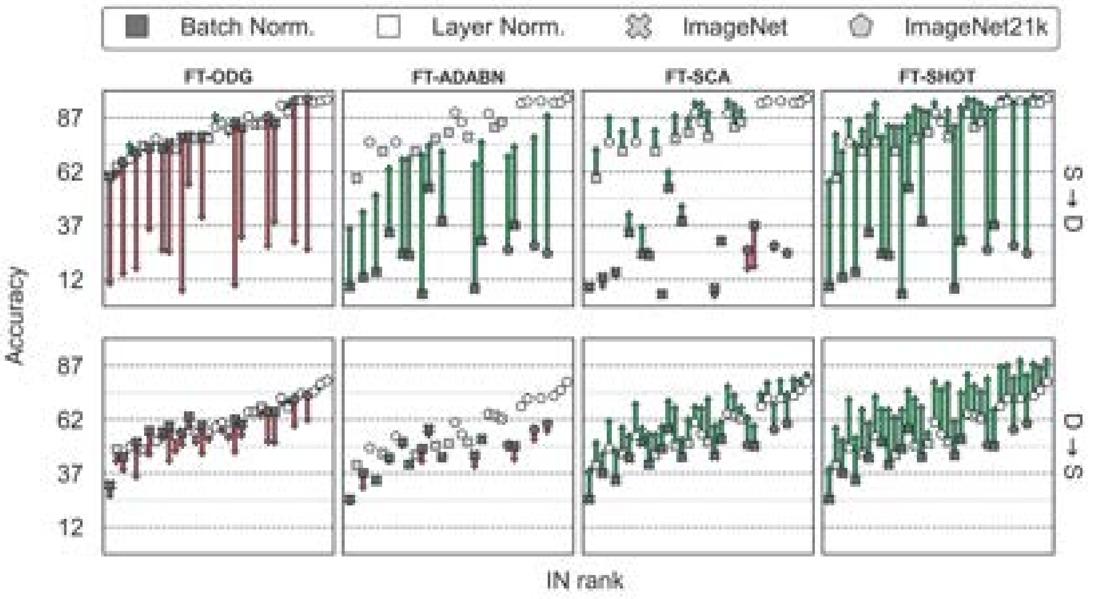


Figure 3.4 Example of fine-tuning impact for Modern Office31 dataset (Synthetic → DSLR in the first row and DSLR → Synthetic in the second row). Each marker is a backbone. Fine-tuning effect on the final accuracy is reported as arrows. In the first column, we report the accuracy variation between LP-ODG and FT-ODG (lower bound). Then, we report the accuracy difference with FT-ODG for the case when ADABN is applied (second column) and for FT-SCA and FT-SHOT (third and fourth columns).

However, the literature presents techniques to mitigate BN limitations, such as ADABN [105]. Our findings reveal that, while these techniques can, in some cases, retrieve a portion of the lost accuracy, they remain inconsistent. We show our findings in Fig. 3.4, reporting results for the *DSLR* and *Synthetic* domain pair in Modern Office 31 dataset, taken as an example. ADABN occasionally falls short of complete recovery. Additionally, as depicted in the figure's second row, in some situations (i.e. particular source-target pairs), ADABN amplifies the degradation, raising questions about the scenarios in which such techniques might be beneficial.
Therefore, while various techniques can sometimes enhance BN models performance, their unpredictability across different domains often makes them less appealing for SF-UDA. LN

models, on the contrary, demonstrate better stability reducing failure rates, making them a more reasonable choice for domain adaptation tasks.

3.9 Discussion

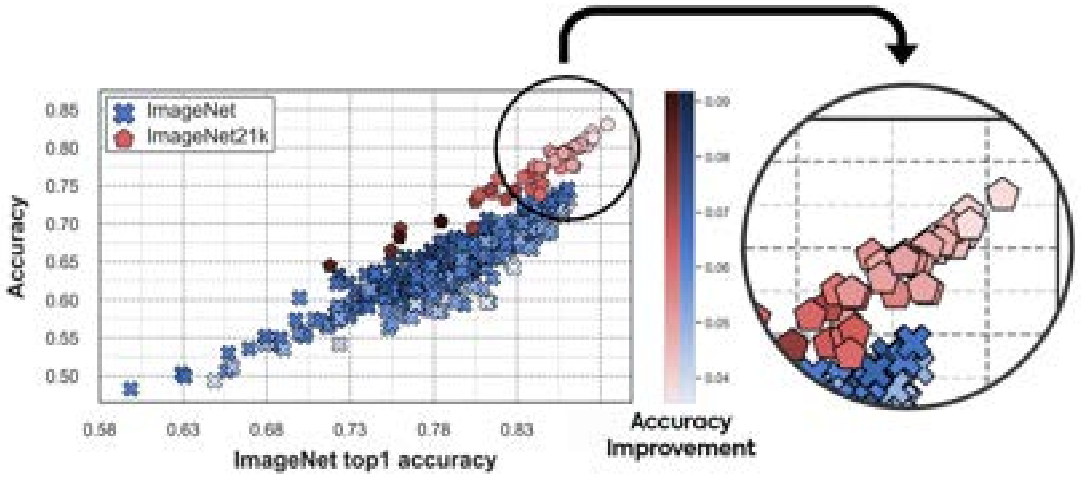


Figure 3.5 We report the relation between the ImageNet top-1 accuracy and target accuracy of SCA across 74 domain pairs, for over 500 different backbones. Different colors of the markers represent different pre-training dataset, while the shade intensity signifies the SCA accuracy improvement with respect to the lower bound LP-ODG.

Our work proposes an open-source benchmark framework that enables the execution of a systematic large-scale experimental analysis of SF-UDA. We believe that this framework might put the basis for future work, being a useful tool for structured SF-UDA evaluation. Moreover, we present a detailed study of current SF-UDA state-of-the-art, highlighting relevant insights. Firstly, in Sec. 3.6, we pointed to limitations in prevalent and common benchmark standards. We study the reproducibility, possibility to execute distributed training, dependence from datasets and backbones and hyperparameters sensitivity of SF-UDA methods. In this perspective, we believe that the method design and hyperparameter selection should not be driven by test accuracy after trial and error experiments. Otherwise, SF-UDA methods effectiveness and generality might be potentially overestimated.

Then, in Sec. 3.7, we analyze the impact of pre-training strategy and backbone choice for SF-UDA. Architectures that perform well on ImageNet, also present improved SF-UDA accuracy and pre-training on larger datasets, such as ImageNet21k, amplifies this outcome. Although certain Self-Supervised Learning methods might not match supervised training's efficacy in general, techniques, like DINO v2, obtain competitive results (Sec. 3.7.2). However, an important question is still open: as the architectural backbones get larger and with improved ImageNet performance, is there a diminishing return on the advantages conferred by SF-UDA? Preliminary observations, captured in Fig. 3.5, suggest that this might be the case, since best ImageNet21k models often see less enhancements provided by SF-UDA. Finally, in Sec. 3.8, we provide an analysis on the advantages and disadvantage of fine-tuning the backbone weights on the source domain during the first transfer. While it is often beneficial, there are scenarios, particularly involving backbones with Batch Normalization layers, where it can lead to hard failures.

Limitations. Although extensive, our study did not encompass the entire spectrum of SF-UDA methodologies. We chose a subset, emphasizing those that we found reproducible and applicable to different datasets and backbones. Some important aspects like the evaluation of normalization free networks [106] or with alternative normalization layers, the pre-training on different, larger, datasets and unsupervised hyperparameter selection for SF-UDA remain avenues for future exploration.

3.10 Conclusion

In our investigation, we have shown many critical aspects of the SF-UDA landscape, offering an analysis on current practices and directions for forthcoming research. In particular we have highlighted the weaknesses and strengths of some SF-UDA methodologies, the role of the first transfer, the importance of architectural choices and pre-training strategies, that undeniably have a significant impact on SF-UDA outcomes. In essence, this work represents a pivotal reference for researchers in SF-UDA and practitioners aiming to apply SF-UDA in real-world applications.

Chapter 4

Trust And Balance: A novel SF-UDA approach

4.1 Context

Most of the new SF-UDA methods presented in the literature tend to lean towards complexity. In this contribution we introduce a novel and simple SF-UDA approach for image classification marked by two key contributions: Few Trusted Samples Pseudo-labeling (FTSP) and Temperature Scaled Adaptive Loss (TSAL). FTSP employs a limited subset of trusted samples from the target data to construct a classifier to infer pseudo-labels for the entire domain, showing simplicity and improved accuracy. Simultaneously, TSAL, designed with a unique dual temperature scheduling, adeptly balance diversity, discriminability, and the incorporation of pseudo-labels in the unsupervised adaptation objective. Our methodology, that we name Trust And Balance (TAB) adaptation, is rigorously evaluated on standard datasets like Office31 and Office-Home, and on less common benchmarks such as ImageCLEF-DA and Adaptiope, employing both ResNet50 and ViT-Large architectures. Our results compare favorably with, and in most cases surpass, contemporary state-of-the-art techniques, underscoring the effectiveness of our methodology in the SF-UDA landscape.

4.2 Introduction

DNNs have made significant advancements in computer vision tasks, including image classification, detection, and semantic segmentation [107]. However, they often face challenges when the distribution of the test data, or the *target domain*, differs from the training data,

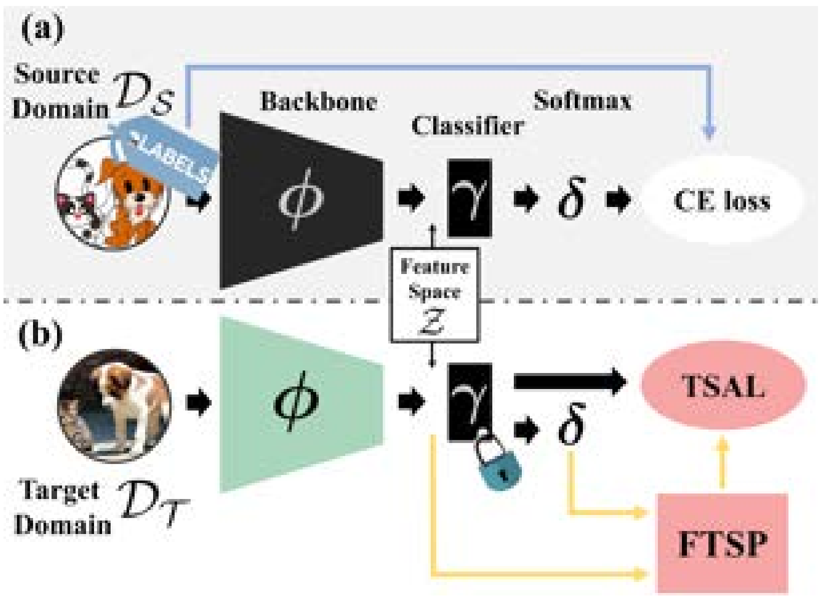


Figure 4.1 **SF-UDA Pipeline (our contributions in red).** In the upper section **(a)**, the source model is trained on the source domain through a conventional supervised method (indicated by the blue arrow). In the lower section **(b)**, adaptation to the target domain is conducted using our proposed pseudo-labeling method (FTSP) and objective function (TSAL), as shown by the yellow arrows. Consistent with the method of [56], the classifier γ remains unchanged during the adaptation phase, while the backbone (in green) is adapted.

known as the *source domain*. Such domain discrepancies, stemming from environmental changes, device variations or different image styles, limit the effectiveness of DNNs in real-world applications.

Unsupervised Domain Adaptation (UDA) aims to apply knowledge from a labeled source domain to an unlabeled target domain [17]. While conventional UDA strategies demand access to both domains to mitigate the domain shift, there exist scenarios, especially in sensitive sectors like healthcare, where accessing the source data is constrained due to privacy or storage issues. This led to the advent of Source-Free Unsupervised Domain Adaptation (SF-UDA) in image classification [56], building upon ideas from Hypothesis Transfer Learning [55]. Essentially, SF-UDA leverages a model trained on the source, without a direct access to source data. Contemporary advances in SF-UDA encompass methodologies like entropy-minimization, generative modeling, class prototyping, self-training and many others [75]. As we will see in Sec. 4.3, our approach shares some parallels with pseudo-label denoising and entropy-minimization techniques.

In particular, we present a novel pseudo-labeling paradigm, **Few Trusted Samples Pseudo-labeling (FTSP)**, which accentuates simplicity and the quality of pseudo-labels.

Unlike conventional, more complex, pseudo-labeling techniques, our method centers on creating a training set using a restricted subset of *trusted* samples (i.e. with high likelihood to be correctly labeled by the source classifier) from the target domain (limited up to 3 samples per class). While our framework is agnostic to the choice of classifier, for simplicity, we adopted Multinomial Logistic Regression (MLR) in our main experiments and we present an ablation study with different classifiers in App. A.2. Despite potential overfitting concerns with MLR on this limited dataset, it empirically demonstrates proficient generalization capabilities across the broader target domain, effectively inferring high-quality pseudo-labels. We also propose a pseudo-label refinement phase, including a deletion mechanism based on classifier uncertainty and a pseudo-label completion step via *Label Spreading* [108].

The analysis of Yang et al. [57] emphasized that most SF-UDA methods revolve around an objective involving two core components: a *diversity term* for prediction variability and a *discriminability term* to enhance target samples differentiation. Inspired by Information Maximization objective of SHOT [56] we propose the **Temperature Scaled Adaptive Loss (TSAL)**: a novel and advanced objective to guide the adaptation process. In particular TSAL is specifically designed to use a dual temperature scheduling to dinamically balance the discriminability, diversity and the incorporation of pseudo-labels and their significance throughout the whole adaptation phase, showing improved performance in SF-UDA.

In summary, our key contributions are:

- **Few-Trusted Samples Pseudo-labeling (FTSP)**: an effective pseudo-labeling technique involving the training of a classifier employing a curated very-limited subset of *trusted samples* from the target domain. We further propose incorporating pseudo-label deletion and completion steps (with Label Spreading) for additional refinement.
- **Temperature Scaled Adaptive Loss (TSAL)**: our advanced balance strategy to effectively calibrate the equilibrium between diversity, discriminability, and pseudo-label significance in the objective, resulting in enhanced SF-UDA results.
- **Robust Benchmarking and Analysis**: our method undergoes rigorous evaluations on standard datasets like Office31 and Office-Home, and on emerging benchmarks such as ImageCLEF-DA and Adaptiope, using both ResNet50 and ViT-Large. Beyond traditional single-seed evaluations, we present a multi-seed robustness analysis (5 seeds) and recreate some selected state-of-the-art techniques for a thorough comparative insight.

The structure of this paper is as follows: Sec. 4.3 provides an overview of pertinent literature. The SF-UDA setting is presented in Sec. 4.4. The proposed methodology is delineated in Sec. 4.5. Experimental procedures and results are detailed in Sec. 4.6. Concluding remarks are presented in Sec. 4.7. Additional details and ablation studies are presented in the supplementary material, while the code will be released at https://github.com/andreamaracani/TAB_SFUDA.

4.3 Related Work

Unsupervised Domain Adaptation (UDA). UDA aims to adapt models from a source domain (where labels are available) to an unlabeled target domain. The foundational principles of UDA are rooted in the theoretical works by Mansour et al. [44] and Ben-David et al. [45]. Early methods include sample selection [109] and feature projection [110], followed by techniques designed to adapt DNNs, such as adversarial training [47], Maximum Mean Discrepancy [48], Bi-directional Matching [50], Margin Disparity Discrepancy [49] and many others [17]. Though initially centered on image classification, UDA has expanded to include tasks like object detection [52] and semantic segmentation [53]. A notable challenge in UDA is the need for simultaneous access to both source and target data during training, which may be a nuisance or even impracticable in some contexts, e.g. due to intellectual property or privacy issues.

Source-Free UDA (SF-UDA). As a subdomain of UDA, SF-UDA negates the direct access to source domain data during adaptation. The field gained traction following Liang et al. [56]. Thereafter, a multitude of methods emerged, achieving interesting results on common UDA benchmarks. Noteworthy SF-UDA techniques include generative model-driven methods like 3C-GAN [60], algorithms based on the feature space's neighborhood structure (e.g., NRC [58] and AAD [57]), methods transferring Batch Normalization statistics [111, 71], strategies constructing surrogate source domains during adaptation [112, 72], techniques utilizing knowledge distillation within a mean-teacher [113] paradigm [73, 114, 82], and those incorporating Contrastive learning [114, 74, 115]. A comprehensive review of contemporary SF-UDA approaches can be found in [75].

Learning with pseudo-labels. SF-UDA methods often necessitates the creation of target pseudo-labels for improved training. However, the potential presence of errors in these pseudo-labels parallels training with noisy labels. Numerous methods aim to contrast the potential noise-fitting caused by these inaccuracies. Notable approaches encompass the utilization of reliable labels through co-teaching dual networks [116], Negative Learning

(NL) implementation [117], and the adoption of noise-resistant loss functions [118]. In the SF-UDA setting, Zhang et al. [119] advanced a technique that refines noise rate estimation and emphasizes early-stage sample retention. Luo et al. [120] presented a method to rectify pseudo-label errors using negative learning, tailored for semantic segmentation. Yang et al. [121] fused pseudo-label denoising with self-supervised knowledge distillation. Litrico et al. [122] integrated insights from nearest neighbors and entropy-based uncertainty estimation, further augmented by a temporal queue mechanism and self-learning methodologies.

Many contemporary techniques lean toward complexity, but our methodology distinguishes itself through its efficiency and effectiveness. While we might optionally utilize the well-established Label Spreading to alleviate label noise, the essence of our method lies in generating inherently accurate pseudo-labels with a classifier trained on a meticulously selected set of a very limited number of trusted target samples. Additionally, the harmonious integration of discriminability and diversity in our TSAL objective further enhances the method's robustness against pseudo-label noise. As detailed in Sec. 4.6, our approach consistently aligns with or even surpasses state-of-the-art (SOTA) performance across benchmarks, asserting the robustness of our loss function to pseudo-label noise.

4.4 Problem Definition

Before presenting our proposed approach, we set the foundation for SF-UDA in the context of image classification. Let $\mathcal{X} \in \mathbb{R}^{H\times W\times 3}$ denote the space of RGB images with height H and width W. The label space, covering C distinct categories, is represented by $\mathcal{Y} = \{c\}_{c=1}^{C}$. We postulate two distinct distributions over $\mathcal{X} \times \mathcal{Y}$: the source domain $\mathcal{D}_S$ and the target domain $\mathcal{D}_T$. We consider the Close-set assumption: the label space remains consistent between these domains, guaranteeing that each category possesses a non-zero probability of manifestation in both.

Consider $f_\theta : \mathcal{X} \to \mathcal{Y}$, a function parametrized by θ, which maps each input image to its associated label in $\mathcal{Y}$. The main objective in both UDA and SF-UDA is to identify this function along with its optimal parameters, ensuring accurate target domain predictions. While DNNs are the prevalent choice for this function, data restrictions depend on the specific adaptation setting. Specifically, the SF-UDA framework consists of **two stages** (see Fig. 4.1):

1. A labeled dataset from the source distribution, represented as $D_S = \{(\mathbf{x}_S^{(i)}, y_S^{(i)})\}_{i=1}^{N} \sim \mathcal{D}_S^N$, is employed to determine the function parameters θ_S such that the function performs optimally on the source domain.

2. The source dataset becomes inaccessible, though the parameters θ_S remain available together with an unlabeled dataset from the target domain (marginal) distribution, represented as $D_T = \{\mathbf{x}_T^{(i)}\}_{i=1}^M \sim \mathcal{D}_T^M(\mathcal{X})$. This is employed to adjust the model parameters to θ_T, with the goal of obtaining an improved performance on the target domain.

4.4.1 Architecture

In alignment with the conventions established in earlier studies, the function f (we omit parameters θ for notation simplicity) is articulated as an composition of multiple functions, as illustrated in Fig. 4.1:

$$f(\mathbf{x}) \mapsto \arg\max_{c \in \mathcal{Y}} \{\delta(\gamma(\phi(\mathbf{x})))_c\} = \hat{y} \tag{4.1}$$

where function $\phi : \mathcal{X} \to \mathcal{Z} \subset \mathbb{R}^d$ is the backbone and it operates as a feature extractor mapping images into the d-dimensional feature space $\mathcal{Z}$. $\gamma : \mathcal{Z} \to \mathbb{R}^C$ is as a classifier that maps features into the C-dimensional space of logits. Lastly, $\delta : \mathbb{R}^C \to \Delta^{C-1}$ denotes the Softmax function, which translates logits into the $C-1$ simplex that signifies classification probabilities for each class. The ultimate prediction class $\hat{y}$ is extracted using the $\arg\max$ operation on these probability values.

4.5 Method

A fundamental guiding principle in our method design is ensuring backbone independence. While specialized architectural modifications, such as adapting Batch Normalization layers, freezing some specific layers, or introducing specific additional modules, can offer advantages in certain benchmarks (e.g., with ResNet50), we deliberately avoid them. This decision is rooted in our understanding that in scenarios extending beyond typical benchmarks, more advanced models could be employed within the SF-UDA framework. Therefore, our ambition is to devise a universally applicable solution. We present an overview of our proposed method and in the next sections we will give a detailed description of the algorithm.
Stage 1: source fine-tuning. We initiate training using a pre-trained (e.g., on ImageNet) feature extractor and we adopt an end-to-end fine-tuning approach, adjusting the backbone's weights with the labeled source dataset in alignment with previous SF-UDA algorithms, leveraging insights from [123] that highlight the benefits of such source fine-tuning.

Stage 2: target adaptation. When unlabeled target data becomes available the model undergoes unsupervised self-training. At the beginning of each epoch, pseudo-labels for the entire target domain are reassessed using our FTSP methodology (Sec. 4.5.1), then our TSAL objective is minimized to balance diversity, discriminability and pseudo-labels significance with a dual temperature scaling (Sec. 4.5.2). During this adaptation phase, the weights of the backbone $\phi(\cdot)$ are updated while the classifier, $\gamma(\cdot)$, remains unchanged in consistency with [56].

4.5.1 Pseudo-labeling through few trusted samples

Our algorithm's development was heavily influenced by a clear insight: the selection of an extremely limited number of high-quality target domain samples can lay a foundation for constructing a classifier that surpasses the performance of the original source classifier γ. This perspective deviates from traditional methods that often rely on large sample sizes or intricate techniques. By identifying a restricted set of K **trusted samples** (TS) for each class (i.e. samples that are very likely to be correctly classified), we build a classifier using the combined dataset with $K \times C$ samples, providing a strong basis for predictions across the target domain.

Trusted samples training set. In our quest for a simplified methodology, for each class $c \in \mathcal{Y}$, we choose the K feature samples with the top predicted probabilities according to the source classifier $\delta(\gamma(\cdot))$:

$$\text{TS}_c := \{\mathbf{z}_c^{(1)}, \dots, \mathbf{z}_c^{(K)}\} = \mathit{argmax}^K_{\mathbf{z} \in \mathcal{Z}_T} \{\delta(\gamma(\mathbf{z}))_c\} \tag{4.2}$$

To clarify, the notation argmax^K denotes the function returning the K arguments with the greatest values. $\mathcal{Z}_T$ represents the set of all target features (evaluated with backbone ϕ), and $\delta(\gamma(\mathbf{z}))_c$ indicates the predicted probability of the feature $\mathbf{z}$ being categorized into class c by the source classifier. Repeating this for each class produces a *few-trusted-samples* training set with known labels (that are likely to be correct).

Trusted samples classifier. Considering this dataset, we first normalize the features vectors and then we train a simple classifier from scratch, subsequently deploying it to infer pseudo-labels for the entire target domain. Our method is independent of the chosen classifier; however, in our experiments, we consistently employed by default Multinomial Logistic Regression (MLR). While the results highlight the efficacy of MLR (as elaborated in Sec. 4.6.2), we acknowledge its potential limitations. To further explore these aspects, a post hoc ablation study is provided in the appendix considering different classifiers and hyperparameters (Sec. A.2). This study indicates that Linear Discriminant Analysis (LDA) may offer additional advantages to our approach.

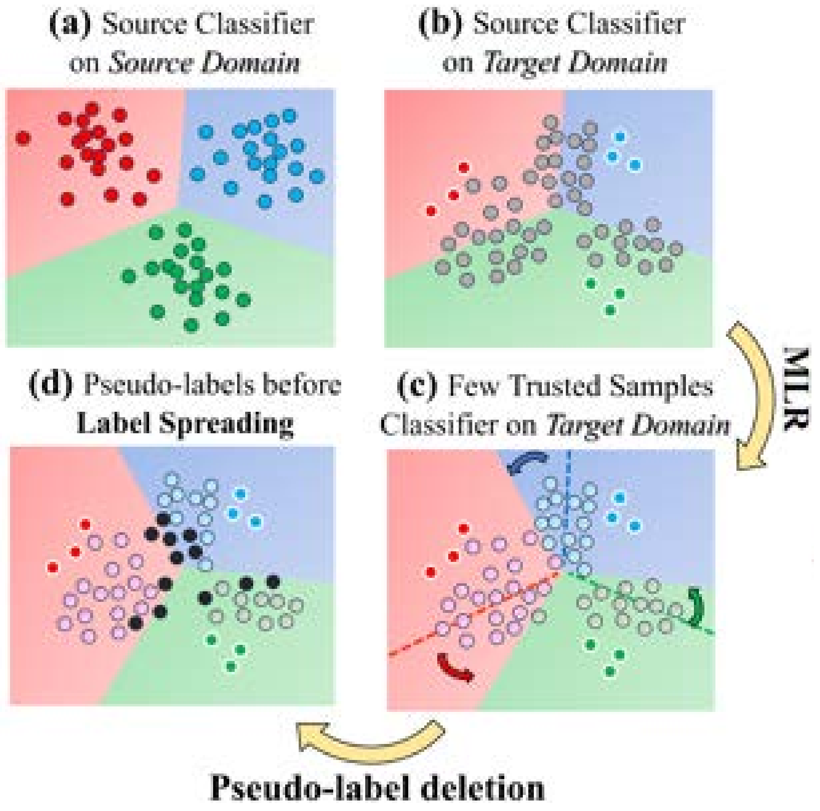

(a) Source Classifier on Source Domain
(b) Source Classifier on Target Domain
MLR
(c) Few Trusted Samples Classifier on Target Domain
(d) Pseudo-labels before Label Spreading
Pseudo-label deletion

Figure 4.2 **Pseudo-labeling with Few Trusted Samples**: **(a)** Classifier trained on the source domain demonstrates robust performance within the same domain. **(b)** The same classifier underperforms on the unlabeled target domain (represented by grey dots). A minimal set of trusted target samples (indicated by colored dots with white outlines) is selected, being deemed most likely to be correctly classified. **(c)** Using these few trusted samples, a Multinomial Logistic Regression (MLR) classifier is trained, leading to decision boundaries that align more closely with the target domain and subsequently providing pseudo-labels for the entire target domain. **(d)** A fraction of uncertain pseudo-labels is eliminated prior to the application of Label Spreading, finalizing the Few Trusted Samples Pseudo-labeling (FTSP) process.

Pseudo-label refinement. For improved pseudo-label quality, we propose an additional refinement phase in our algorithm. Specifically, we introduce a pseudo-label deletion step, which entails removing a certain percentage (per class) of the least certain pseudo-labels, based on the MLR classifier output probabilities. This is followed by a pseudo-label completion step using the established semi-supervised learning method of Label Spreading [108]. These procedures are useful to reassess and enhance the overall label consistency.

Their advantages are explored in the ablation study in Sec. 4.6.3 and in the supplementary material (Sec. A.2).

We refer to our distinctive pseudo-labeling technique as **Few Trusted Samples Pseudo-labeling (FTSP)**, illustrated in Fig. 4.2.

4.5.2 Temperature scaled loss for adaptive training

Intuition and motivation. The analysis in [57] shows that most SF-UDA objectives can be delineated into two primary goals. The first is to enhance prediction distinction (discriminability term, *dis*), and the second is to diversify these predictions (diversity term, *div*).

$$loss_{\text{SF-UDA}} = dis + div \tag{4.3}$$

In particular, the Information Maximization (IM) objective of SHOT [56] has exhibited consistent performance across diverse architectures and datasets [123]. Such robustness is not universally observed among all state-of-the-art UDA and SF-UDA methods, as highlighted in Kim et al.'s study [63]. Nevertheless, we have observed some limitations and weaknesses of the SHOT objective:

- The discriminability component uses both the model's current predictions (to minimize the entropy) and some pseudo-labels pre-computed through clustering. But as training moves forward, the model becomes more sure of its own predictions. This increased (over-)confidence can make it harder to adjust predictions based on pseudo-labels (see Fig. 4.3a).
- The diversity term is represented by the negative entropy of the average output probabilities. Early in the adaptation process, under common domain shifts, the network often lacks confidence in its predictions, resulting in an already high average entropy (so a very low negative entropy). This can cause the discriminability term to be overly emphasized in the initial stages.

To address these challenges we design a new objective that incorporates a dual-temperature scaling approach to balance discriminability, diversitity and pseudo-label significance across the whole adaptation process. At the start, we use a standard temperature value ($= 1$) for the discriminability term. As the model becomes more confident, we increase the temperature (> 1) to moderate the model's growing certainty. Conversely, for the diversity term, we adopt a lower temperature (< 1) to refine predictions early in training, transitioning to a standard temperature ($= 1$) towards the training's conclusion. We now present the designed objective encapsulating these insights.

Temperature Scaled objective. For a batch comprising B images, denoted as $\mathcal{B} = \{\mathbf{x}^{(i)}\}_{i=1}^{B}$, the model discerns the output logit vectors $\hat{\mathcal{L}} = \{\hat{\mathbf{l}}^{(i)}\}_{i=1}^{B}$. Further, one-hot pseudo-labels are computed through our FTSP to yield $\hat{\mathcal{Y}} = \{\hat{\mathbf{y}}^{(i)}\}_{i=1}^{B}$. An initial step entails the softening of pseudo-labels to mitigate erroneous pseudo-label impacts, resulting in the smooth pseudo-label set $\hat{\mathcal{Y}}_S$. Specifically, we utilize conventional label smoothing with a default factor S of 0.1:

$$\hat{\mathbf{y}}_S^{(i)} = \hat{\mathbf{y}}^{(i)} \cdot (1 - S) + \mathbf{1}_C \cdot S/C \tag{4.4}$$

where $\mathbf{1}_C$ is a C-dimensional vector containing 1s. We now construct an objective target distribution for a generic target sample $\mathbf{x}_i$ as a mixture of temperature scaled predicted probability and the pseudo-label:

$$\hat{\mathbf{q}}^{(i)}(t) = \delta\left(\frac{\hat{\mathbf{l}}^{(i)}}{\tau_{\text{dis}}(t)}\right) + \alpha \cdot \hat{\mathbf{y}}_S^{(i)} \tag{4.5}$$

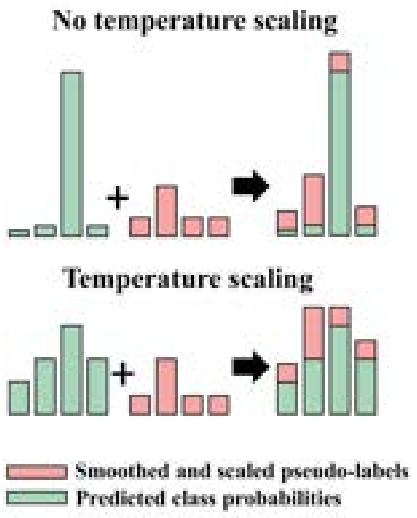
Where α is a constant set to 0.3 and $\tau_{\text{dis}}(\cdot)$ is the temperature function in order to scale the predicted probabilities and make them softer at the end of the training. The t variable, in our schedule (that we will discuss shortly) is an integer corresponding to the number of epoch. The integration of both the predictions of the network (self-regularization) and the pseudo-labels in the objective distribution enables the competition between model predictions and pseudo-labels computed with FTSP. The loss's **discriminability term** is hence:

$$dis(\hat{\mathcal{L}}, \hat{\mathcal{Y}}_S; t) := \frac{1}{B}\sum_{i=1}^{B} H(\hat{\mathbf{q}}^{(i)}(t), \delta(\hat{\mathbf{l}}^{(i)})) \tag{4.6}$$

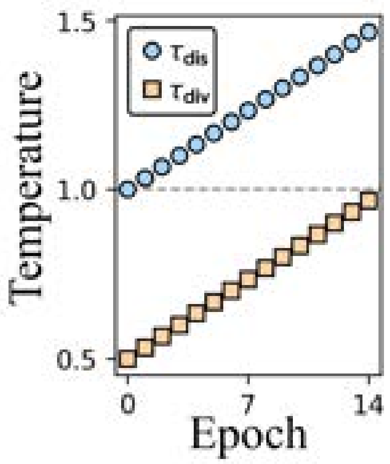
where $H(\cdot,\cdot)$ is the Cross-entropy function. For diversity, a temperature scaled variant of [56] is employed. Let define the output average (scaled) probability as:

$$\bar{\mathbf{p}}(t) := \frac{1}{B}\sum_{i=1}^{B} \delta\left(\frac{\hat{\mathbf{l}}^{(i)}}{\tau_{\text{div}}(t)}\right) \tag{4.7}$$

Where $\tau_{\text{div}}(\cdot)$ is the second temperature schedule function in order to scale the predicted probabilities and make them sharpen at the beginning of the adaptation procedure. Then the **diversity term** is:


No temperature scaling
Temperature scaling
Smoothed and scaled pseudo-labels
Predicted class probabilities

(a) Temperature scaling in the discrimability term.



(b) Temperature schedules.

Figure 4.3 **(a)** the temperature scaling in our discrimability term enables a fair competition between model predictions and the pseudo-labels at the end of the training when the network becomes overconfident. **(b)** our schedules $\tau_{\text{dis}}(\cdot)$ and $\tau_{\text{div}}(\cdot)$ respectively for the discrimanbility and diversity term.

$$div(\hat{\mathcal{L}};t) := -H(\bar{\mathbf{p}}(t)) \tag{4.8}$$

where $H(\cdot)$ is the entropy function.

The overall objective, that we refer to as **Temperature Scaled Adaptive Loss (TSAL)** is:

$$loss_{\text{TSAL}}(\hat{\mathcal{L}},\hat{\mathcal{Y}}_S;t) := dis(\hat{\mathcal{L}},\hat{\mathcal{Y}}_S;t) + div(\hat{\mathcal{L}};t) \tag{4.9}$$

Temperature Scaling schedule. As shown in Fig. 4.3b the functions $\tau_{\text{dis}}(\cdot)$ and $\tau_{\text{div}}(\cdot)$ undergo adjustments every epoch. While $\tau_{\text{dis}}(\cdot)$ gradually enhances prediction softness, $\tau_{\text{div}}(\cdot)$ initially sharpens predictions, only to soften them towards the training's closure. The essence of these functions is rooted in the preliminary motivations. Specifically, $\tau_{\text{dis}}(\cdot)$ transitions linearly from 1 to 1.5, whereas $\tau_{\text{div}}(\cdot)$ moves from 0.5 to 1 (with 1 signifying no temperature modulation).

4.6 Experimental results

We evaluate the proposed approach, that we name **Trust And Balance (TAB)**, and compare it with SOTA methods for SF-UDA on image classification.

4.6.1 Setup

Datasets. For our evaluation, we chose a combination of widely-recognized datasets (Office31 and Office-Home), as well as datasets that are slightly less prevalent in typical benchmarks (Adaptiope and ImageCLEF-DA). This selection underscores the versatility of our method. **Office-31** [89]: this dataset features 4 110 images and includes three domains: Amazon (A), DSLR (D), and Webcam (W). **Office-Home** [90]: a medium-scale dataset that comprises 15 500 images, partitioned into 65 categories and spread across 4 domains: Art (A), Clip Art (C), Product (P), and Real World (R). **Adaptiope** [93]: a large-scale dataset containing 36 900 images. It is categorized into 123 classes and spans three domains: Product (P), Real Life (R), and Synthetic (S). **ImageCLEF-DA** [92]: a small dataset including 2 400 images. It is divided into 12 classes and 4 domains: Bing (B), Caltech (C), ImageNet (I), and Pascal (P).
Backbones. For a fair comparison with a wide range of other popular SOTA methods we adopted ResNet50 [124] (pre-trained on ImageNet [5]) for our experiments and the typical single-run results. To evaluate the robustness of our approach we further investigate multi-run results (we use 5 seeds in the robustness analysis) and we adopted also a better performing architecture to prove the versatility of our approach, namely ViT-Large [33] (pre-trained on ImageNet21k). To have a comparison we selected 3 popular SOTA SF-UDA methods that we recognize as easily reproducible and we run them through our robustness analysis: SHOT [56], AAD [57] and NRC [58].
Implementation Details. Our method is developed using the PyTorch [125] framework and adheres to the standard guidelines and hyperparameters found in the SF-UDA literature, such as [56], [58], and [57]. We employ the SGD optimizer for training, configured with a momentum of 0.9, weight decay of 10^{-3}, batch size of 64, and an input image dimension of 224×224. The pre-trained backbone is enriched by a newly initialized bottleneck layer that maps features to 256 dimensions and the final classifier. The initial learning rates are set to 10^{-3} for the backbone and ten times higher for both the bottleneck and classifier. These rates then follow exponential scheduling throughout training. Notably, the classifier's weights are frozen during the adaptation phase. Additionally, we incorporate MixUp regularization [51] throughout the training process. For FTSP we use a value of $K = 3$ for ResNet50 and $K = 7$

for ViT-L (accounting for the more precise predictions of this advanced architecture) and a Multinomial Regression Classifier. In the label deletion step we delete, for each class, the 20% of less confident pseudo-labels, and then we apply Label Spreading. Depending on the computational needs of various experiments, we utilized either Nvidia V100 16GB or Nvidia A100 80GB GPUs. Comprehensive details can be found in the appendix (Sec. A.1).

Table 4.1 Performance comparison of various SOTA methods on the **Office31** dataset using both ResNet50 and ViT-Large backbones. Each column represents an experiment SRC→TGT, while the rightmost column provides the average accuracy. The top results are highlighted in **bold**, while the runners-up are underlined. All ViT-L outcomes were independently obtained by us.

Method	A→D	A→W	D→A	D→W	W→A	W→D	**Avg**
ResNet50 [124]	68.9	68.4	62.5	96.7	60.7	99.3	76.1
3C-GAN$_{\text{R50}}$ [60]	92.7	93.7	75.3	98.5	**77.8**	99.8	89.6
BNM-S$_{\text{R50}}$ [126]	93.0	92.9	75.4	98.2	75.0	99.9	89.1
SHOT$_{\text{R50}}$ [56]	94.0	90.1	74.7	98.4	74.3	99.9	88.6
AAD$_{\text{R50}}$ [57]	96.4	92.1	75.0	99.1	76.5	**100.0**	89.9
NRC$_{\text{R50}}$ [58]	96.0	90.8	75.3	99.0	75.0	**100.0**	89.4
DIPE$_{\text{R50}}$ [80]	**96.6**	93.1	75.5	98.4	77.2	99.6	**90.1**
A^{2}Net$_{\text{R50}}$ [59]	94.5	94.0	76.7	**99.2**	76.1	**100.0**	**90.1**
TAB$_{\text{R50}}$	94.4	**94.7**	**76.9**	97.4	76.0	99.8	89.9
ViT-L [33]	91.8	94.1	80.5	98.5	86.7	99.6	91.4
SHOT$_{\text{ViT}}$ [56]	98.2	97.9	82.9	97.2	85.7	**99.8**	93.6
AAD$_{\text{ViT}}$ [57]	98.8	98.5	79.8	99.3	84.4	**99.8**	93.4
NRC$_{\text{ViT}}$ [58]	98.0	98.1	85.9	99.0	**87.0**	**99.8**	94.6
TAB$_{\text{ViT}}$	**100.0**	**98.9**	**86.4**	**99.9**	86.9	**99.8**	**95.3**

4.6.2 Results

Office31. The results for the Office31 benchmark are presented in Table 4.1. Our approach yields results that are competitive with SOTA methods when using the ResNet50 architecture. Additionally, when employing the advanced ViT-L architecture, our method surpasses the performance of the considered techniques, achieving an average accuracy of 95.3%.

Office-Home. As detailed in Table 4.2, our method's outcomes on the Office-Home benchmark are either on par or superior to SOTA methods using ResNet50. Moreover, with the ViT-L architecture, our method outperforms other techniques, achieving an average accuracy of 88.2%.

Table 4.2 Performance comparison of various SOTA methods on the **Office-Home** dataset using both ResNet50 and ViT-Large backbones. Each column represents an experiment SRC→TGT, while the rightmost column provides the average accuracy. The top results are highlighted in **bold**, while the runners-up are $\underline{\text{underlined}}$. All ViT-L outcomes were independently obtained by us.

Method	A→C	A→P	A→R	C→A	C→P	C→R	P→A	P→C	P→R	R→A	R→C	R→P	**Avg**
ResNet50 [124]	46.3	67.5	75.9	59.1	59.9	62.7	58.2	41.8	74.9	67.4	48.2	74.2	61.3
G-SFDA$_{R50}$ [127]	57.9	78.6	81.0	66.7	77.2	77.2	65.6	56.0	82.2	72.0	57.8	83.4	71.3
SHOT$_{R50}$ [56]	57.1	78.1	81.5	68.0	78.2	78.1	67.4	54.9	82.2	73.3	58.8	84.3	71.8
NRC$_{R50}$ [58]	57.7	**80.3**	82.0	68.1	**79.8**	78.6	65.3	56.4	83.0	71.0	58.6	**85.6**	72.2
AAD$_{R50}$ [57]	**59.3**	79.3	$\underline{82.1}$	68.9	**79.8**	$\underline{79.5}$	67.2	$\underline{57.4}$	83.1	72.1	58.5	$\underline{85.4}$	$\underline{72.7}$
ELR(NRC)$_{R50}$ [128]	58.4	78.7	81.5	$\underline{69.2}$	$\underline{79.5}$	79.3	66.3	**58.0**	82.6	$\underline{73.4}$	$\underline{59.8}$	85.1	72.6
DIPE$_{R50}$ [80]	56.5	79.2	80.7	**70.1**	**79.8**	78.8	67.9	55.1	$\underline{83.5}$	**74.1**	59.3	84.8	72.5
A^{2}Net$_{R50}$ [59]	58.4	79.0	**82.4**	67.5	79.3	78.9	$\underline{68.0}$	56.2	82.9	**74.1**	**60.5**	85.0	**72.8**
TAB$_{R50}$	$\underline{58.9}$	$\underline{79.6}$	81.5	68.6	78.0	**79.8**	**69.3**	56.8	**83.7**	73.2	59.5	84.7	**72.8**
ViT-L [33]	75.3	88.5	91.4	85.3	89.7	89.9	$\underline{83.3}$	75.0	91.5	86.8	74.7	$\underline{92.0}$	85.3
SHOT$_{ViT}$ [56]	$\underline{80.9}$	$\underline{92.0}$	$\underline{91.9}$	**89.9**	$\underline{92.2}$	76.1	77.0	**81.9**	**92.1**	$\underline{88.9}$	**82.8**	**93.9**	$\underline{86.6}$
NRC$_{ViT}$ [58]	79.7	91.0	91.8	85.8	89.7	91.7	42.7	$\underline{76.3}$	91.3	85.6	$\underline{82.6}$	88.3	83.0
AAD$_{ViT}$ [57]	66.3	91.2	91.7	89.7	90.5	$\underline{92.2}$	78.3	75.6	91.4	86.4	77.1	**93.9**	85.4
TAB$_{ViT}$	**81.3**	**92.7**	**93.2**	$\underline{89.8}$	**92.9**	**93.4**	**86.9**	76.1	$\underline{91.7}$	**89.4**	80.9	89.7	**88.2**

Adaptiope. Table 4.3 shows the results for this challenging benchmark, including both mean and standard deviation over five runs. On the ResNet50 architecture, our method significantly outperforms other techniques with a notable margin of $+6.9\%$. Furthermore, when utilizing the ViT-L architecture, our method continues to lead, registering an average accuracy of 91.7%.

ImageCLEF-DA. The results are provided in Table 4.4 (mean and std). On this small dataset, our method's performance is consistent with other techniques when implemented on both ResNet50 and ViT architectures. However, it is marginally outpaced by the NRC method, which achieves an average lead of $+0.4\%$ for both architectures.

4.6.3 Analysis

The results highlight that our method, with its inherently simple yet effective pseudo-labeling approach and clear design, achieves performance that is on par with or even surpasses state-of-the-art methods across the examined benchmarks. Significantly, our approach outperforms competitors in 3 out of 4 datasets when using the ViT-Large architecture. It is especially notable that our method exceeds others substantially in the challenging Adaptiope benchmark when employing ResNet50, showcasing its robustness. However, in the less restricted UDA setting (where access to source data is permitted), some methods, such as CAN [48]

Table 4.3 Multi-run (5 seeds) performance comparison of various SOTA methods on the **Adaptiope** dataset using both ResNet50 and ViT-Large backbones. The top results are highlighted in **bold**, while the runners-up are underlined. All results have been obtained by us both for ResNet50 and ViT-L. **Note**: high standard deviations are due to the failure of methods for one or more seeds in the considered experiment.

Method	P→R	P→S	R→P	R→S	S→P	S→R	**Avg**
ResNet50 [124]	67.0 ± 0.6	35.0 ± 1.0	87.6 ± 0.1	30.4 ± 0.9	13.4 ± 1.8	2.7 ± 0.8	39.3 ± 0.5
SHOT$_{R50}$ [56]	$\underline{78.5 \pm 0.2}$	58.8 ± 2.0	91.9 ± 0.2	$\underline{57.4 \pm 1.6}$	58.9 ± 2.0	$\underline{44.6 \pm 2.9}$	$\underline{65.0 \pm 0.9}$
AAD$_{R50}$ [57]	76.7 ± 0.7	53.5 ± 3.5	$\underline{92.1 \pm 0.2}$	48.5 ± 3.3	53.2 ± 3.0	35.1 ± 3.2	59.9 ± 1.5
NRC$_{R50}$ [58]	77.2 ± 0.2	$\underline{60.8 \pm 0.7}$	88.7 ± 0.3	55.0 ± 1.9	$\underline{63.8 \pm 2.2}$	44.0 ± 1.4	64.9 ± 0.6
TAB$_{R50}$	**79.9** ± 0.4	**65.2** ± 2.5	**92.2** ± 0.2	**64.1** ± 1.2	**72.3** ± 0.9	**57.7** ± 1.7	**71.9** ± 0.3
ViT-L [33]	93.5 ± 0.2	68.9 ± 0.4	97.4 ± 0.1	66.2 ± 0.7	93.2 ± 0.2	87.2 ± 0.5	84.4 ± 0.1
SHOT$_{ViT}$ [56]	$\underline{94.5 \pm 0.3}$	**87.6** ± 0.7	96.7 ± 2.6	**86.9** ± 0.4	77.6 ± 42.9	**91.7** ± 1.8	$\underline{89.2 \pm 6.8}$
AAD$_{ViT}$ [57]	23.4 ± 26.6	41.7 ± 23.4	76.7 ± 42.3	47.1 ± 4.9	$\underline{94.0 \pm 1.6}$	12.4 ± 22.4	49.2 ± 6.1
NRC$_{ViT}$ [58]	93.2 ± 0.4	83.3 ± 1.0	$\underline{97.5 \pm 0.2}$	65.7 ± 35.9	77.3 ± 42.4	$\underline{90.5 \pm 2.1}$	84.6 ± 8.0
TAB$_{ViT}$	**94.6** ± 0.3	$\underline{86.2 \pm 0.5}$	**97.6** ± 0.1	$\underline{86.0 \pm 1.2}$	**96.5** ± 0.5	89.1 ± 1.3	**91.7** ± 0.3

Table 4.4 Multi-run (5 seeds) performance comparison of various SOTA methods on the **ImageCLEF-DA** dataset using both ResNet50 and ViT-Large backbones. The top results are highlighted in **bold**, while the runners-up are underlined. All results have been obtained by us both for ResNet50 and ViT-L.

Method	B→C	B→I	B→P	C→B	C→I	C→P	I→B	I→C	I→P	P→B	P→C	P→I	**Avg**
ResNet50 [124]	90.0 ± 3.1	84.6 ± 2.8	68.0 ± 2.5	59.3 ± 1.3	83.9 ± 1.3	69.1 ± 1.8	60.6 ± 0.2	92.8 ± 0.8	75.2 ± 0.5	58.7 ± 1.4	91.1 ± 1.0	88.5 ± 2.2	76.8 ± 0.3
SHOT$_{R50}$ [56]	$\underline{96.6 \pm 0.6}$	$\underline{92.8 \pm 0.6}$	77.7 ± 2.2	65.1 ± 0.5	$\underline{92.8 \pm 0.3}$	78.0 ± 0.6	64.4 ± 0.8	96.5 ± 0.6	78.2 ± 0.6	64.4 ± 0.6	96.1 ± 0.4	92.5 ± 1.0	82.9 ± 0.1
AAD$_{R50}$ [57]	96.5 ± 0.7	92.7 ± 0.6	$\underline{78.0 \pm 1.2}$	$\underline{65.7 \pm 0.7}$	92.6 ± 0.5	77.9 ± 0.7	**65.6** ± 0.6	$\underline{96.6 \pm 0.8}$	$\underline{78.9 \pm 1.4}$	$\underline{64.7 \pm 1.3}$	$\underline{96.2 \pm 0.7}$	**93.2** ± 0.7	$\underline{83.2 \pm 0.2}$
NRC$_{R50}$ [58]	$\underline{96.6 \pm 0.7}$	**93.2** ± 0.3	**78.3** ± 1.5	**65.9** ± 0.8	**93.1** ± 0.5	$\underline{78.4 \pm 0.7}$	$\underline{64.8 \pm 0.7}$	96.4 ± 0.5	**79.0** ± 0.8	**65.2** ± 1.0	96.1 ± 0.7	$\underline{93.0 \pm 0.7}$	**83.3** ± 0.0
TAB$_{R50}$	**96.8** ± 0.6	92.5 ± 0.0	77.8 ± 0.9	65.5 ± 0.9	92.2 ± 0.4	**78.5** ± 0.3	64.5 ± 1.2	**97.0** ± 0.2	78.2 ± 0.3	63.6 ± 0.6	**96.5** ± 0.2	92.0 ± 0.1	82.9 ± 0.3
ViT-L [33]	96.2 ± 0.9	94.9 ± 0.7	78.2 ± 1.3	69.3 ± 1.1	94.6 ± 0.7	78.2 ± 1.2	69.9 ± 1.1	96.2 ± 0.4	81.2 ± 0.9	66.8 ± 0.9	92.6 ± 2.9	97.3 ± 0.8	84.6 ± 0.4
SHOT$_{ViT}$ [56]	$\underline{98.3 \pm 0.2}$	$\underline{97.9 \pm 0.2}$	82.5 ± 0.4	$\underline{71.5 \pm 1.3}$	98.0 ± 0.2	82.7 ± 0.3	72.2 ± 0.9	98.2 ± 0.4	82.8 ± 0.5	72.5 ± 0.9	98.3 ± 0.3	98.3 ± 0.2	87.8 ± 0.3
AAD$_{ViT}$ [57]	95.4 ± 6.9	**98.0** ± 0.2	$\underline{83.1 \pm 0.6}$	70.9 ± 3.5	98.0 ± 0.2	$\underline{82.8 \pm 0.5}$	$\underline{74.2 \pm 1.0}$	$\underline{98.3 \pm 0.5}$	83.4 ± 0.6	**74.3** ± 1.1	96.9 ± 3.5	98.3 ± 0.4	87.8 ± 1.1
NRC$_{ViT}$ [58]	$\underline{98.3 \pm 0.2}$	**98.0** ± 0.2	**83.2** ± 0.4	**74.4** ± 0.5	**98.4** ± 0.3	82.6 ± 0.7	**74.3** ± 0.6	98.2 ± 0.5	**83.6** ± 0.3	$\underline{73.5 \pm 0.8}$	$\underline{98.4 \pm 0.5}$	**98.5** ± 0.4	**88.4** ± 0.2
TAB$_{ViT}$	**98.8** ± 0.1	**98.0** ± 0.3	82.9 ± 0.4	70.6 ± 3.1	$\underline{98.3 \pm 0.2}$	**82.9** ± 0.1	72.7 ± 0.2	**98.8** ± 0.2	$\underline{83.4 \pm 0.5}$	71.8 ± 0.8	**98.7** ± 0.2	$\underline{98.4 \pm 0.2}$	$\underline{88.0 \pm 0.3}$

(achieving 75.2% average accuracy as reported by [93]), can outperform our results in this specific benchmark.

Table 4.5 **Ablation**: the introduction of Pseudo-label Refinement (PR), i.e. Label Deletion + Label Spreading, enhance the performance. The addition of TSAL give an additional boost.

FSPL	PR	TSAL	Office31	Office-Home
✓			89.4	72.0
✓	✓		89.6	72.2
✓		✓	89.6	72.4
✓	✓	✓	**89.9**	**72.8**

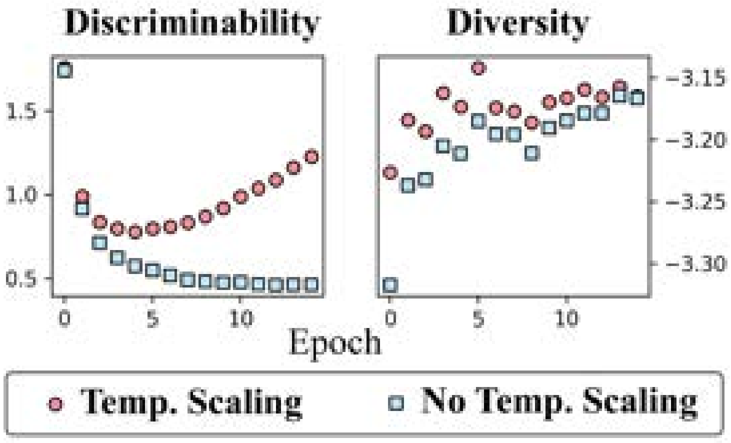


Figure 4.4 Discriminability and Diversity terms of the SF-UDA objective averaged across each training epoch with and without temperature scaling. The plot shows the experiment Amazon $\rightarrow$ Webcam of Office31.

Ablation Study. Table 4.5 demonstrates the effectiveness of our pseudo-labeling procedure, FTSP, which achieves strong performance on Office31 and Office-Home datasets. The addition of pseudo-label refinement phase (PR) and our TSAL objective further improve this performance. Figure 4.4 presents the impact of TSAL's temperature scaling on both the discriminability and diversity terms during the Amazon $\rightarrow$ Webcam experiment of Office31. As expected, as training progresses, the discriminability term rises due to the increasing temperature. This counteracts the network's over-confidence and keeps the pseudo-label information relevant. In contrast, without temperature scaling, the diversity term drops close to its lowest possible value in the first epoch ($\log(1/31) \simeq -3.43$, where 31 is the number of classes). But with temperature scaling, the network's predictions are sharpen, allowing for a higher diversity value. These effects result in an improved adaptation (from 92.8%, without temperature scaling, to 94.7% accuracy in the considered experiment). A comprehensive ablation study involving different classifiers for FTSP, hyperparameters and values of trusted samples (K) is presented in supplementary material, Sec. A.2.

Robustness. We evaluated our algorithm across four different benchmarks. For ImageCLEF-

DA (Tab. 4.4) and Adaptiope (Tab. 4.3), experiments were conducted with 5 seeds each. On the small ImageCLEF-DA, all methods we considered show stable results. However, in Adaptiope with its pronounced domain gaps, certain methods encountered difficulties. AAD did not achieve satisfactory results for both ResNet50 and ViT-L architectures. NRC yielded less than optimal results for both, and while SHOT performed well with ViT-L, it faced challenges with ResNet50. In contrast, our proposed method demonstrated consistent performance across both architectures, surpassing other approaches.

Computational Efficiency. The computational demands of our approach align with those of the most efficient state-of-the-art SF-UDA methods, such as SHOT [56]. Notably, our publicly available implementation facilitates the distribution of TAB computations across multiple GPUs. This capability significantly enhances the scalability of our algorithm, making it adaptable to larger datasets and more complex architectures. For a detailed analysis of the computational processes involved, readers are referred to the Appendix A.3.

Remarks and limitations. The experiments on all datasets were conducted using constant, and potentially not-optimal, hyperparameters. Our **post hoc** ablation study in Sec. A.2 demonstrates the robustness of our method to hyperparameters variations. Moreover, it suggests also that by choosing different configurations, performance can be further enhanced. For example, with ResNet50, we achieved an average accuracy of **90**.3% on Office31 and **72**.9% on Office Home using $K = 5$. While our approach already shows competitive and superior results when compared with SOTA methods, we recognize that adopting unsupervised hyperparameter selection techniques (e.g., [103]), as utilized by other approaches [57, 58], could further boost our method's performance. We leave this exploration for future work.

4.7 Conclusion

In this work, we introduced **Trust And Balance** (TAB), a novel, simple and effective method for SF-UDA in image classification. It is characterized by two key innovations: Few Trusted Samples Pseudo-labeling, which computes high quality pseudo-labels for the target domain, and Temperature Scaled Adaptive Loss, which balances the diversity, the discriminability and the pseudo-labels significance in the objective. The empirical evaluations show that TAB, despite its simple design, obtains performance similar to or better than SOTA methods in all benchmarks considered and an increased robustness.

Chapter 5

In-Domain vs Out-of-Domain Transfer Learning for Plankton Image Classification

5.1 Context

In this contribution, we present a detailed study on the roles of pre-training, DNN architecture choice, and fine-tuning in a specific real-world problem of image classification: plankton recognition. We specifically design three Transfer Learning pipelines to identify the best practices in this domain and to evaluate a comparison between In-Domain and Out-Of-Domain pre-training of DNNs. The first pipeline involves pre-training the model from scratch on a large-scale plankton dataset. In the second, the model is pre-trained on large-scale natural image datasets (ImageNet1k or ImageNet21k). The third implements a two-stage fine-tuning process (ImageNet $\rightarrow$ large-scale plankton dataset $\rightarrow$ target plankton dataset). Our results show that Out-Of-Domain ImageNet21k pre-training outperforms the In-Domain plankton pre-training, with an average boost in test accuracy of around 6%. In the next part of this work, we adopt three ImageNet21k pre-trained Vision Transformers and one ConvNeXt model, obtaining results (with a single model) that are on par with (or slightly superior to) the state-of-the-art that uses an ensamble of many CNNs. Finally, we design and test an ensemble of our Vision Transformers and the ConvNeXt, surpassing the existing state-of-the-art works in plankton image classification on all the three target datasets.

We note that the specific task considered in this work, i.e., plankton recognition, is significant as plankton are microorganisms that play a crucial role in the aquatic food web.

Recently, there have been proposals to use plankton as biosensors, since they can react to even minimal changes in the aquatic environment with specific physiological responses, potentially leading to morphological and behavioral alterations. The development of high-resolution in-situ automatic acquisition systems has enabled the research community to gather a vast amount of plankton image data. Notable examples include the ZooScan and Woods Hole Oceanographic Institution (WHOI) datasets, which comprise millions of plankton images. However, obtaining unbiased annotations is costly in terms of time and resources, and in-situ acquired datasets generally suffer from severe imbalance, with only a few images available for several species.

5.2 Introduction

The term *plankton* refers to a large class of drifting aquatic microorganisms. Plankton plays a key role in the aquatic ecosystem, being at the bottom of the marine food chain. Moreover, phytoplankton is estimated to have produced around 50% of the total atmosphere oxygen with fundamental involvement in local and global climate regulation [129]. Plankton community composition is deeply impacted by natural or artificial perturbations of the aquatic environment [130]. Plankton microorganisms can respond to changes in the environment with physiological changes, potentially causing morphological, and behavioral modifications [131]. For these reasons, their usage as biosensors has been proposed: detecting deviations from a computed healthy baseline as indicators of potentially dangerous environmental changes [132, 133].
The development of advanced in-situ high-resolution automatic acquisition systems, e.g., the submersible flow cytometer [134, 135] and the In Situ Ichthyoplankton Imaging System (ISIIS) [136], is leading to a large amount of available plankton image data. In particular, from 2006 to 2014 the Woods Hole Oceanographic Institution (WHOI) acquired a large-scale dataset comprising millions of plankton images, labeled by experts in the field in 103 categories. Another example is the ZooScan dataset (acquired by means of the homonymous instrument [137]) which includes 1.4 million images labeled into 98 different categories. While there is a growing availability of such data, high-quality unbiased annotations can be costly in terms of both time and resources [138, 139], furthermore there is a pressing need to develop highly accurate algorithms for automatic plankton image classification. To address this challenge, researchers have turned to machine learning solutions, particularly supervised training of Convolutional Neural Networks (CNNs) [140, 141, 142, 143, 144], which have demonstrated superior performance compared to traditional Computer Vision

methods, as highlighted by several studies. CNNs are powerful Deep Learning architectures commonly used for image classification and object recognition tasks: they consist of multiple convolutional layers that can learn and extract hierarchical representations of input images, allowing the network to identify features of varying complexity [145].

A widely used approach in plankton image classification is Transfer Learning, where the weights of a pre-trained CNN architecture on a large general dataset (such as natural images) are fine-tuned with the images and labels of a specific plankton dataset, as proposed by various works [143, 146, 147] (the *knowledge* acquired by the model in the pre-trained dataset is *transferred* to make the downstream task, i.e., plankton classification, easier). To achieve state-of-the-art performance in plankton classification, current approaches typically involve fine-tuning multiple CNN models, often six or more, and combining their predictions through ensemble methods to obtain highly accurate results [143, 148, 149]. These methods typically rely on the widely-used ImageNet1k dataset for pre-training and are evaluated on small-to-medium-sized plankton datasets that have been curated from larger image collections, including WHOI22 [150], Kaggle38 [136] and ZooScan20 [137], where the number following the name denotes the number of classes included in the dataset (see Sec. 5.4.1). ImageNet1k denotes the subset of ImageNet that consists of $1,000$ natural image classes and was used in the ImageNet Large Scale Visual Recognition Challenge 2012 [151]. Conversely, we will refer to the entire dataset, which contains $21,841$ classes, as ImageNet21k.

An important limitation of the aforementioned approaches is that the ensemble requires training of multiple DNNs and, also, they should be used concurrently at inference time, impacting the efficiency of the resulting method. Furthermore, only *classical* CNNs are typically considered for plankton classification, and new architectures, designed in recent years, have not been yet fully explored. Additionally, to the best of our knowledge, no study has comprehensively examined the impact of the model pre-training on various large-scale, in-domain plankton datasets versus out-of-domain natural image datasets.

To address these gaps, in this study, we first design three transfer learning pipelines to compare the effect of in-domain (extended versions of the three cited plankton datasets, comprising up to 1.4 million images) and out-of-domain (ImageNet1k [151] and ImageNet21k [152]) source datasets when adopting transfer learning on the three plankton benchmark datasets, exploiting a classical CNN model: ResNet50.

Our experiments indicate that using ImageNet21k for pre-training results in a significant improvement of approximately 6% in test accuracy compared to in-domain dataset pre-training alone. This suggests that the complexity and diversity of ImageNet21k provide valuable learning opportunities for effective plankton classification. While representations

learned from large-scale in-domain plankton datasets are more specialized to the domain, they may be less discriminative than those learned from ImageNet21k.
In the next part of this work, we adopt more recent and complex architectures trained on ImageNet21k: three types of Vision Transformers (i.e., ViT [153], Swin [154] and BEiT [155]) and a modern CNN (i.e. ConvNeXt [97]). Vision Transformers have been introduced in [153] and, in contrast to CNNs, exploit a self-attention mechanism [32] to aggregate information from patches of an image, enabling the model to recognize objects by attending to different parts of the image simultaneously. We fine-tune each of the Transformers and the ConvNeXt on our three target plankton datasets and our results show that the BEiT outperforms the ResNet50 model with an average improvement of 2% in terms of test accuracy. Comparing our results with the best ensembling methods, our experiments show that the ImageNet21k pre-trained BEiT Transformer outperforms the state-of-the-art ensembles on Kaggle38 and ZooScan20 and obtains a similar performance on the WHOI22 dataset. Additionally, we investigate whether combining the three Vision Transformers and the ConvNeXt models within an average ensemble architecture could bring further improvement in accuracy. However, the accuracy gain (compared to our best single model) is minimal and counterbalanced by the resulting additional computational complexity. Nevertheless, the ensemble classifier outperforms the state-of-the-art results for all three investigated datasets.

The remainder of the chapter is organized as follows: first, we introduce the related works on Transfer Learning for plankton image classification. Then, we provide details on the datasets (Sec. 5.4.1) and the implemented pipeline (Sec. 5.4.2). Finally, we provide the experiment details (Sec. 5.5.1), presenting and discussing the obtained results (Sec. 5.5.2).

5.3 Related Works

In recent years, there has been a growing interest in the computer vision community toward plankton image classification [143]. Starting from 2014, when the Kaggle National Data Science Bowl was organized with the aim to create an accurate classifier for plankton images, machine learning has been extensively applied to the task at hand [133]. The main approaches involve designing and extracting features that are later used to train Random Forest or Support Vector Machine (SVM) classifiers [140, 150, 139] or exploit Deep Learning in the form of Convolutional Neural Networks (CNNs) [156, 157, 158, 159, 139, 148, 160, 161]. Nowadays, large-scale annotated plankton datasets are publicly available (e.g., the ZooScan98 [137] and the WHOI80 datasets [135]). However, plankton datasets are typically imbalanced [162], and obtaining high-quality annotations is expensive both in terms of time and resources. A popular solution to deal with these challenges involves the usage of a Transfer Learning framework [149, 160, 143, 148]. In [160] the authors compare the performance of an SVM classifier trained on features extracted by means of CNNs (i.e., the DeepSea [163] and the AlexNet [164]) pre-trained on the extended Kaggle plankton dataset [136] with 30 thousand images and ImageNet1k. The authors find only a slight difference in the performance of AlexNet pre-trained on the Kaggle plankton dataset and ImageNet1k when using it as a features extractor on their in-house dataset. In [143], the authors adopt an ensemble of different CNN models with three different classification pipelines involving Transfer Learning, testing them on the same benchmark datasets used in this work. In particular, they compare: (i) a CNN pre-trained on ImageNet1k and fine-tuned on the plankton target datasets; (ii) a two-round fine-tuning procedure, where the ImageNet1k pre-trained model is fine-tuned on a source plankton dataset and further trained on the target plankton datasets. In this work, the source dataset is obtained by fusing the extended version of the Kaggle dataset [136] ($15,962$ images and 83 classes) and a dataset referred to as *Esmeraldo* ($11,005$ images and 13 samples). The two-round fine-tuning procedure provides small improvements or degradation of test accuracy, depending on the model and the target dataset, with respect to a direct fine-tuning of the pre-trained model. Moreover, the designed ensemble of CNNs provides a boost in accuracy. In [149] the authors adopt average and stacking ensembling of six CNN models including a DenseNet [165] and EfficientNets [166]. All the CNN models are pre-trained on ImageNet1k. Their ensemble of six CNNs outperforms previous state-of-the-art results for the classification of the investigated plankton datasets.

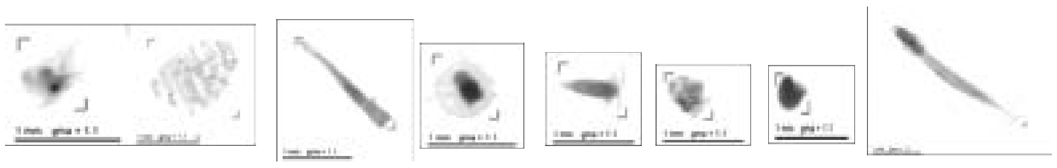
In [161] the authors compare different Transfer Learning scenarios using an ImageNet1k pre-trained AlexNet, fine-tuned on the extended Kaggle dataset, an extended version of the WHOI dataset with $53,239$ images, and both of them in cascade. Their results show that the ImageNet1k pre-trained CNN is more accurate than the same model pre-trained on a plankton dataset, with the two-stage fine-tuning giving only a slight improvement. The previously cited works focus on plankton image classification, which is the same task considered in our study. However, it is worth noting that the advantages of pre-training within a Transfer Learning framework have been investigated in other computer vision tasks applied to plankton, such as specimen detection [167], where the classification of plankton microorganisms is coupled with localization. Up to our knowledge, no work for specimen detection performs a systematic analysis on the effect of in-domain pre-training for the detection task, with most of the methods based on the fine-tuning of a pre-trained model on the plankton target dataset. In these works, the usage of models pre-trained on out-of-domain source datasets allows compensation for the limited availability of data, that prevents training from scratch. In the context of object detection, DNNs are typically pre-trained on Microsoft Common Objects in Context (MS-COCO), which is a popular out-of-domain object detection dataset. In [168], the authors design a mask region CNN to perform multi-class microorganisms detection. The proposed model is pre-trained on MS-COCO and then fine-tuned on a plankton dataset, achieving good detection performance also on an out-of-domain blood dataset. In [169], the authors introduce a phytoplankton image dataset, to be used as a candidate source dataset for the specimen detection task. In this work, a Faster R-CNN with an ImageNet pre-trained backbone is fine-tuned on the introduced dataset, showing high detection accuracy. In [170] an ImageNet pre-trained CNN is exploited to extract features from plankton images in a specimen detection task. The pre-trained features are shown to provide higher accuracy with respect to a set of hand-crafted features, without any fine-tuning on the plankton detection task.

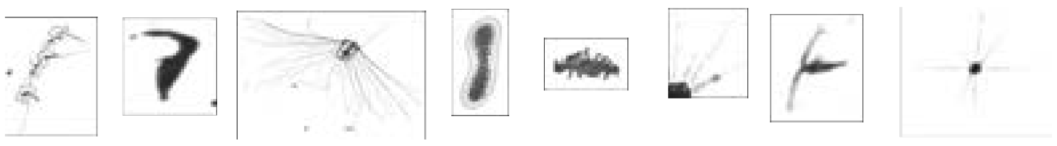
Previous works have not systematically addressed the problem of in-domain versus out-of-domain Transfer Learning in plankton image analysis. They instead rely on small-scale plankton datasets as sources and typically employ classical CNN models. The ensembles of CNNs designed in these works tend to yield better performance than single models, however, limited insights are provided on the trade-off between increased complexity and computational training/test time and accuracy improvement. To address these gaps, this work proposes three Transfer Learning pipelines to systematically evaluate the effectiveness of plankton in-domain and natural images out-of-domain pre-training datasets in a Transfer Learning framework. We consider source in-domain plankton datasets with up to one million images to allow a fair comparison in terms of the number of images with ImageNet datasets. Finally, we design an ensemble of three Transformers and one ConvNeXt, evaluating its effect in terms of the trade-off between complexity and accuracy gains for the task at hand.

5.4 Methods

5.4.1 Datasets

In this work, we exploit three popular benchmark plankton image datasets. The target datasets are the same used in [140, 143, 149, 148]: (i) WHOI22, (ii) Kaggle38; (iii) ZooScan20. Each of these datasets is a subset extracted from a corresponding larger collection of annotated images. We consider the correspondent large-scale datasets as in-domain source datasets to pre-train our models when testing the proposed Transfer Learning pipelines. In the next paragraph, we provide a short description. Fig. 5.1 shows sample images of eight species for each of the three included datasets, while Table 5.1 provides more details on the number of images and classes included.


(a) ZooScan dataset.


(b) Kaggle dataset.

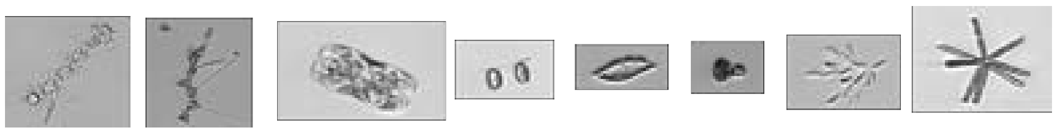

(c) WHOI dataset.

Figure 5.1 Sample images from seven different classes included in the datasets considered for our analysis.

WHOI dataset

The WHOI dataset [135] (see Fig. 5.1c) refers to a public large collection of plankton images acquired by the Woods Hole Oceanographic Institution (WHOI) using automated submersible imaging-in-flow cytometry by means of an Imaging FlowCytobot (IFCB), from 2006 to 2014 [134]. The dataset includes 3.4 million images labeled into 103 categories. A subset of the WHOI dataset, introduced in [150], includes $6,600$ images labeled into 22 categories. This subset is referred to as WHOI22. Starting from the whole WHOI dataset, we eliminate all the 22 classes of the WHOI22 and the class labeled as *mix*, obtaining $253,952$ images belonging to 80 different species of plankton. In this work, we refer to the resulting dataset as WHOI80. We use the WHOI80 as an in-domain source dataset, while the WHOI22 is exploited as a target dataset. The dataset is natively available with a test set, with a number of images equal to the training set.

Table 5.1 Schematic overview of the eight datasets used in this work including number and type of images, number of classes, and role in the Transfer Learning pipeline.

Dataset	**# images**	**# classes**	**imeges type**	**role**
ImageNet21k	$14,197,122$	$21,841$	natural	out-domain source
ImageNet1k	$1,281,167$	$1,000$	natural	out-domain source
ZooScan98	$1,400,000$	98	plankton	in-domain source
WHOI80	$253,952$	80	plankton	in-domain source
Kaggle83	$15,962$	83	plankton	in-domain source
Kaggle38	$14,374$	38	plankton	target
WHOI22	$6,600$	22	plankton	target
ZooScan20	$3,771$	20	plankton	target

Kaggle dataset

The Kaggle dataset [136] (see Fig. 5.1b) refers to a collection of plankton images acquired in the Straits of Florida by means of the In Situ Ichthyoplankton Imaging System (ISIIS), and exploited for the National Data Science Bowl 2015 Kaggle competition. The original labeled version of the dataset includes $30,336$ images belonging to 121 different classes. In [140, 143] the authors use a subset of such dataset, including $14,374$ greyscale images labeled into 38 classes. We refer to such a subset as Kaggle38 in the remainder of this chapter. Starting from the whole labeled dataset, we remove the samples belonging to the 38 classes of the Kaggle38 subset, obtaining $15,962$ plankton images belonging to 83 different categories (as done in [143]). We refer to this version of the dataset as Kaggle83. We use the Kaggle83 as an in-domain source dataset and the Kaggle38 as a target dataset to test our Transfer Learning pipelines. Since no test set is available, we adopt the same test protocol of [143, 140] using a 5-fold cross-validation procedure.

ZooScan dataset

The ZooScan dataset [171] (see Fig. 5.1a) refers to a large-scale collection of plankton images acquired by means of an instrument named ZooScan[137]. The complete version of the dataset includes 1.4 million images labeled into 98 classes (we refer to this dataset as ZooScan98). A popular benchmark plankton dataset extracted from ZooScan98 is used in many works[140, 143]. We refer to such a subset as ZooScan20, it contains $3,771$ greyscale images labeled into 20 classes. We use ZooScan98 as an in-domain source dataset and ZooScan20 as a target dataset to test our Transfer Learning pipelines. Since no test set is available, we use again the same test protocol of [140, 143] adopting a 5-fold cross-validation procedure.

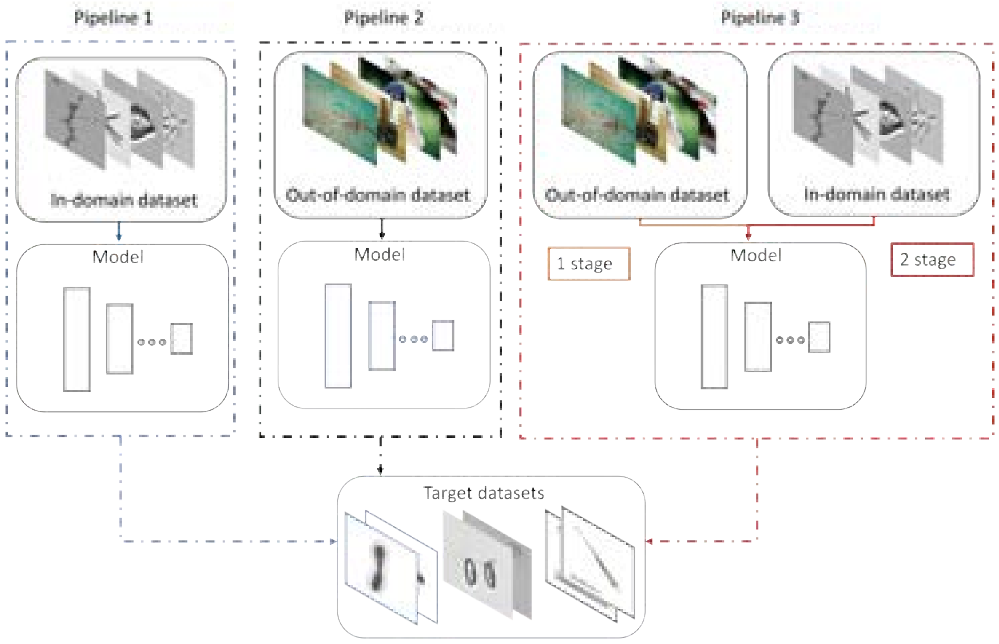


Figure 5.2 Schematic representation of the three implemented Transfer Learning pipelines. The dashed blue square corresponds to the first pipeline, where a model is pre-trained from scratch on a large-scale in-domain plankton dataset; the dashed black square identifies the adoption of out-of-domain ImageNet pre-training; the dashed red square represents the two-stage fine-tuning procedure (ImageNet $\rightarrow$ in-domain plankton dataset $\rightarrow$ target dataset).

5.4.2 Transfer Learning pipelines

Fig. 5.2 shows a schematic representation of the pipelines we designed to evaluate the impact of in-domain and out-of-domain Transfer Learning on plankton image data. In the first Transfer Learning pipeline (dashed blue square in Fig. 5.2), we use the extended version of the plankton datasets included in our analysis (see Sec. 5.4.1) as in-domain source datasets to train a ResNet50 model [172] from scratch. The resulting model is then fine-tuned on each of the three target datasets and evaluated in terms of accuracy and F_1 score on the test sets (see Sec. 5.5.1 for further details).

In the second Transfer Learning pipeline (dashed black square in Fig. 5.2) we use two popular natural image datasets as out-of-domain source datasets to train a ResNet50 model: ImageNet1k and ImageNet21k. The first is a collection of 1.2 million images belonging to $1,000$ different classes, while the second includes 14 million images labeled into $21,841$

categories [173]. We fine-tune the resulting model on each of the three target datasets and evaluate it in terms of accuracy and F_1 score on the test sets. Finally, for the two in-domain plankton datasets with less than one million images (i.e., WHOI80 and Kaggle83), we design a third Transfer Learning pipeline (dashed red square in Fig. 5.2) adopting a two-stage fine-tuning procedure, in the attempt to mitigate the effect of the number of images, when comparing to the out-of-domain ImageNet datasets. In particular, we first fine-tune a ResNet50 model pre-trained on ImageNet21k on one plankton in-domain dataset, later performing another stage of fine-tuning on each of the three target datasets.

5.4.3 Ensemble of Transformers and ConvNeXt architectures for plankton image classification

In this work, we first test the designed Transfer Learning pipelines exploiting a ResNet50 architecture. Then, we consider deeper and more complex architectures, namely Vision Transformers and a ConvNeXt. In particular, we adopt and compare ViT [153], a hierarchical Transformer (i.e., Swin)[154], a BEiT Transformer [155] and ConvNeXt [97] to accurately classify our target plankton image datasets. All the models are pre-trained on ImageNet21k and fine-tuned on the target datasets. Finally, following the state-of-the-art approaches for plankton image classification, we combine the four models into an ensemble, to evaluate the impact on performance on the target datasets. In particular, we average the output probabilities for each of the models, selecting the output class based on the maximum of the obtained values.

5.5 Results

5.5.1 Experiment details

Image pre-processing

The plankton datasets used in this work include images of different sizes and aspect ratios. An important requirement for the efficient training of a neural network consists in having input images of the same size, allowing them to be batched into tensors for hardware acceleration. Additionally, for Transformer architectures, square input images are desirable as they are divided into a grid of pre-defined square patches during training. Therefore, we follow the resizing strategy employed in previous works [143]: (i) the aspect ratio is maintained by padding the smallest dimension of each image, achieving a square shape; (ii) all the images are resized to a fixed size; and (iii) a square region is cropped from the resulting image. For ZooScan images, prior to the described pipeline, we automatically remove the artifact represented by the size indication legend. The resize and crop sizes are consistent with the ones used for pre-training each architecture: for ResNet50, images are resized to 256×256 and then cropped to 224×224, while for other architectures (ViT, BEiT, Swin, and ConvNeXt), images are resized to 439×439 and then cropped to 384×384. During training, the crop is randomly performed across the image as an augmentation technique. During testing, the crop is centered on the image.

Training details

Before fine-tuning the model weights, we proceed by substituting the existing fully-connected layers on top of each model with a newly initialized bottleneck. This bottleneck comprises a linear layer with 512 neurons, a normalization layer, and a non-linear activation function. Finally, a linear classification layer is added with the number of output dimensions matching the number of classes. The normalization is a Layer Normalization [174] (with GELU activation function) or a Batch Normalization [175] (with ReLU activation function) according to the used backbone (the former for Vision Transformers and ConvNeXt, the latter for ResNet50). We train the final classifier applying Weight Normalization [176]. We use data augmentation based on random horizontal and vertical flips, Stochastic Gradient Descent (SGD) [177] with Nesterov momentum (0.9) for the optimization, and cross-entropy as loss function. We use regularization with weight decay (10^{-2}) and label smoothing (0.1). The initial learning rates are 10^{-3} for the pre-trained backbone and 10^{-2} for the bottleneck and the classifier. They are decayed with exponential scheduling: at training step t, the learning rate is evaluated as the initial learning rate multiplied by $\text{decay}(t) = \left(1 + \gamma \frac{t}{n}\right)^{\beta}$ where $\gamma = 10$, $\beta = 0.75$ and n is the total number of training steps (# epochs $\cdot$ # steps in one epoch). We use 100 epochs with early stopping (training/validation split is 85/15). The batch size is 64, but we split every batch across 4 GPUs (NVIDIA V100 16 GB), exploiting gradient accumulation, when needed. We synchronize batch normalization statistics across GPUs. For our experiments, we used Python (version 3.9.12) with PyTorch library (version 1.11.0) and CUDA 10.2. We imported the architecture implementations from the TIMM library [88]. The ConvNeXt model used in our work is ConvNeXt-XL architecture, while for the Transformers the BEiT-L, ViT-L, and Swin-L implementations are adopted.

Evaluation metrics

We evaluate our results by exploiting two common metrics for plankton image classification (as done in [148]): accuracy and F_1 score, defined as:

$$\text{Accuracy} := \frac{\text{Total True Positives}}{\text{Total Instances}} \tag{5.1}$$

$$\mathrm{F}_1 \text{ score} := \frac{1}{C}\sum_{i=1}^{C} \mathrm{F}_1 \text{ score}_i \tag{5.2}$$

$$\mathrm{F}_1 \text{ score}_i := 2 \cdot \frac{\text{Precision}_i \cdot \text{Recall}_i}{\text{Precision}_i + \text{Recall}_i} \tag{5.3}$$

In Eq.5.1, *Total True Positives* represents the sum of true positives across all classes, and *Total Instances* represents the total number of images in the test dataset. In Eq.5.2, C represents the total number of classes, and F_1 score$_i$ represents the F_1 score corresponding to instances in class i. The latter is computed as shown in Eq. 5.3, where $\text{Precision}_i = \frac{\text{TP}_i}{\text{TP}_i+\text{FP}_i}$ and $\text{Recall}_i = \frac{\text{TP}_i+\text{TN}_i}{\text{TP}_i+\text{FP}_i+\text{FN}_i+\text{TN}_i}$. True Positive (TP$_i$), True Negative (TN$_i$), False Negative (FN$_i$), and False Positive (FP$_i$) correspond to the element in the confusion matrix of class i. In summary, the accuracy metric provides a measure of performance, considering each instance equally important. The F_1 score provides a measure of performance considering each class equally important when calculating the average. If a dataset is balanced, with the same number of instances per class, F_1 score and accuracy coincide, however, in the case of imbalanced datasets, such as the plankton ones [162], F_1 score may be considered a relevant additional metric in the evaluation of a classification task. Finally, for Kaggle38 and ZooScan20 datasets, the evaluation metrics are averaged among the 5 folds (see Sec. 5.4.1).

5.5.2 Experiment results

In-domain versus out-of-domain Transfer Learning

We apply the Transfer Learning pipelines described in Sec. 5.4.2 to the three datasets used in this work (see Sec. 5.4.1). The experiments reported in this section, are performed using ResNet50 as a baseline architecture. Table 5.2 shows the obtained results in terms of accuracy and F_1 score evaluated on the test set. It is worth noticing that the three extended versions of the plankton datasets used as source datasets for the in-domain Transfer Learning pipeline have a different number of images: (i) $15,962$ for the Kaggle83; (ii) $253,952$ for the

Table 5.2 Performance comparison (accuracy and F_1 score) of ResNet50 using the proposed Transfer Learning pipelines across the three benchmark datasets. The best results are highlighted in bold, second best results are underlined.

Target dataset ➔	**WHOI22**		**Kaggle38**		**ZooScan20**	
➘ Source dataset(s)	**Accuracy**	**F_1 score**	**Accuracy**	**F_1 score**	**Accuracy**	**F_1 score**
WHOI80	0.878	0.878	0.876	0.831	0.826	0.837
Kaggle83	0.862	0.862	0.878	0.834	0.847	0.863
ZooScan98	0.912	0.912	0.914	0.884	-	-
ImageNet21k	**0.946**	**0.946**	**0.930**	**0.909**	<u>0.887</u>	**0.899**
ImageNet1k	<u>0.939</u>	<u>0.939</u>	0.921	0.895	0.851	0.868
ImageNet21k $\rightarrow$ WHOI80	**0.946**	**0.946**	0.924	0.905	**0.891**	<u>0.898</u>
ImageNet21k $\rightarrow$ Kaggle83	0.938	0.938	<u>0.929</u>	<u>0.907</u>	0.877	0.896

WHOI80 and (iii) 1.4 million for the ZooScan98. As a comparison, ImageNet21k has 14 million images belonging to $21,841$ classes. ImageNet1k is a subset of ImageNet21k with 1.2 million images belonging to $1,000$ classes (with a size comparable to the ZooScan98 plankton dataset). As we can see in Table 5.2, ImageNet21k pre-training leads to the most accurate model for the WHOI22 and the Kaggle target datasets both in terms of accuracy and F_1 score. ImageNet21k also leads to the best F_1 score for the ZooScan dataset, while there is a slight improvement when using a two-stage fine-tuning involving the WHOI dataset (+0.004%) w.r.t. the test accuracy, on this dataset. Moreover, if we consider only the in-domain Transfer Learning pipeline, it is possible to notice that the ZooScan98 dataset leads to the best results for both the WHOI22 and the Kaggle dataset, with an average improvement of around 3.6% w.r.t. pre-training on the other two extended plankton datasets. We do not use ZooScan98 as a source dataset for the fine-tuning on ZooScan20, because it contains all the images and the classes included in the target dataset. In fact, differently from WHOI80 and Kaggle83 extended dataset, we do not remove the classes in common with the target dataset for ZooScan98, because we are interested in considering a dataset with a size comparable to ImageNet1k, in order to fairly compare one in-domain plankton dataset to the external natural images dataset removing the number of images as potential influencing parameter. Our findings suggest that using in-domain plankton datasets as sources in Transfer Learning frameworks, has a limited or no effect on the accuracy of tested models, while the number of classes and images in a source dataset are important factors that contribute to the quality of a pre-training dataset.

Table 5.3 Performance comparison (Accuracy and F_1 score) of Vision Transformers, ConvNeXt, and ResNet50 (as baseline) pre-trained on ImageNet21k across the three benchmark datasets. The best results are highlighted in bold, second best results are underlined.

Dataset →	**WHOI22**		**Kaggle38**		**ZooScan20**	
↓ Model	**Accuracy**	**F_1 score**	**Accuracy**	**F_1 score**	**Accuracy**	**F_1 score**
ResNet50	0.946	0.946	0.930	0.909	0.887	0.899
BEiT	**0.961**	**0.961**	**0.951**	**0.942**	**0.914**	**0.931**
Swin	<u>0.960</u>	<u>0.960</u>	0.947	0.932	0.904	0.917
ViT	0.959	0.959	0.948	<u>0.933</u>	<u>0.908</u>	<u>0.918</u>
ConvNeXt	0.957	0.957	<u>0.949</u>	0.932	0.904	0.911

Exploiting the pre-training on ImageNet21k: Transformers and ConvNeXt for plankton classification

The out-of-domain natural image dataset ImageNet21k corresponds to the best source dataset when pre-training a ResNet50 in our experiments, in terms of test accuracy. Having this in mind, we investigate the performance of more complex architectures that could benefit even more from an ImageNet21k pre-training. In particular, we consider three different Transformers: ViT [153], the hierarchical Swin Transformer [154] (Swin) and BEiT [155]. We also include a modern CNN, i.e., ConvNeXt [97], in our analysis. Table 5.3 shows the performance of each of these models on the three plankton benchmark datasets. In our experiments, the three Transformers and the ConvNeXt model are pre-trained on ImageNet21k. As we can see, BEiT Transformer shows the highest performance both in terms of test accuracy and F_1 score, with an average improvement of 2% with respect to the ResNet50 model pre-trained on ImageNet21k (see Table 5.2). As a benchmark, we compare our results with four recent state-of-the-art works on plankton image classification [149, 140, 143, 148]. Table 5.4 summarizes state-of-the-art results on the three investigated target plankton datasets. Excluding [140], the state-of-the-art benchmark results are obtained by ensembling several ImageNet1k pre-trained CNN models (six CNNs in [149], eleven in [143]). As we can see in Table 5.3, our single BEiT model outperforms the state-of-the-art results for the Kaggle and the ZooScan dataset, with performance comparable to [149] on the WHOI22 dataset, where an ensemble of six CNN models is used.
Nonetheless, inspired by previous state-of-the-art results in plankton image classification, we design an average ensemble of our ImageNet21k pre-trained Transformers and ConvNeXt (see Sec. 5.4.3 for further details) to assess the effect on performance with respect to the three target datasets. As we can see in Table 5.4, the resulting ensemble model provides a

minimal effect on accuracy, with an average increase of around 0.6% with respect to our best performing Transformer (i.e., BEiT).

Table 5.4 Performance comparison (Accuracy and F_1 score) of our best single model (BEiT) and our ensemble of 4 models with state-of-the-art approaches on three investigated plankton datasets. The best results are highlighted in bold and the second best results are underlined.

Dataset →	**WHOI22**		**Kaggle38**		**ZooScan20**	
↓ Method	**Accuracy**	**F_1 score**	**Accuracy**	**F_1 score**	**Accuracy**	**F_1 score**
Best 6 average [149]	0.961	0.961	0.947	0.937	0.898	0.915
Best 6 stack [149]	0.958	0.958	0.943	0.934	0.891	0.911
SFFS [143]	0.958	0.958	0.942	0.927	0.885	0.900
WS [143]	0.958	0.958	0.942	0.927	0.888	0.902
Fus 2R + Fus 1R [148]	-	0.953	-	0.926	-	0.897
Fus PR+ Fus 2R +Fus 1R [148]	-	0.953	-	0.926	-	0.896
NLMKL [140]	-	0.900	-	0.846	-	0.894
BEiT (**ours**)	0.961	0.961	0.951	0.942	0.914	0.931
Ensemble (4 models, **ours**)	**0.966**	**0.966**	**0.955**	**0.945**	**0.925**	**0.937**

However, the minimal increase in accuracy is counterbalanced by a significant increase in time and resources needed for training and inference. Table 5.5 reports an indication of training and inference time, as the number of images that can be processed per second, by the different architectures considered in our study (and by the ensemble of the 4 architectures) on a single NVIDIA V100 GPU. These numbers depend on the specific hardware and implementation. However, they highlight the difference, in terms of efficiency, among the architectures, and the increase in time needed for computation when ensembling the four models. Thus, the trade-off between complexity and accuracy gain should be carefully evaluated, depending on the specific application (e.g., real-time or post-acquisition analysis).

Table 5.5 The average number of images processed by our models in one second at training and inference time. The values have been evaluated based on 1000 iterations. The higher the value, the faster the processing time.

Model	BEiT	ViT	SWIN	ConvNeXt	Ensemble
Training (imgs/s) ↑	20.32	21.72	32.57	13.16	4.95
Inference (imgs/s) ↑	65.68	70.26	102.70	52.88	17.21

5.6 Conclusion

In this work, we compare in-domain and out-of-domain Transfer Learning approaches for plankton image classification. We design three different Transfer Learning pipelines using three large-scale in-domain source plankton datasets (i.e., WHOI80, Kaggle83, and ZooScan98) and two out-of-domain natural image datasets (i.e., ImageNet1k and ImageNet21k).

The general framework consists in fine-tuning a pre-trained model on three target plankton datasets (i.e., WHOI22, ZooScan20, and Kaggle38). In the first pipeline, we train a model from scratch on an in-domain plankton dataset. In the second pipeline, we adopt an ImageNet1k or ImageNet21k pre-trained model, while in the third, we implement a two-stage fine-tuning procedure, fine-tuning an ImageNet pre-trained model on an in-domain source plankton dataset.

Regarding the first pipeline, we exploit three in-domain source datasets with different numbers of images and classes (see Sec. 5.4.1). Our experiments show that the ZooScan98 dataset with 1.4 million images and 98 classes provides the best performance when used as a source dataset, with an average improvement of 3.6% compared to the pre-training with the other two in-domain datasets.

From the second pipeline, we obtain that ImageNet21k provides better performance compared to ImageNet1k, with an average improvement of 4%. These results suggest that there is no benefit in using a large-scale in-domain plankton dataset as a source dataset for Transfer Learning compared to the out-of-domain ImageNet. Moreover, little or no benefit is obtained when adopting a two-stage fine-tuning procedure. It is worth noticing that ZooScan98 has a higher number of images than ImageNet1k, but leads to lower performance when used as a source dataset. These results may indicate that the number of images and classes are key factors for a pre-training dataset in a plankton image classification task. It is worth noticing that, despite acquiring and annotating large-scale plankton datasets (as ZooScan98) is expensive in terms of time and resources, our experiments show that the usage of in-domain pre-training datasets provides no benefit with respect to ImageNet.

In the next experiments, we adopt current state-of-the-art architectures (ViT, Swin, BEiT, and ConvNeXt, pre-trained on ImageNet21k). The pre-trained models are fine-tuned on the target plankton datasets, providing an average accuracy boost of 2% with respect to the ResNet50 model pre-trained on ImageNet21k. As a benchmark, we compare the obtained results to recent state-of-the-art plankton image classification works, where ensembles of CNN models (up to 11) are used for the task at hand. Our results show that our single BEiT model achieves better performance than state-of-the-art on the Kaggle and the ZooScan datasets, with similar performance to [149] for the WHOI dataset. Following the current trend in plankton image classification, we further design and test an average ensemble of the three transformers and the ConvNeXt. The designed ensemble brings a slight improvement with respect to the ImageNet21k pre-trained BEiT. However, it should be noted that such a boost in accuracy (0.6% on average) is counterbalanced by a significant increase in the computational resources and the training/inference time for the final model.

5.7 Data and code availability

All the code needed to reproduce our results is open-source and available at https://github.com/Malga-Vision/plankton_transfer. The target plankton datasets are available at: Kaggle38 [136]; ZooScan20 [171] and WHOI22 [150]. The code for downloading the extended version is included in the shared repository.

Chapter 6

Sim2Real Bilevel Adaptation for Object Surface Classification using Vision-Based Tactile Sensors

6.1 Context

In this contribution, we present a novel bilevel UDA approach for a specific robotic application. In particular we design a pipeline to address the Sim2Real domain shift in the field of vision-based tactile sensors for object surface classification. First, we train a Diffusion Model using a relatively small dataset of real-world (unlabeled) images randomly collected from everyday objects via the DIGIT sensor. Subsequently, we employ a simulator to generate images by uniformly sampling the surface of objects from the YCB Model Set. These simulated images are then translated into the real domain using the Diffusion Model and automatically labeled to train a classifier. During this training, we further align features of the two domains using an adversarial procedure. Our evaluation is conducted on a dataset of tactile images obtained from a set of ten 3D printed YCB objects. The results reveal a total accuracy of 81.9%, a significant improvement compared to the 34.7% achieved by the classifier trained solely on simulated images. This demonstrates the effectiveness of our approach. We further validate our pipeline using the classifier on a 6D object pose estimation task from tactile data.

6.2 Introduction

Perception of object properties is a fundamental requirement to accomplish everyday tasks. Humans usually have a sense of the object surfaces through vision but sometimes they have to integrate or substitute this information with tactile information. Knowing the kind of surface they are dealing with while manipulating an object can be of primary importance to understand the pose of the object or to decide the next actions.

Among the available tactile sensors, vision-based tactile sensors [178, 179] represent the state-of-the-art when it comes to help robots perceive object properties as they produce high resolution RGB tactile images of the surface in contact. Such images can be exploited with Deep Learning techniques, despite that they require a huge amount of data for training, which can be difficult to collect in the real world. Although simulators [180, 181] have been proposed to overcome this issue, they hardly reproduce the effect of mechanical properties, e.g., gel deformation, and light distribution with high fidelity.

In this work, we aim to fill the gap between simulated and real images by translating the simulated images to the real domain using a Diffusion Model (DM) trained on few real *unlabeled* images coming from a DIGIT sensor [178] (some exaples are shown in Fig. 6.1). We propose a surface classifier which can distinguish among four classes: flat, curve, edge and corner. The classifier is trained with simulated images that are sampled on the surface of objects from the YCB Model Set [182] and then translated with the DM. To label the images, we sample a point cloud on the object mesh and we employ an *automatic* procedure to evaluate, for each point, the local curvature from which we extract the labels. Notably, the overall procedure to generate training data does not require manual annotations, as it uses a small set of unlabeled data from real contacts and a larger dataset that is acquired in simulation and automatically labeled.

We test the classifier on real tactile images acquired on ten YCB objects. We compare our model by training the classifier with simulated images or with images converted with an alternative DM-based model from the literature [183] having more complex training requirements than ours. The experiments show that our method can achieve better accuracy in the classification task. Moreover, we employ our classifier within a pipeline for 6D object pose estimation [184] from multiple tactile sensors.

We summarize our contributions as follows:

- An image translation procedure to fill the Sim2Real gap using a Diffusion Model which easily generalizes to different vision-based tactile sensors;

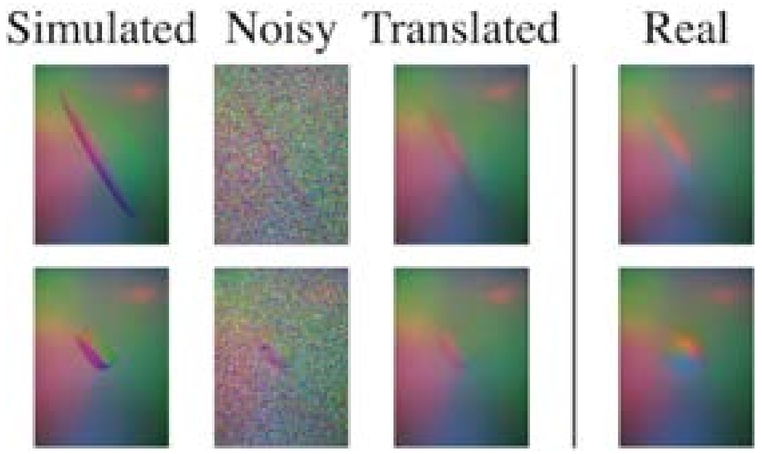


Figure 6.1 Our pipeline uses a Diffusion Model to translate simulated images towards the real domain so as to reduce the Sim2Real gap.

- An object-agnostic surface type classifier trained using simulated images and few real images;
- A method to automatically label the object surfaces given the object mesh, that we use to train the above classifier.

6.3 Related Work

This work draws from the literature on both Sim2Real translation for vision-based tactile sensors and perception of object properties using the same sensors.

Sim2Real

Some work attempt to reduce the Sim2Real gap by carefully emulating data from real sensors. In [185] and [186] the authors mimic the behaviour of the GelSight sensor using Phong's model. In [180], the authors present a simulator for the DIGIT sensor based on OpenGL using pyrender. A limitation of these methods is that they do not take into account the mechanical properties of the sensors. In [181] the authors integrate the dependency on the mechanical properties of the gel by adding a calibration procedure based on the sensor output. Other approaches attempt to mitigate the domain shift between simulated and real images. In [187], the authors try to bridge the Sim2Real gap by employing a CycleGAN [188] to simulate the complex light transmission observed in real sensors, a characteristic not captured by the simulator. In contrast, our work utilizes simulated images, from TACTO

[180], which undergo a transformation via a Diffusion Model. trained on real images, in order to mimic the real deformation of the gel and light transmission of the sensor. A similar approach is considered in [183], where, however, the Diffusion Model is trained with additional conditioning depth images that are obtained from a network [189] that had been previously trained. In our case, we dispense with the need for this network and rely solely on RGB images. In the experimental section, we compare with a variant of our pipeline where the DM for image translation is substituted with that of [183].

Object perception with vision-based tactile sensors

To the best of our knowledge, there is no work that directly classifies the surfaces of an object using vision-based tactile sensors. Nonetheless, we refer to other works which infer similar properties of the object.
In [190] the authors integrate vision with touch to infer the shape of an object. In our work, no visual information is needed. In [191] the authors train a network to retrieve the location and shape of the contacts in order to infer the shape of the object while the sensor slides along the object surface. In [192], the authors reconstruct the normals of the local surface of small objects thanks to a network trained with real and simulated images. On the contrary, we concentrate on objects whose size is not comparable to that of the sensor surface, thus producing more ambiguous tactile images and on general classes of surfaces. In [189, 184] the authors identify the possible contact point on the object surface by comparing the input image with a database of images that are sampled on the object surface in advance, while our work does not need any database at inference time.

6.4 Method

Our approach leverages unlabeled real-world data and labeled simulated data in order to obtain good classification performance in real-world scenarios. To achieve this goal, we devise an automated method for acquiring and labeling synthetic data. We incorporate two levels of adaptation to mitigate domain shift and enhance performance. Specifically, we employ a probabilistic Diffusion Model to translate the simulated images. We further adapt the model features through an adversarial process using the Domain-Adversarial Training of Neural Networks (DANN) method [47].

We remark that we always remove the background signal of the DIGIT, i.e. its RGB output when not in contact with an object, from both real and simulated images. As DIGIT

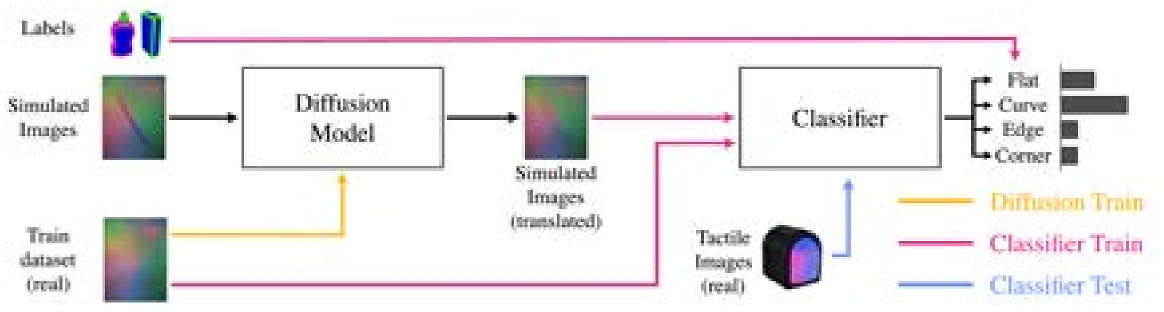


Figure 6.2 Overview of our pipeline for object surface classification.

sensors can exhibit slight background variations owing to manufacturing differences, this ensures that the methodology remains agnostic to the specific background.

This section offers a comprehensive overview of all the components illustrated in Fig. 6.2 within our pipeline.

6.4.1 Acquisition and labeling of simulated data

We employ Poisson disk sampling [193] to extract uniformly distributed point clouds from object meshes augmented with surface normals. For each point we simulate the image produced by the DIGIT, using the simulator [180], while considering several rotations of the sensor around the normal direction and several penetration depths, so as to ensure variability in the collected data.
To automate the labeling process, we devised a simple yet effective algorithm to proficiently categorize each point within the point clouds as either *flat*, *curve*, *edge*, or *corner*. Let suppose to have a point cloud of M points: $P = \{p_i\}_{i=1}^{M}$, where $p_i \in \mathbb{R}^3$. For every point $p_j \in P$ the algorithm, which operates in four distinct steps, computes automatically the class of the point:

1. The neighborhood N_j of p_j is computed: $N_j = \{p \in P : ||p_j - p||^2 \leq R\}$, where the radius R is an hyperparameter of the algorithm.
2. We extract the ordered singular values of the 3×3 covariance matrix of the points in N_j, i.e., $\sigma_1^{(j)} \geq \sigma_2^{(j)} \geq \sigma_3^{(j)}$.
3. We define the curvature level of the point p_j as the ratio:

$$\mathrm{Curv}(p_j) := \frac{\sigma_3^{(j)}}{\sigma_1^{(j)} + \sigma_2^{(j)} + \sigma_3^{(j)}} \tag{6.1}$$

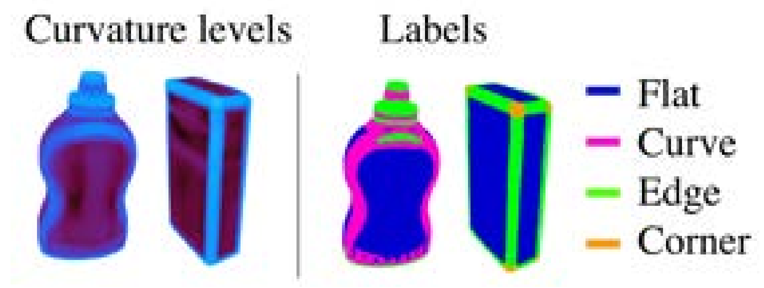


Figure 6.3 On the left: curvature levels (Eq. 6.1) for the YCB objects “mustard bottle” and “sugar box”. On the right: the resulting labels of surface types.

This method provides an estimation of the local surface behavior. Intuitively, when the local curvature level is 0, the local area is entirely flat (as the smallest singular value $\sigma_3 = 0$) and the points would be distributed just on the plane defined by the first two eigenvectors corresponding to σ_1 and σ_2. Conversely, when the local curvature level is $1/3$ (the maximum value attainable by the curvature function), the surface displays pronounced curvature, as all three singular values are equal. By setting two thresholds ($0 < t_1 < t_2 < 1/3$) on the local curvature level, we are able to partition and classify all the points into three categories: *flat*, *curve*, and *hard-curve*.

4. Finally, points previously classified as *hard-curve* are further separated into *edges* and *corners*, a task that cannot be achieved using curvature levels alone. In this step, we utilize K-means clustering on the *normal* vectors of the neighborhood of *hard-curve* points to classify them as either *edges* or *corners*. Intuitively, edges are defined by two flat surfaces represented by two normal vectors. Consequently, the normal vectors of the neighborhood of edge points should be easily clustered with $K = 2$, and the advantage of using $K = 3$ should be negligible. On the other hand, corner points are defined by three planes, and the normal vectors of the neighborhood cannot be easily clustered with $K = 2$. Following this intuition, we compute the following quantity:

$$\Delta \mathrm{loss}_{23} := loss(K = 2) - loss(K = 3), \tag{6.2}$$

where $loss(K = q)$ is the final loss of K-Means when K is set to q. A threshold on the value of $\Delta \mathrm{loss}_{23}$ is set to partition *edges* and *corners*: low values of this metric identify edges, while high values identify corners.

We show the effectiveness of the overall approach for two YCB objects in Fig. 6.3.

6.4.2 Image-level adaptation

As simulated images and actual images acquired with a DIGIT sensor exhibit notable differences (see Fig. 6.1), we propose an unsupervised image-to-image translation method to address the domain shift between these two domains. Our approach leverages the reverse process of a probabilistic Diffusion Model. Diffusion Models are latent variable models that involve two processes: the forward (diffusion) process and the reverse process. Let x_0 represent an image (in our work, a real image acquired with a DIGIT sensor). The forward process is a fixed Markov chain that gradually introduces Gaussian noise to x_0 over a predefined number of steps T, as determined by a variance schedule defined by the parameters $\{\beta_j\}_{j=1}^{T}$. Specifically, at any given time step $t \in 1,\dots,T$, the forward process adds Gaussian noise to the image according to the following transition:

$$q(x_t|x_{t-1}) := \mathcal{N}(x_t; \sqrt{1-\beta_t} \cdot x_{t-1}, \beta_t I) \tag{6.3}$$

where $\mathcal{N}$ is a Gaussian distribution and I is the identity matrix. The reverse process is also a Markov chain with Gaussian transitions that can be learned using a model, such as a neural network. This allows us to recover the image from the previous step (x_{t-1}) given the noisy image at step t (x_t). By iterating this process, we can ultimately obtain the fully denoised image, x_0. To be more specific, the reverse transitions are as follows:

$$p(x_{t-1}|x_t) := \mathcal{N}(x_{t-1}; \mu(x_t,t), \Sigma(x_t,t)) \tag{6.4}$$

Here, $\mu(\cdot,\cdot)$ and $\Sigma(\cdot,\cdot)$ represent functions that can be learned during the training process of the Diffusion Model. In our work, we utilized a U-Net architecture [194], which is commonly employed in the literature.
Following the training, it becomes possible to sample an image with random Gaussian noise (bearing a strong resemblance to the images generated in the forward process at step T) and iteratively denoise it. This iterative process enables the generation of an image from the distribution of the training dataset. For more intuitions and for a more detailed description on probabilistic Diffusion Models the reader can refer to [195] and [54].
To narrow the domain gap between simulated and real images, our approach encompasses the following steps:

1. We train a U-Net model solely with unlabeled real images acquired with the DIGIT sensor to acquire knowledge of the reverse process.

2. Once trained, we introduce a moderate level of noise to the simulated images, advancing in the forward process until step $T' < T$, with the aim of altering the *style* of the images while preserving their semantic information.

3. We employ the reverse process learned in step 1 to denoise the images, moving from step T' to step 0. As the Diffusion Model was trained using real images, the output comprises images that retain the semantic information of the simulated ones but exhibit a style more akin to real images as shown in Fig. 6.1.

Unlike previous works, such as [183], our diffusion method does not necessitate additional information during training or inference. It operates exclusively on raw images without any conditioning. The *semantic conditioning* for image generation is accomplished, as mentioned earlier, by introducing a controlled level of noise into the simulated images, as it is similarly done in [196]. This preserves crucial information while preventing complete degradation.

6.4.3 Feature-level adaptation

Despite the significant reduction in domain shift achieved by the Diffusion Model, some residual differences between the domains still exist. To address this, we adopted a classical yet

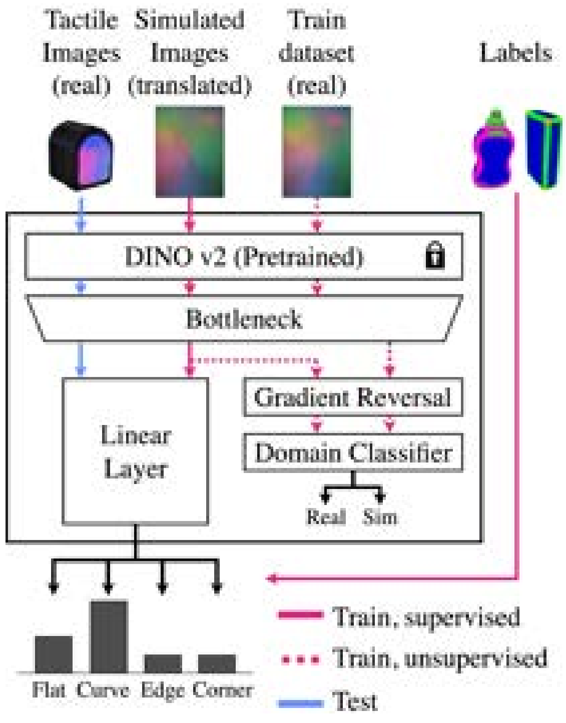


Figure 6.4 Diagram of the classification architecture.

effective adversarial approach, known as Domain-Adversarial Training of Neural Networks (DANN), to facilitate the learning of domain-invariant representations.
Specifically, we utilized a Vision Transformer (ViT) [33], pretrained with DINO v2 method [42], as a feature extractor, keeping it fixed throughout the training process. We introduce and train a bottleneck layer (a linear layer, followed by Layer Normalization [174] and a GELU [197] activation) to map ViT features into a domain-invariant space, along with a classifier that maps these bottleneck features to our target classes, as depicted in Fig. 6.4.
During training, we employ a discriminator to distinguish between real and simulated images, while the bottleneck layer is optimized to deceive the discriminator by making the features from both domains indistinguishable. To be more precise, we define $\phi(\cdot)$ to be our feature extractor (ViT) that maps images to features, $\beta(\cdot)$ as our bottleneck that reduces the dimension of features to 256, $\gamma(\cdot)$ as a classifier that maps bottleneck features to our four classes and $\psi(\cdot)$ as a discriminator that maps bottleneck features to a domain label: 0 for simulated images and 1 for real images. During training we sample a batch of real images (without labels) x_{real} and a batch of simulated images (with labels) (x_{sim}, y_{sim}). We compute the bottleneck features ($\hat{z}$), the predictions of the network $\hat{y}_{sim}$ and the domain predictions of the discriminator $\hat{d}_{all}$ as follow:

$$\hat{z}_{real} = \phi(\beta(x_{real})) \tag{6.5}$$

$$\hat{z}_{sim} = \phi(\beta(x_{sim})) \tag{6.6}$$

$$\hat{y}_{sim} = \gamma(\hat{z}_{sim}) \tag{6.7}$$

$$\hat{d}_{all} = \psi(\hat{z}_{all}) \tag{6.8}$$

where $\hat{z}_{all}$ is the concatenation of $\hat{z}_{sim}$ and $\hat{z}_{real}$ along the batch dimension of the two inputs. The bottleneck and the classifier are trained to minimize the following loss:

$$L_{cls} = \text{CE}(y_{sim}, \hat{y}_{sim}) - \alpha \cdot \text{BCE}(d_{all}, \hat{d}_{all}) \tag{6.9}$$

where d_{all} are all the domain labels (0s and 1s based on the domain of the inputs), CE is the cross-entropy loss, BCE is the binary cross-entropy loss and α is a hyperparameter that we set to 1.2 in all of our experiments.
The discriminator, on the other hand, is trained to minimize:

$$L_{dis} = \alpha \cdot \text{BCE}(d_{all}, \hat{d}_{all}) \tag{6.10}$$

For this reason the training proceeds as an adversarial game between the bottleneck and the discriminator. In practice, we trained the whole network (with the exception of the feature extractor that is fixed) in an end-to-end fashion using a Gradient Reversal Layer (GRL) as proposed in DANN [47] (refer to this work for more details).

6.4.4 Training and testing datasets

For our experiments, we collected three different datasets:

1. The first dataset comprises $5,000$ real images acquired using a DIGIT sensor on the surfaces of randomly selected everyday objects, excluding all objects from the YCB Model Set [182]. We refer to this *unlabeled* dataset as $\textbf{Train}_{real}$.

2. The second dataset consists of $50,000$ simulated images, $12,500$ per class, acquired from randomly selected points sampled on the meshes of ten YCB objects (*cracker box, tomato soup can, tuna fish can, pudding box, gelatin box, banana, bleach cleanser, bowl, power drill, wood block*) and translated to the real domain as in Sec. 6.4.2. We employ the algorithm described in Section 6.4.1 to associate a ground truth label for each image, and we refer to this dataset as $\textbf{Train}_{sim}$.

3. The third dataset includes 792 real images acquired using our setup from ten 3D printed YCB objects (*master chef can, sugar box, mustard bottle, tuna fish can, potted meat can, banana, pitcher base, bleach cleanser, bowl, power drill*) (see Fig. 6.5). For every tactile image we manually label the type of surface depending on the actual type of contact. We use the latter as the ground-truth label for evaluation purposes. We refer to this dataset as $\textbf{Test}_{real}$.

To demonstrate the generalization capabilities of our approach, we intentionally include 5 overlapping objects (*tuna fish can, banana, bleach cleanser, bowl, power drill*) in both the $\textbf{Train}_{sim}$ and $\textbf{Test}_{real}$ datasets. As a result, our algorithm performs well on objects seen during training and also on new objects.
We used the unlabeled $\textbf{Train}_{real}$ dataset for training the Diffusion Model, while the classifier is trained (with DANN) using both $\textbf{Train}_{sim}$ and the unlabeled $\textbf{Train}_{real}$. The testing phase was exclusively conducted on the $\textbf{Test}_{real}$ dataset.

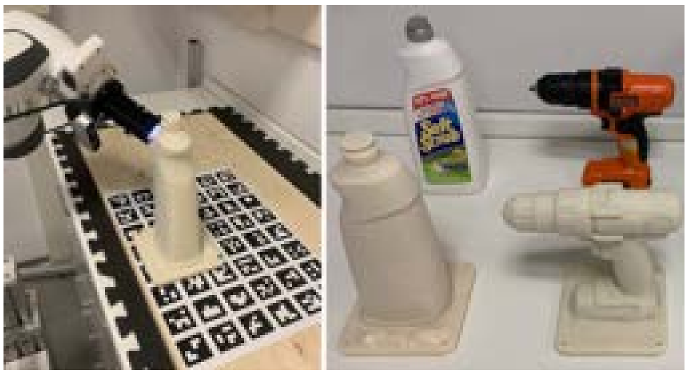


Figure 6.5 **Left**: the testing setup with the DIGIT sensor touching the "bleach cleanser" object. **Right**: example of YCB objects and their 3D printed counterparts.

6.5 Experimental Results

We evaluate the performance of the classifier by considering the accuracy for each object and the F1-Score when analysing the results class-wise. In the results we also consider a variation of the pipeline where the proposed Diffusion Model is substituted with the state-of-the-art alternative [183]. Moreover, we perform several ablation studies in order to investigate the role of the Diffusion Model and of the DANN procedure.

Beyond the classification task, we apply our method within a pipeline for 6D object pose estimation from multiple tactile sensors [184], to show its effectiveness on a practical task. The results of further experiments are provided in the supplementary video.

In order to collect the $\textbf{Test}_{real}$ dataset, we use a DIGIT sensor mounted on a 7-DoF Franka Emika Panda robot to touch ten 3D printed YCB objects in several configurations. We show the setup and examples of the printed objects in Fig. 6.5.

In order to evaluate the outcome of the 6D object pose estimation experiments, we also collect the pose of the object in the robot root frame. To do so, we employ a fiducial system based on a RealSense camera and ArUco markers. To make sure that the object does not move while touching it we fixed it to a table as shown in Fig. 6.5. We remark that the necessity to fix the objects is the only reason for printing them in 3D.

Code and data will be made publicly available online[1].

6.5.1 Experiments on surface classification

Table 6.1 presents the results in terms of *accuracy* for each considered object. In Table 6.2, we instead detail the results for each class using the F1-Score defined as follows:

[1]https://github.com/hsp-iit/sim2real-surface-classification

Table 6.1 Results of the classification experiments: accuracy for each object and averaged.

Image transl.	None		Tactile Diff. [183]		Ours	
Train w/ DANN	✗	✓	✗	✓	✗	✓
master chef can	52.3%	41.2%,	58.7%	80.9%	66.6%	**92.0%**
sugar box	35.0%	35.0%	84.0%	71.0%	57.9%	**93.0%**
mustard bottle	41.0%	48.0%	34.0%	60.0%	63.0%	**71.0%**
tuna fish can	57.5%	68.1%	59.0%	**80.3%**	74.2%	75.7%
potted meat can	35.0%	43.0%	54.0%	76.0%	**87.0%**	84.0%
banana	2.4%	29.2%	29.2%	7.3%	56.0%	**78.0%**
pitcher base	40.6%	39.0%	85.9%	90.6%	64.0%	**98.4%**
bleach cleanser	38.0%	43.0%	39.0%	64.0%	69.0%	**84.0%**
bowl	40.3%	43.5%	62.9%	88.7%	53.2%	**90.3%**
power drill	3.1%	33.3%	50.0%	38.54%	23.9%	**60.4%**
Mean	34.7%	42.4%	55.6%	66.6%	61.6%	**81.9%**

Table 6.2 Results of the classification experiments: F1 score for each surface type.

Image transl.	None		Tactile Diff. [183]		Ours	
Train w/ DANN	✗	✓	✗	✓	✗	✓
flat	46.5%	28.6%	61.5%	81.3%	86.4%	**91.1%**
curve	3.6%	31.1%	18.6%	30.3%	45.2%	**73.5%**
edge	28.%	56.2%	78.1%	74.4%	58.1%	**83.2%**
corner	0.0%	0.0%	46.6%	26.3%	21.1%	**68.3%**

$$\text{F1-Score} = \frac{2 \cdot \text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}},$$

where *precision* and *recall* are defined as usual.

In order to investigate how the image translation and the DANN procedure contribute to the performance we repeated the experiments using different training configurations of the pipeline. As regards the image translation we trained the classifier using:

- simulated images, indicated as **"None"**;
- translated images from the Diffusion Model proposed in [183], indicated as **"Tactile Diffusion"**;
- translated images from the Diffusion Model proposed in this work, indicated as **"Ours"**.

For each configuration above, we repeated the experiments with the DANN procedure enabled or disabled.

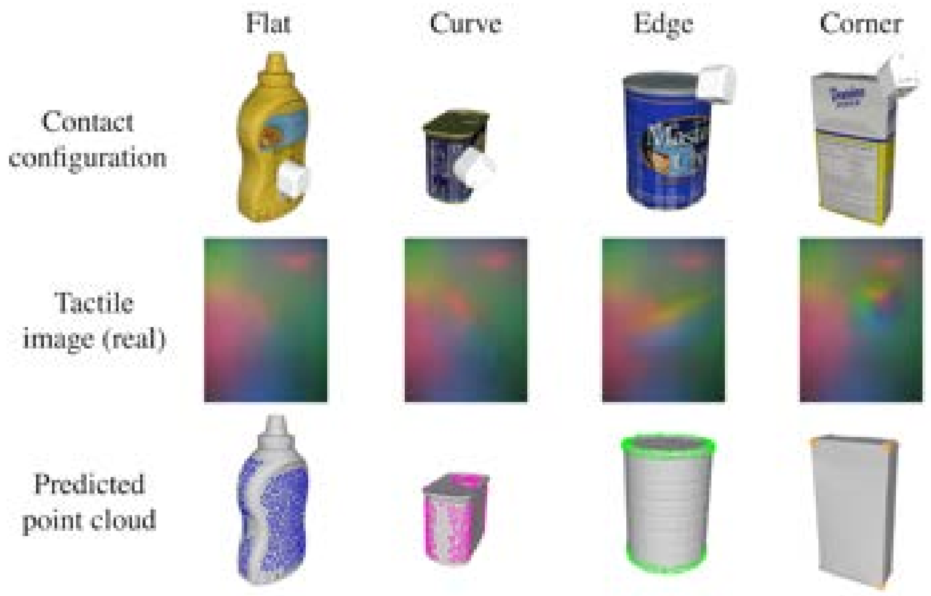


Figure 6.6 Visualization of the predicted surface type in terms of the object-sampled points having the same label as the predicted one.

The results in Table 6.1 shows that the best performance, except few cases, is achieved when using the proposed Diffusion Model as the translation mechanism combined with the DANN procedure.
Using DANN, the proposed Diffusion Model increases the accuracy by ≈ 40 points with respect to the “None” case and by ≈ 15 points with respect to the method [183].
As regards the DANN procedure, on average it helps increase the accuracy in all the configurations. The best improvement, of ≈ 20 points, is obtained in conjunction with the proposed Diffusion Model.
Table 6.2 shows that the proposed pipeline achieves the best performance on all surface types. Remarkably, both the DM and DANN are essential to improve performance on all classes, and specifically to gain acceptable performance on the class corners.
In Fig. 6.6 we present some qualitative results. Specifically, for each tactile image, we show the name of the predicted class and all the points sampled on the object mesh, as per Sec. 6.4.1, having the same label as the predicted one. These results indicate that a combination of the proposed classifier and the automatic labeling procedure of Sec. 6.4.1 is useful to provide hypotheses on the contact location of the sensor on the object surface.

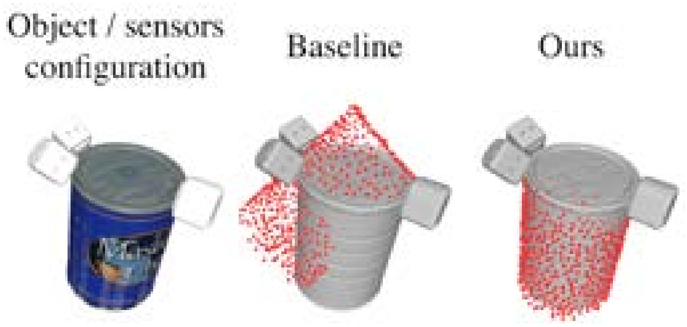


Figure 6.7 Visualization of the outcome of the 6D object pose estimation experiment using real tactile images.

6.5.2 Experiments on 6D object pose estimation

For these experiments we make use of the algorithm presented in [184] that estimates the 6D pose of an object in contact with N tactile sensors given input tactile images and the pose of the sensors from the robot proprioception.

Specifically, [184] extracts several hypotheses on the 3D location of each sensor on the object surface given the input tactile image. Then, it uses gradient-based optimization to fuse the hypotheses from all sensors resulting in a set of candidate 6D poses of the object.

In this work, we substitute the hypothesis extraction part, based on a convolutional autoencoder and a set of per-object databases of latent features, with the proposed classifier.

We choose to run the experiments using $N = 3$ sensors. As in [184], we considered four different configurations of sensors for every object under study.

We compare the performance against a purely geometric baseline, as done in [184]. The baseline skips the hypothesis extraction part and executes the optimization assuming that the N sensors might touch the object everywhere.

To evaluate the performance we compare the output pose with the ground truth pose in terms of the positional error and a variant of the ADI-AUC metric [198], introduced in [184], which reports on the rotational error. Remarkably, we report quantitative results of experiments run with real tactile images while [184] restricts the quantitative analysis to simulated data and only provides qualitative results on real images.

Table 6.3 shows that using the tactile feedback, in the form of the proposed classifier, helps halving the positional error and increasing the rotational metric by more than ten points, on average.

In Fig. 6.7 we show a sample outcome of the experiment and compare it against the baseline. The ground-truth pose is shown as the greyed-out mesh while the estimated pose as the red point cloud. As can be seen, although the pose estimated by the baseline is compatible with

Table 6.3 Results of the object pose estimation experiments: averaged positional error and ADI-AUC with threshold set to 2 cm.

Metric	Positional error (cm) $\downarrow$		ADI-AUC$_{2\text{cm}}$ (%) $\uparrow$	
Method	Baseline	Ours	Baseline	Ours
master chef can	2.97	**0.69**	93.54	**97.23**
sugar box	3.61	**3.12**	**93.75**	72.61
mustard bottle	2.97	**1.83**	69.83	**96.61**
tuna fish can	2.18	**1.07**	**96.08**	95.97
potted meat can	1.81	**1.26**	94.71	**96.63**
banana	6.72	**2.39**	48.49	**96.93**
pitcher base	6.71	**2.73**	25.0	**47.19**
bleach cleanser	4.97	**2.96**	70.34	**94.28**
bowl	**1.74**	2.08	**97.29**	96.27
power drill	8.36	**4.09**	67.77	**73.76**
Mean	4.21	**2.22**	75.68	**86.75**

the *position* of the contacts, two out of three sensors are in contact with the wrong part of the object. Instead, the pose estimated by our method is compatible with all contact positions and types. This further demonstrates the advantages of using the tactile feedback, in the form of the proposed classifier, for this task.

6.6 Limitations

The rigidity of the elastomer of the DIGIT sensor requires a more than moderate force, when interacting with objects, to highlight surface differences. Although this fact depends on the sensor itself, it might impact the effectiveness of our method if the contact forces are too weak.

We acknowledge that our method makes use of a Diffusion Model whose training and image-translation times are non-negligible. Nonetheless, one of the advantages of the proposed method is that it is trained on images without background, thus it can be used on different devices without a re-train.

6.7 Conclusion

In this work, we tackled the Sim2Real gap in the context of vision-based tactile sensors to classify local surfaces of objects. Our method combines a bilevel adaptation, at the image and feature level, with an automatic labeling procedure, allowing us to train the classifier

using a supervised approach while avoiding manual annotation. The extensive experiments on real data and the comparison against a state-of-the-art method validate the robustness and the effectiveness of our approach that we also applied successfully in the context of 6D object pose estimation from tactile data.
As future work, we plan to exploit our classification approach for several other robotic tasks and to study new adaptation mechanisms to further increase the classification accuracy.

Chapter 7

From Handheld to Unconstrained Object Detection: a Weakly-Supervised On-line Learning Approach

7.1 Context

In this contribution, we present a novel Weakly-Supervised Learning (WSL) approach that incorporates an Active Learning (AL) framework and a Self-Supervised Learning (SSL) strategy to bridge the gap between two domains for Object Detection in a robotic application. The large number of annotations required for high performance DL-based Object Detectors can be mitigated by using robots embodiment that can automatically acquire labeled training data via a natural interaction with a human showing the object of interest, handheld. However, learning solely from this data may introduce biases (because of the *domain shift*), and prevents adaptation to novel tasks. While WSL offers a well-established set of techniques to cope with these problems in general-purpose Computer Vision, its adoption in challenging robotic domains is still at a preliminary stage. In this work, we target the scenario of a robot trained in a teacher-learner setting to detect handheld objects. The aim is to improve detection performance in different settings by letting the robot explore the environment with a limited human labeling budget. We compare several techniques for WSL in detection pipelines to reduce model re-training costs without compromising accuracy, proposing solutions which target the considered robotic scenario. We show that the robot can improve adaptation to novel domains, either by interacting with a human teacher (Active Learning) and/or with an autonomous supervision (Self-Supervised Learning). We integrate our strategies into an on-line detection method, achieving efficient model update capabilities with few labels. We experimentally benchmark our method on challenging robotic object detection tasks under domain shift[1].

7.2 Introduction

In the state-of-the-art, Object Detection is typically addressed with DL-based approaches [199, 200] that achieve remarkable performance. Despite their high accuracy, they are constrained by requiring long training times and large annotated datasets, limiting their adoption in such applied settings where quick adaptation to novel tasks is required. In robotics, the embodiment of a robotic agent can be exploited to interact with the environment, including humans, to mitigate this burden and actively acquire training data. Regarding the interaction with humans, past work shows that a teacher-learner scenario can be exploited to automatically collect labeled images for object recognition [201] and detection [202]. Specifically, in those works the human teacher shows an object, while holding it in their hand, to the robot and 3D information is used to automatically collect the location information. However, while effective and allowing for a natural interaction, this approach supports limited generalization to novel, unseen, scenarios [202, 203]. A further possibility is to exploit robots ability to navigate and autonomously explore the environment, acquiring training images during operation. Such images come in streams and can carry useful information, eventually containing the objects of interest, but they are not labeled. *Weakly-supervised Learning* (WSL) [204], is a well-established general purpose Computer Vision framework which targets learning from partially-annotated datasets. However, despite initial work in robotic vision [205, 203] the robotic literature misses a thorough comparison that investigates advantages and limitations of existing techniques, especially in the context considered in this paper. For instance, in [203], the unlabeled images are processed with a pre-trained model to either select the hard ones and ask a human expert to help and annotate them (*Active Learning* (AL) [206]) or add the predictions of the easy ones to the training set (*Self-Supervised Learning* (SSL) [207, 204]). These frameworks allow for a natural interaction with the environment and the human teacher to improve the visual system and the work presented in [203] effectively reduces the amount of manual annotation, but it has some limitations. Firstly, the unsupervised data processing is *pool-based* [206], that is, all unlabeled images are evaluated before query selection. This is not suitable for a robotic system that is exploring the environment and needs to decide interactively whether to request annotations or not. To this aim, *stream-based*

[1]The code of the experiments is available at: https://github.com/hsp-iit/Weakly-supervised-online-detection

techniques [206] are preferable, because they allow to process images frame by frame and to make individual query decisions on-line. This strategy, however, might yield to lower accuracy since queries are constructed using limited information on the unlabeled set [206]. Moreover, the pre-trained detection method in [203] iterates multiple times over the unlabeled data, which, while allowing to refine the data selection, slows down learning. Finally, while succeeding in reducing the human effort required for refinement, [203] still needs a relatively high number of manual annotations, which prevents its adoption in on-line applications.

In this work, we study how WSL techniques can be used to exploit the robot interaction with the environment and the human teacher to update and improve performance of object detection models previously trained with data of handheld objects. We focus on the stream-based scenario with the aim of increasing the human labeling efficiency of weakly-supervised on-line object detection. Moreover, we consider the case in which only one pass over the unlabeled data is allowed. The main contributions of this work are as follows. We present and empirically evaluate several AL techniques for detection, typically used in general purpose computer vision. We compare *pool-based* and *stream-based* AL in challenging robotic scenarios and propose a solution to overcome limitations of the latter. We also consider the case where no human labeling is allowed for adaptation. Specifically, we investigate the domain shift effects occurring when using a model trained on data of handheld objects in different settings and how wrongly self-annotated data can degrade accuracy in those cases. Finally, we propose an SSL sampling method to overcome this problem and we empirically demonstrate that, in case no labeling is allowed, it can effectively improve model performance. This paper is organized as follows: we introduce WSL in Sec. 7.3 and we cover related work in Sec. 7.4. In Sec. 7.5, we present our efficient detection methods, which are analyzed and validated in Sec. 7.6. Sec. 7.7 presents some final remarks about this work.

7.3 Background

The supervised learning approach to object detection is centered on learning the detector function from an annotated dataset $D_{\text{train}} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{N}$ of images ($\mathbf{x}_i$) and corresponding bounding boxes and class annotations ($\mathbf{y}_i$). The methods described in Sec. 7.4.1 fall in this category. They contributed to a clear progress in detection accuracy and inference speed. However, they need expensively-annotated large-scale datasets to be optimized. This property does not meet the robotic requirement for a detector to adapt to a variety of tasks, potentially unknown a-priori, in a short time span. However, while large annotated datasets might not be available, plenty of unsupervised images are usually accessible to robots. In this context, a training set $D'_{\text{train}} = L \cup U$ is typically composed of a labeled subset $L = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{N}$ and an unlabeled subset $U = \{\mathbf{x}_i\}_{i=1}^{M}$. WSL allows the agent to select unsupervised images from U and acquire their labels semi-autonomously for updating the detector, minimizing human effort and improving accuracy. WSL includes several sub-classes of methods, depending on the label-acquisition mechanism [207, 204]. The most relevant for this work are Active and Self-Supervised Learning.

Active Learning. AL [206] interactively queries unsupervised examples for expert labeling to minimize human annotation and maximize accuracy. Unlabeled examples are chosen from U according to a *scoring function* and a *sampling strategy*. Their labels are then queried to an expert, and newly-annotated examples are added to L for training. If all images in U are accessible at selection time, sampling is referred to as *pool-based*. Otherwise, if only one candidate from U is accessible, sampling becomes a binary decision on keeping or dropping it and is called *stream-based*. The AL selection criterion we focus on is *uncertainty sampling*, which picks the examples the model is *least confident* about.

Self-Supervised Learning. In SSL [204], unlabeled images are annotated by the detector itself with no human intervention, propagating predicted labels to high-confidence regions of the input space by exploiting the geometry of the input data distribution. This technique is effective if the detector is not overconfident of its predictions and if the confidence threshold for propagating predicted labels is strict enough.

7.4 Related Work

7.4.1 Object Detection

Early approaches to object detection were based on feature dictionaries [208] or specific kinds of image descriptors [209]. Feature vectors were separately classified by supervised learning methods. Despite yielding limited accuracy, these approaches had the advantage of being parsimonious in terms of computations and dataset size. More recently, object detection experienced significant progress thanks to the introduction of DL-based methods. This determined clear improvements in terms of predictive performance, mainly due to the powerful representation capabilities of DNNs. Such approaches include two-stage detectors based on Region Proposal Networks (RPNs) [210] (like e.g. Faster R-CNN [210] and Mask R-CNN [199]) and related extensions [211, 212, 213]. These methods employ a deep network to perform (i) region candidates predictions, (ii) per-region feature extraction and (iii) region classification and refinement. Alternative end-to-end approaches include one-stage detectors, which replace the RPN with a fixed, dense grid of candidate bounding boxes. One such example is SSD [214, 215], achieving accuracies competitive with the RPN-based Faster R-CNN and high frame rate. Another one-stage method, RetinaNet [216], rebalances foreground and background examples through the so-called Focal Loss.

7.4.2 Efficient Object Detection for Robotics

Despite their high accuracy, the approaches described above typically require (i) long training time and (ii) large-scale annotated datasets for adaptation to novel tasks. These aspects limit their adoption in robotics.

Training time efficiency

A well-known issue of DL-based pipelines is that they suffer from *catastrophic forgetting* when optimized on new data [217]. This limitation implies retraining these models on the full dataset (both old and new data), causing long adaptation time. To address this issue, a recent work for robotic object detection leverages fast classifiers to enable on-line adaptation [218, 219]. Specifically, in [219], an efficient multi-stage pipeline is proposed by combining DL-based RPNs and feature extractors (namely, based on Faster R-CNN or Mask R-CNN) with large-scale Kernel classifiers [220, 221, 222]. According to this approach, the feature extractor is pre-trained off-line on a large representative dataset, yielding a powerful and transferable learned representation, which is kept fixed during on-line operation. The actual regions classification is performed integrating an efficient hard-negatives bootstrapping approach (the Minibootstrap [219]) with a set of FALKON classifiers [220, 221].

Labeling efficiency

Labeling efficiency is another key requirement for robotic object detection. The broad class of WSL methods [207, 204] provides a rich set of tools towards this goal in general purpose Computer Vision, in particular AL and SSL – introduced in Sec. 7.3. After successful applications to DL-based Object Classification [223, 224, 225], AL has been recently applied also to Object Detection [226, 227, 228]. For instance, recently, detection-specific image scoring functions (like e.g., *localization tightness* and *stability* [229]) have been proposed. Instead, when no further annotation is allowed to exploit the unsupervised samples, SSL techniques can be used. Similarly to AL, also SSL has been recently applied to object detection. For instance, in [230], SSL is employed for dataset augmentation and training object detectors. Moreover, the authors point out that vanilla SSL can degrade accuracy in presence of domain shift. We also observed the same issue in our robotic setting and we propose a simple yet effective solution in Sec 7.5.4. Recent approaches integrate both AL and SSL techniques into the same detection pipeline, such as Self-Supervised Sample Mining [231, 232] (SSM). SSM sorts unsupervised images into separate candidate sets for further AL and SSL processing, according to the predictive confidence scores of the underlying DL-based detection model [211]. Another related field in Computer Vision is *Unsupervised Domain adaptation* for object detection [233]. Specifically, the *pseudo-labeling* approach [234, 235] proposes to adapt a detection model to novel and unknown domains (i.e., datasets) by using confident model predictions as pseudo-ground truth. The aforementioned approaches have been proposed and benchmarked on general purpose Computer Vision datasets. However evaluation of WSL techniques on robotic scenarios is still at an initial stage (e.g., see [205, 203]). For instance, in [203], SSM is extended to enable on-line adaptive object detection for Robotics, by integrating the WSL sample selection strategy with the on-line object detection method [219]. However, [203] still requires a relatively large number of manual annotations, does not investigate the effect of severe domain shift in Self-Supervision and focuses on a *pool-based* processing. While showing encouraging results, all these limitations prevent its adoption in on-line applications. In this work, we present an empirical analysis of different general purpose Computer Vision AL and SSL techniques in a challenging robotic scenario, targeting a low annotation budget regime. We focus on how WSL techniques can be used to exploit the robot interaction with the environment and the human teacher to update and improve performance of object detection models previously trained with data of handheld objects. Moreover, we propose solutions to overcome the aforementioned limitations, improving the AL performance and addressing the SSL failure cases under domain shift, increasing overall labeling efficiency.

7.5 Methods

In this work, a robot is asked to detect a set of object instances in an unconstrained environment (referred to as TARGET). A first detection model is trained during a brief interaction with a human, in a teacher-learner scenario, like e.g. in [202] where objects are handheld (the TARGET-LABELED). Then, the robot autonomously explores the environment, acquiring a stream of images in a new setting, where automatic annotation is not possible. Therefore, these images are not labeled (TARGET-UNLABELED) and are used to adapt the detector on-line exploiting the robot interaction with the environment and the human teacher. In the next sections, we present the proposed pipeline (Sec. 7.5.1) and the learning protocol

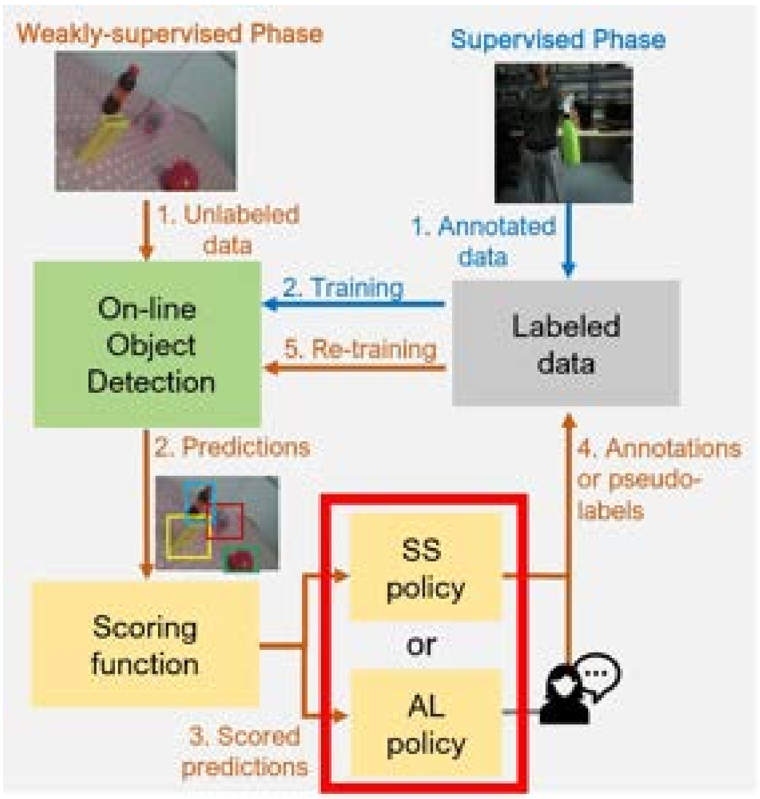


Figure 7.1 Overview of the proposed pipeline. Refer to Sec. 7.5.1 for details.

(Sec. 7.5.2). Then, we present all the considered AL and SSL techniques and the proposed approaches (Sec. 7.5.3 and 7.5.4, respectively).

7.5.1 Pipeline Description

The proposed WSL pipeline (see Fig. 7.1) is composed of four main modules: (i) the *On-line Object Detection*, (ii) the *Scoring function*, (iii) the *AL Selection policy*, and (iv) the *SS Selection policy*.

On-line Object Detection (OOD)

For this module, we follow the method proposed in [219], but considering the implementation presented in [236] and [237]. This is an on-line learning approach consisting of two stages: (i) region proposals and feature extraction, and (ii) region classification and bounding-box refinement. The first stage relies on layers from Mask R-CNN [199] (specifically, the convolutional layers, the RPN [238] and the RoI Align layer [199]). In particular, this part is used to extract a set of Regions of Interest (RoIs) from an image and encode them into a set of features. The second stage is composed of a set of FALKON [220] binary classifiers (one for each class of the TARGET) and Regularized Least Squares (RLS) [239], respectively for the classification and the refinement of the proposed RoIs. Classifiers are trained with an approximate bootstrapping approach, called Minibootstrap [219], which addresses the well-known issue of background-foreground class imbalance in object detection [240], while maintaining a short training time. In this work, the adoption of OOD permits to achieve a convenient speed/accuracy trade-off, since it allows to maintain a competitive accuracy with other DCNN-based approaches with a fraction of the optimization time required (seconds or minutes) [219, 236].

Scoring function

This function assigns a confidence score to the predictions for the images in the TARGET-UNLABELED. This score is then used by the AL and SS Selection policies to decide which images need to be manually annotated or can be considered as pseudo-ground truth. For this part, we employ the Cross-Image Validation (CIV) proposed in SSM [231]. CIV stitches predicted image patches from the TARGET-UNLABELED on random images, sampled from TARGET-LABELED. Then, it executes the detector on the stitched images and computes a *consistency score* from the obtained confidence scores [231].

AL and SS Selection policies

Given the predicted detections obtained by the OOD and the *consistency score* computed by the Scoring function, these two policies decide whether an image of the TARGET-UNLABELED is queried for annotation or the predicted detections are confident enough to be used for self-supervision. Our main contribution relies on these last two components. Firstly, we target a stream-based scenario, since it is more suitable for on-line applications. Secondly, we consider a robotic setting with low annotation budget and a large domain shift of the TARGET-UNLABELED with respect to the TARGET-LABELED. Specifically, for the *AL Selection policy*, we consider several AL techniques, comparing their performance on the considered robotic setting and proposing a solution to enforce diversity during sampling. The adopted AL baselines and the proposed solution are listed in Sec. 7.5.3. Instead, for the *SS Selection policy*, we consider a stream-based baseline and a novel strategy to overcome issues caused by the domain shift, both described in Sec. 7.5.4. Finally, another major difference with respect to previous work [203] is that we consider the case in which only one pass over the TARGET-UNLABELED data is allowed, while typically in standard Computer Vision, and also in [203], an iterative process is used. This aspect is crucial for speeding up WSL. However, it makes detector refinement more challenging.

7.5.2 Learning Protocol

The learning process is divided into: (i) *Supervised phase* (represented by the light blue arrows in Fig. 7.1), and (ii) *Weakly-supervised phase* (represented by the orange arrows in Fig. 7.1). Both phases rely on pre-trained Mask R-CNN's weights as feature extractor for the OOD. Those weights remain fixed, while model training and adaptation is performed by optimizing on the new data only the second stage of the OOD, i.e., (i) the FALKON classifiers with the Minibootstrap technique and (ii) the RLS box-refinement model (see Sec. 7.5.1 for details). The *Supervised phase* is performed within a few seconds of interaction with a human on the TARGET-LABELED, yielding a first detection model (the *seed model*). In this phase the human shows the objects to the robot, handling them in their hand and annotations are automatically collected. Then, in the *WSL phase*, the SSL pseudo-ground truth and AL queries are selected from the TARGET-UNLABELED as described in Sec. 7.5.1, using the *seed model*'s confidence scores. Finally, they are added to the dataset which is used to re-train the on-line detector.

7.5.3 Active Learning Strategies

For AL selection, we considered both (i) stream-based approaches, which are the focus of this work, being suited to robotic scenarios, and (ii) pool-based ones.

A simple, yet often effective, pool-based strategy is to sample uniformly at random the images with a confidence score below a threshold (`Uniform random` in Sec. 7.6). Another diversity sampling strategy is to execute k-means clustering [239] on the image-level features and select the resulting cluster centers (`K-means-based AL` in Sec. 7.6). In our analysis, we report results for both strategies.

In stream-based AL settings, a simple selection strategy involves confidence score thresholding followed by coin flipping [231] for implementing uncertainty and diversity sampling, respectively (`coin-flip AL` in Sec. 7.6). Another, more practical, solution is to exploit temporal coherence in image sequences to enforce sampling diversity [241]. Leveraging temporal coherence is particularly suitable for on-line robotic tasks, since data, coming in streams, needs to be acquired sequentially and is therefore highly temporally correlated. To this aim, we consider the `Fixed temporal window` strategy, which employs a temporal window of fixed size Δ so that if frame t is selected, any other frame within $[t-\Delta, t+\Delta]$ can

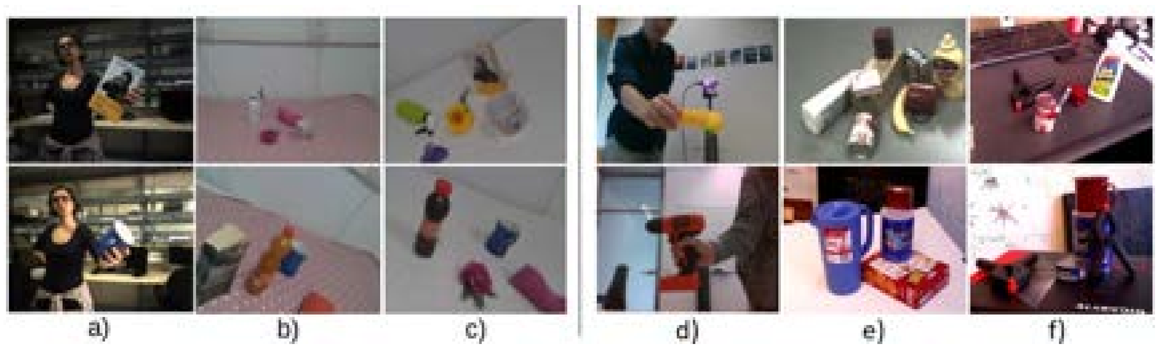


Figure 7.2 Example images of the datasets used for this work: **a)** ICWT dataset; **b)** POIS in TABLE-TOP dataset; **c)** WHITE in TABLE-TOP dataset; **d)** HO-3D; **e)** YCB-Video training set, **f)** YCB-Video test set.

no longer be considered for selection. While enforcing diversity, this strategy, by using a fixed Δ, does not take into account: (i) the exploration session duration, that is, the size n_U of TARGET-UNLABELED, which might be known a-priori even in stream-based scenarios, and (ii) the available manual annotation budget k. We show in Sec. 7.6.2 that this results in poor performance for low k when the TARGET-UNLABELED is redundant.

To overcome this limitation, we propose to use an adaptive temporal window size, defined as

$$\Delta_{n_U,k} = \frac{n_U \cdot \alpha}{k}$$

and referred to as `Adaptive temporal window` in Sec. 7.6. This strategy allows to tailor the strictness (window size) of the temporal diversity-enforcing sampling to the overall amount of available unsupervised data n_U, while at the same time ensuring to make full use of the available budget k. For instance, given a budget k, the adaptive window size grows linearly with n_U in order to cover the entire duration of the exploration session. $\alpha \in (0,k) \subseteq \mathbb{R}$ is a hyperparameter accounting for the proportion of AL candidates with respect to n_U, which is unknown a priori.

7.5.4 Self-Supervised Learning Strategies

For SS selection, we consider two stream-based baselines. The first is the `SS baseline`, which selects all the images passing CIV as pseudo ground truth. However, we show in Sec. 7.6.3 that under domain shift this leads to model degradation due to the abundance of false negatives. For this reason, in this work, we propose a more conservative strategy, namely `SS pos. only`, which only selects positive predictions and leaves out negative ones. In Sec. 7.6.3, we show that our approach successfully counteracts severe model degradation.

7.6 Experiments

The objective of our experiments is to evaluate the performance of the presented WSL techniques in improving detection performance under domain shift. Specifically, we consider the scenario of a robot previously trained with human interaction to detect handheld objects. We aim to generalize to a different setting (i.e., a table top) by exploiting the unlabeled data collected by the robot during autonomous exploration (*Weakly-Supervised phase* in Sec. 7.5.2).

7.6.1 Experimental Setup

For the OOD, the weights of the Feature extractor are learned by training Mask R-CNN on the MS COCO [242] dataset. ResNet50 [243] has been considered as Mask R-CNN's convolutional backbone (we use the available pre-trained Mask R-CNN weights[2]). During *Supervised* and *Weakly-Supervised phases*, the feature extractor is fixed, while the FALKON classifiers and RLS are updated as explained in Sec. 7.5.2 (we relied on [219] for hyper-parameters selection). This allows to achieve a training time of few seconds or minutes for each learning step.
Given the aforementioned target scenario, in our experiments we consider two different cases of domain adaptation from handheld (*Supervised phase*) to table-top objects (*Weakly-Supervised phase*). Specifically, we adapt (i) from iCubWorld Transformations [201] (iCWT) to a set of sequences depicting a subset of iCWT's objects on a table-top (TABLE-TOP) and (ii) from HO-3D [244] to YCB-Video [245].

From iCWT to TABLE-TOP (iCW domain)

iCWT contains images for 200 handheld objects. Each object is demonstrated by a human teacher to the robot (as in [202]) and is acquired with different sequences representing specific viewpoint transformations: 2D rotation (2D ROT), generic rotation (3D ROT), translation (TRANSL), scaling (SCALE) and all transformations (MIX) (see [201]). For the *Supervised phase*, we employ a subset of the iCWT, considering 21 of the total 200 objects. All the transformations, except from MIX, are considered, resulting in a TARGET-LABELED of size n_L ~6K. The TABLE-TOP depicts the same 21 objects randomly placed on a table with two different tablecloths: (i) pink/white pois (POIS) and (ii) white (WHITE). The two datasets

[2]https://github.com/facebookresearch/maskrcnn-benchmark/blob/master/MODEL_ZOO.md

contain the same objects, but with an important domain shift: iCWT frames include the hand of the teacher, whereas TABLE-TOP has different backgrounds and light conditions and depicts objects on a table. Refer to Fig. 7.2 for a visual representation of the domain shift. For the *Weakly-Supervised phase*, we consider the WHITE sequence as TARGET-UNLABELED while we leave the POIS sequence as test set to evaluate performance. These two sets are respectively of size $\sim$2K and $\sim$1K.

From HO-3D to YCB-Video (YCB domain)

Similarly, in HO-3D and YCB-Video, objects from the YCB [246] dataset are presented handheld by a human and in table-top sequences, respectively. Specifically, YCB-Video presents sequences for 21 objects while in HO-3D a subset of 9 of those objects are considered. Note that for our experiments we do not consider the labels for the remaining 12 objects in YCB-Video. For the *Supervised phase*, we take from HO-3D at most four sequences for each object resulting in a TARGET-LABELED of size n_L $\sim$20K. For the *Weakly-Supervised phase*, we consider a set of $\sim$11.3K frames obtained by extracting one image every ten from the total 80 training video sequences available in the YCB-Video. As test set, instead, we consider the $\sim$3K keyframe [245] images chosen from the remaining 12 sequences in the YCB-Video.

Evaluation metrics

We report performance in terms of mAP (mean Average Precision) at the IoU (Intersection over Union) threshold set to 0.5, as defined for Pascal VOC 2007 (see [247]). Specifically, we repeat each experiment for three trials and we present the results, reporting the mean and the standard deviation of the obtained accuracy[3].

7.6.2 Active Learning Sampling Strategy Evaluation

In this section, we compare the AL techniques described in Sec. 7.5 considering different manual annotation budgets for both iCW and YCB domains. To this aim, we report in Fig. 7.3a and b the mAP trends obtained by increasing the AL query budget during the *Weakly-Supervised phase*. Specifically, we report in orange shades the performance obtained by the pool-based strategies (namely, `k-means-based AL` and `Uniform random` from Sec. 7.5), and in green shades the stream-based ones (namely, `Coin-flip AL`, `Fixed`

[3]All experiments have been executed on a machine equipped with Intel Xeon E5-2690 v4 CPUs @2.60GHz, and an NVIDIA Tesla P100 GPU.

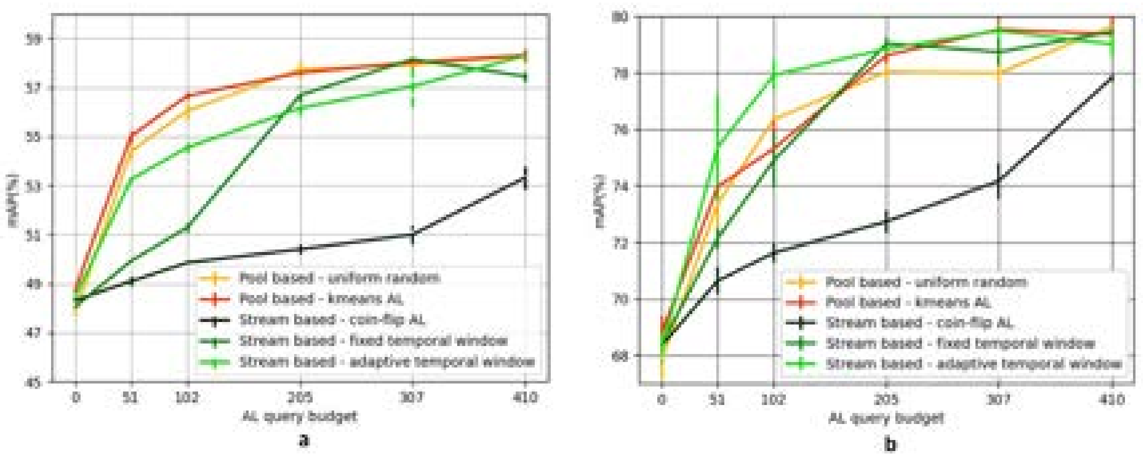


Figure 7.3 mAP comparison of pool-based (orange) and stream-based (green) AL strategies with varying query budgets for iCW (a) and YCB (b) domains.

`temporal window`, and the proposed `Adaptive temporal window` from Sec. 7.5). We empirically set the fixed temporal window size as $\Delta = 6$ and the adaptive temporal window hyperparameter as $\alpha = 0.5$ for the iCW domain and $\alpha = 0.4$ for the YCB domain. As it can be observed in Fig. 7.3a, for iCW domain the pool-based methods achieve the best mAP trends. Notably, we observe that the `Uniform random` baseline is almost as effective as k`-means based-AL` and they both present an early steep slope for limited manual annotation budgets. These two aspects are due to the fact that the considered TABLE-TOP dataset in the iCW domain, contains sequences of similar (and thus redundant) frames which need to be properly filtered during data selection. This aspect of the dataset is also the main cause for the poor performance obtained by the two stream-based techniques: `Coin-flip AL` and `Fixed temporal window`, for low numbers of manual annotations. Indeed, while being more suited for a robotic application, by reasoning only on a frame-by-frame fashion, they lack global information on the whole data distribution, which turns out to be a critical drawback especially for limited manual annotation budgets. However, for higher budgets, the `Fixed temporal window` baseline achieves accuracies closer to the pool-based ones. Finally, the proposed `Adaptive temporal window` presents the best mAP trend, among the stream-based approaches, especially for low annotation budgets and it closely matches the pool-based ones. On the contrary, as it can be observed in Fig. 7.3b, for the YCB domain the pool-based methods, the `Fixed` and the `Adaptive temporal window` present similar mAP trends. Specifically, the proposed `Adaptive temporal window` has the steepest slope. This is due to the fact that the YCB-Video dataset, differently from the TABLE-TOP, presents less redundant sequences and a smarter data selection based on temporal coherence provides

Table 7.1 Results obtained by `SS baseline` (2nd column) and `SS positives` (3rd column) for large (1st row) and small (2nd row) domain shift from the supervised (1st column) to the Weakly-Supervised phase.

	Sup. phase (mAP(%))	**SS baseline (mAP(%))**	**SS pos. only (mAP(%))**	**SS samples**
Large DS	48.8 ± 0.3	37.9 ± 1.8	50.9 ± 0.06	$\sim 12\%$
Small DS	40.3 ± 0.9	47.1 ± 0.1	46.6 ± 0.2	$\sim 35\%$

the best performance.
Finally, it is important to note that the proposed `Adaptive temporal window` stream-based approach achieves significantly higher mAP values, than other stream-based techniques, for low annotation budgets for both domains. This makes it the most successful stream-based approach in such regime, which is the target of the presented work.

7.6.3 Self-Supervised Learning Evaluation

In this section, we investigate the impact of domain shift from handheld objects to table-top datasets when no labeling is allowed (i.e., SSL). To this aim, we report in Tab. 7.1 the results of applying the `SS baseline` (as defined in Sec. 7.5) in the two following scenarios:

- **Large domain shift** (*Large DS* in Tab. 7.1). We consider the scenario in which the TARGET-UNLABELED presents a completely different setting (i.e., a table top) with respect to the TARGET-LABELED (i.e., hand-held). For this, we used the iCW domain of Sec. 7.6.1.

- **Small domain shift** (*Small DS* in Tab. 7.1). Here, the TARGET-LABELED and TARGET-UNLABELED present similar conditions. The only difference in the latter one is that the objects are presented, unlabeled, with different view poses. For this, we considered as TARGET a 30-object identification task from iCWT. For each object, we then use the TRANSL sequence (~2K images) as TARGET-LABELED and the union of the 2D ROT, 3D ROT, and SCALE sequences (~6K images) as the TARGET-UNLABELED. We test on the MIX sequences of all objects (~4.5K images).

Note that, in this experiment, we use the only iCW domain because the explicit sub-division in different viewpoint transformations of iCWT allows to control the dataset split in TARGET-LABELED and TARGET-UNLABELED such that they present similar, but not identical, conditions. This allows to precisely identify the *Small DS* setting.

Tab. 7.1 reports the results obtained in both cases. For each row, we report the mAP (represented as mean and standard deviation of the different repetitions) after the *Supervised phase* (first column) and after the *Weakly-Supervised phase* for both `SS baseline` and `SS pos. only` (second and third columns). Moreover, in the fourth column we report the average percentage of samples selected by the SS process over the total. As it can be observed, adding self-supervised data with `SS baseline`, with small domain shift, results in an improvement in accuracy. On the contrary, with a larger domain shift, it leads to a significant accuracy deterioration. A reason for this phenomenon can be identified by analyzing the pseudo-ground truth generated by the SS process. We report in Fig. 7.4 some representative images depicting in green the region proposal candidates classified as background by the detection system and that are therefore added as negative samples to the dataset by the `SS baseline`. The actual detections which instead are considered as positive samples in the SS process are shown in red. It can be noticed that, with large domain shift, only few objects are correctly detected and therefore added to the training set as positives, while most others are false negatives which are automatically annotated as background samples. Clearly, retraining the detection model with such a poorly-labeled dataset leads to the sharp performance decay shown in Tab. 7.1. This confirms similar findings from the literature [234], in the considered setting. Note that, we empirically noticed that lowering the confidence threshold used to determine a positive prediction is not suitable since, while not ensuring less false negatives, it leads to imprecise predictions, with a similar negative effect on the subsequent training. In Sec. 7.5, we introduce the `SS positives only` to address this issue. This more conservative strategy includes only the regions predicted as positive in the SS dataset, while the others are filtered away, avoiding adding false negatives. Third column in Tab. 7.1 shows that this strategy, does not modify the basline in case of Small DS where SS data is already reliable. However, for Large DS, not only effectively removes wrong labels from the dataset, recovering from the two-digits accuracy decay, but also successfully yields a performance improvement of $\sim$2 points. Moreover, it allows to drastically reduce the standard deviation of the obtained accuracy, from 1.8% to 0.06%, being less sensitive to statistical fluctuations. This demonstrates that, when no human manual annotation is allowed, a robot trained to detect handheld objects can explore the new domain, self-annotating the newly collected data and improving detection performance.

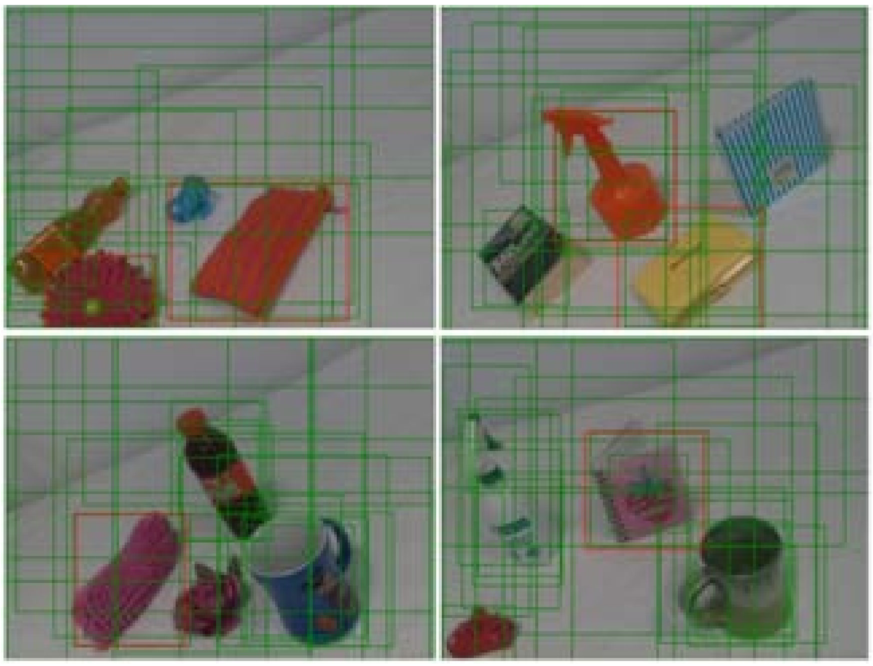


Figure 7.4 Predictions on TARGET-UNLABELED images before model update, chosen by the `SS Baseline` as positives (red) and negatives (green).

7.7 Conclusions

In this paper, we target the scenario of a robot trained with human interaction to detect handheld objects, aiming to improve detection performance in different settings with autonomous exploration and limited human intervention. We empirically demonstrate that general purpose WSL techniques are unsuitable for challenging robotic scenarios and we propose solutions to both (i) enforce diversity sampling for AL queries and (ii) improve strong positives selection for SSL under severe domain shift. Finally, we build on previous work [203], presenting and empirically evaluating a stream-based weakly-supervised on-line object detection pipeline for Robotics, which exploit the robot interaction with the environment and the human teacher to update and improve performance of the visual system. It significantly alleviates the annotation burden for on-line model adaptation to novel settings while maximizing accuracy.

Chapter 8

Conclusion

In this project, we have conducted a detailed analysis of various aspects of Transfer Learning, with a particular focus on fine-tuning, domain adaptation (including UDA and SF-UDA), as well as their practical applications.

The foundational analysis, as outlined in Chapter 3, provides empirical evidences and strategic guidance relevant for advancing the field of SF-UDA. This extensive empirical study includes a variety of methods, architectures, and pre-training strategies. However, we acknowledge the need for further research. Future work could extend this study to incorporate additional SF-UDA methods, diverse normalization layers, and extend the scope to include other vision tasks such as object detection and semantic segmentation. Furthermore, an exploration of SF-UDA methods for Instruction Tuning [248] of Large Language Models, for specific NLP tasks, would be of great interest for the community.

In Chapter 4, we introduced Trust and Balance (TAB), a novel and effective SF-UDA approach for image classification. Through numerous ablation studies of TAB, we identified potential areas for future improvement. An exploration of its applicability to semantic segmentation is also worth pursuing.

In Chapter 5, we considered different Transfer Learning pipelines, achieving state-of-the-art results in plankton image classification. Our findings on the choice of pre-training datasets open up avenues for future research. In particular we believe that possible extensions of this work may include: exploration of different fine-tuning strategies and Transfer Learning pipelines, as well as the application of these methodologies to other biological domains beyond plankton. Particularly, it would be interesting to assess whether ImageNet pre-training remains the optimal choice compared to large-scale in-domain pre-training. This could clarify the value of acquiring domain-specific expensive labeled data versus its potential lack of impact on final performance, as suggested in our study.

Finally, in Chapters 6 and 7, we presented specific applications of Transfer Learning pipelines in robotics, leveraging Denoising Diffusion Models, UDA approaches, and Active Learning. These chapters demonstrate how various techniques can be integrated into effective pipelines for processing real-world data.

Discussion

The primary objective of this dissertation is to explore a variety of subtopics and techniques within the broader domain of Transfer Learning, aiming to enhance data efficiency in Deep Learning across diverse scenarios. The selection of a particular technique for any given real-world application hinges on specific requirements and the availability of data within that context. In numerous instances, it is feasible to employ multiple techniques in a synergistic manner. For example, the methodology described in Chapter 6 incorporates an image-to-image style transfer process (a SF-UDA technique), alongside a feature alignment phase that utilizes a UDA algorithm. Whenever an application permits some level of labeling or human interaction, Active Learning strategies, such as those detailed for Object Detection in Chapter 7, can be included in the pipeline to enhance outcomes further.

Moreover, in light of the recent advancements in Large Vision and Language Models, it is important to acknowledge their potential in generating samples or aiding the adaptation process of smaller models across the various scenarios discussed in this dissertation. This integration should not be perceived as a constraint of our proposed methodologies but rather as complementary approaches that, when combined, can lead to superior performance with minimal task-specific data. A critical direction for future research is investigating how large-scale generative multimodal models can further refine data efficiency within transfer learning pipelines.

Summary and Final Remarks

This dissertation comprehensively examines numerous elements and strategies of contemporary Transfer Learning in Deep Learning. Through systematic analysis of both algorithmic and design choices, we have introduced novel algorithms and pipelines tailored to specific problems. We believe that our analyses and findings will significantly contribute to a deeper understanding of Transfer Learning, serving as a valuable resource for practitioners and guiding future research.

References

[1] Samira Pouyanfar et al. “A survey on deep learning: Algorithms, techniques, and applications”. In: *ACM Computing Surveys (CSUR)* 51.5 (2018), pp. 1–36.

[2] Athanasios Voulodimos et al. “Deep learning for computer vision: A brief review”. In: *Computational intelligence and neuroscience* 2018 (2018).

[3] Daniel W Otter, Julian R Medina, and Jugal K Kalita. “A survey of the usages of deep learning for natural language processing”. In: *IEEE transactions on neural networks and learning systems* 32.2 (2020), pp. 604–624.

[4] Yiran Chen et al. “A survey of accelerator architectures for deep neural networks”. In: *Engineering* 6.3 (2020), pp. 264–274.

[5] Jia Deng et al. “Imagenet: A large-scale hierarchical image database”. In: *2009 IEEE conference on computer vision and pattern recognition*. Ieee. 2009, pp. 248–255.

[6] Joel Hestness et al. “Deep learning scaling is predictable, empirically”. In: *arXiv preprint arXiv:1712.00409* (2017).

[7] Xiaohua Zhai et al. “Scaling vision transformers”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 12104–12113.

[8] Jonathan S Rosenfeld et al. “A constructive prediction of the generalization error across scales”. In: *arXiv preprint arXiv:1909.12673* (2019).

[9] Jared Kaplan et al. “Scaling laws for neural language models”. In: *arXiv preprint arXiv:2001.08361* (2020).

[10] Ben Sorscher et al. “Beyond neural scaling laws: beating power law scaling via data pruning”. In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 19523–19536.

[11] Utkarsh Sharma and Jared Kaplan. “Scaling laws from the data manifold dimension”. In: *The Journal of Machine Learning Research* 23.1 (2022), pp. 343–376.

[12] Yasaman Bahri et al. “Explaining neural scaling laws”. In: *arXiv preprint arXiv:2102.06701* (2021).

[13] Marius Cordts et al. “The cityscapes dataset for semantic urban scene understanding”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 3213–3223.

[14] Zheyan Shen et al. “Towards out-of-distribution generalization: A survey”. In: *arXiv preprint arXiv:2108.13624* (2021).

[15] Xingchao Peng et al. “Visda: The visual domain adaptation challenge”. In: *arXiv preprint arXiv:1710.06924* (2017).

[16] Fuchao Yu, Xianchao Xiu, and Yunhui Li. “A survey on deep transfer learning and beyond”. In: *Mathematics* 10.19 (2022), p. 3619.

[17] Garrett Wilson and Diane J Cook. “A survey of unsupervised deep domain adaptation”. In: *ACM Transactions on Intelligent Systems and Technology (TIST)* 11.5 (2020), pp. 1–46.

[18] Clemens-Alexander Brust, Christoph Käding, and Joachim Denzler. “Active learning for deep object detection”. In: *arXiv preprint arXiv:1809.09875* (2018).

[19] Vladimir Vapnik. “Principles of risk minimization for learning theory”. In: *Advances in neural information processing systems* 4 (1991).

[20] Fengxiang He and Dacheng Tao. “Recent advances in deep learning theory”. In: *arXiv preprint arXiv:2012.10931* (2020).

[21] Ruslan Abdulkadirov, Pavel Lyakhov, and Nikolay Nagornov. “Survey of Optimization Algorithms in Modern Neural Networks”. In: *Mathematics* 11.11 (2023), p. 2466.

[22] Jan Kukačka, Vladimir Golkov, and Daniel Cremers. “Regularization for deep learning: A taxonomy”. In: *arXiv preprint arXiv:1710.10686* (2017).

[23] Connor Shorten and Taghi M Khoshgoftaar. “A survey on image data augmentation for deep learning”. In: *Journal of big data* 6.1 (2019), pp. 1–48.

[24] Z Shen et al. “Towards out-of-distribution generalization: A survey. arXiv 2021”. In: *arXiv preprint arXiv:2108.13624* ().

[25] Alexander Kolesnikov et al. “Big transfer (bit): General visual representation learning”. In: *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part V 16*. Springer. 2020, pp. 491–507.

[26] Minyoung Huh, Pulkit Agrawal, and Alexei A Efros. “What makes ImageNet good for transfer learning?” In: *arXiv preprint arXiv:1608.08614* (2016).

[27] Linda Studer et al. “A comprehensive study of imagenet pre-training for historical document image analysis”. In: *2019 International Conference on Document Analysis and Recognition (ICDAR)*. IEEE. 2019, pp. 720–725.

[28] Ning Ding et al. “Parameter-efficient fine-tuning of large-scale pre-trained language models”. In: *Nature Machine Intelligence* 5.3 (2023), pp. 220–235.

[29] Jason Yosinski et al. “How transferable are features in deep neural networks?” In: *Advances in neural information processing systems* 27 (2014).

[30] Brian Chu et al. “Best practices for fine-tuning visual classifiers to new domains”. In: *Computer Vision–ECCV 2016 Workshops: Amsterdam, The Netherlands, October 8-10 and 15-16, 2016, Proceedings, Part III 14*. Springer. 2016, pp. 435–442.

[31] Chen Sun et al. “Revisiting unreasonable effectiveness of data in deep learning era”. In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 843–852.

[32] Ashish Vaswani et al. “Attention is all you need”. In: *Advances in neural information processing systems* 30 (2017).

[33] Alexey Dosovitskiy et al. “An image is worth 16x16 words: Transformers for image recognition at scale”. In: *arXiv preprint arXiv:2010.11929* (2020).

[34] Andreas Steiner et al. “How to train your vit? data, augmentation, and regularization in vision transformers”. In: *arXiv preprint arXiv:2106.10270* (2021).

[35] Tal Ridnik et al. “Imagenet-21k pretraining for the masses”. In: *arXiv preprint arXiv:2104.10972* (2021).

[36] Jie Gui et al. “A survey of self-supervised learning from multiple perspectives: Algorithms, theory, applications and future trends”. In: *arXiv preprint arXiv:2301.05712* (2023).

[37] Kaiming He et al. “Momentum contrast for unsupervised visual representation learning”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 9729–9738.

[38] Xinlei Chen et al. “Improved baselines with momentum contrastive learning”. In: *arXiv preprint arXiv:2003.04297* (2020).

[39] Kaiming He et al. “Masked autoencoders are scalable vision learners”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2022, pp. 16000–16009.

[40] Hangbo Bao et al. “Beit: Bert pre-training of image transformers”. In: *arXiv preprint arXiv:2106.08254* (2021).

[41] Mathilde Caron et al. “Emerging properties in self-supervised vision transformers”. In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2021, pp. 9650–9660.

[42] Maxime Oquab et al. “Dinov2: Learning robust visual features without supervision”. In: *arXiv preprint arXiv:2304.07193* (2023).

[43] Xiangli Yang et al. “A survey on deep semi-supervised learning”. In: *IEEE Transactions on Knowledge and Data Engineering* (2022).

[44] Yishay Mansour, Mehryar Mohri, and Afshin Rostamizadeh. “Domain adaptation: Learning bounds and algorithms”. In: *22nd Conference on Learning Theory, COLT 2009*. 2009.

[45] Shai Ben-David et al. “A theory of learning from different domains”. In: *Machine learning* 79 (2010), pp. 151–175.

[46] Ievgen Redko et al. “A survey on domain adaptation theory: learning bounds and theoretical guarantees”. In: *arXiv preprint arXiv:2004.11829* (2020).

[47] Yaroslav Ganin et al. “Domain-adversarial training of neural networks”. In: *The journal of machine learning research* 17.1 (2016), pp. 2096–2030.

[48] Guoliang Kang et al. “Contrastive adaptation network for unsupervised domain adaptation”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 4893–4902.

[49] Yuchen Zhang et al. “Bridging theory and algorithm for domain adaptation”. In: *International conference on machine learning*. PMLR. 2019, pp. 7404–7413.

[50] Jaemin Na et al. “Fixbi: Bridging domain spaces for unsupervised domain adaptation”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2021, pp. 1094–1103.

[51] Hongyi Zhang et al. “mixup: Beyond empirical risk minimization”. In: *arXiv preprint arXiv:1710.09412* (2017).

[52] Poojan Oza et al. “Unsupervised domain adaptation of object detectors: A survey”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2023).

[53] Marco Toldo et al. “Unsupervised domain adaptation in semantic segmentation: a review”. In: *Technologies* 8.2 (2020), p. 35.

[54] Jonathan Ho, Ajay Jain, and Pieter Abbeel. “Denoising diffusion probabilistic models”. In: *Advances in neural information processing systems* 33 (2020), pp. 6840–6851.

[55] Ilja Kuzborskij and Francesco Orabona. “Stability and hypothesis transfer learning”. In: *International Conference on Machine Learning*. PMLR. 2013, pp. 942–950.

[56] Jian Liang, Dapeng Hu, and Jiashi Feng. “Do we really need to access the source data? source hypothesis transfer for unsupervised domain adaptation”. In: *International conference on machine learning*. PMLR. 2020, pp. 6028–6039.

[57] Shiqi Yang, Shangling Jui, Joost van de Weijer, et al. “Attracting and dispersing: A simple approach for source-free domain adaptation”. In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 5802–5815.

[58] Shiqi Yang et al. “Exploiting the intrinsic neighborhood structure for source-free domain adaptation”. In: *Advances in neural information processing systems* 34 (2021), pp. 29393–29405.

[59] Haifeng Xia, Handong Zhao, and Zhengming Ding. “Adaptive adversarial network for source-free domain adaptation”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 9010–9019.

[60] Rui Li et al. “Model adaptation: Unsupervised domain adaptation without source data”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 9641–9650.

[61] Zhiqi Yu et al. “A Comprehensive Survey on Source-free Domain Adaptation”. In: *arXiv preprint arXiv:2302.11803* (2023).

[62] Fuzhen Zhuang et al. “A Comprehensive Survey on Transfer Learning”. In: *Proceedings of the IEEE* 109.1 (2021), pp. 43–76.

[63] Donghyun Kim et al. “A broad study of pre-training for domain generalization and adaptation”. In: *European Conference on Computer Vision*. Springer. 2022, pp. 621–638.

[64] Shai Ben-David et al. “Analysis of representations for domain adaptation”. In: *Advances in Neural Information Processing Systems*. 2006.

[65] Qiao Liu and Hui Xue. “Adversarial Spectral Kernel Matching for Unsupervised Time Series Domain Adaptation.” In: *IJCAI*. 2021, pp. 2744–2750.

[66] Alan Ramponi and Barbara Plank. “Neural unsupervised domain adaptation in NLP—a survey”. In: *arXiv preprint arXiv:2006.00632* (2020).

[67] Yong Dai et al. “Adversarial training based multi-source unsupervised domain adaptation for sentiment analysis”. In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 34. 05. 2020, pp. 7618–7625.

[68] Yuecong Xu et al. “Video unsupervised domain adaptation with deep learning: A comprehensive survey”. In: *arXiv preprint arXiv:2211.10412* (2022).

[69] Yeganeh Madadi et al. “Deep visual unsupervised domain adaptation for classification tasks: a survey”. In: *IET Image Processing* 14.14 (2020), pp. 3283–3299.

[70] Rui Wang et al. “Cross-domain contrastive learning for unsupervised domain adaptation”. In: *IEEE Transactions on Multimedia* (2022).

[71] Yunzhong Hou and Liang Zheng. “Source free domain adaptation with image translation”. In: *arXiv preprint arXiv:2008.07514* (2020).

[72] Yuhe Ding et al. “Proxymix: Proxy-based mixup training with label refinery for source-free domain adaptation”. In: *Neural Networks* 167 (2023), pp. 92–103.

[73] Xiaobin Liu and Shiliang Zhang. “Graph consistency based mean-teaching for unsupervised domain adaptive person re-identification”. In: *arXiv preprint arXiv:2105.04776* (2021).

[74] Peshal Agarwal et al. “Unsupervised robust domain adaptation without source data”. In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 2022, pp. 2009–2018.

[75] Yuqi Fang et al. “Source-free unsupervised domain adaptation: A survey”. In: *arXiv preprint arXiv:2301.00265* (2022).

[76] Simon Kornblith, Jonathon Shlens, and Quoc V Le. “Do better imagenet models transfer better?” In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 2661–2671.

[77] Youshan Zhang and Brian D Davison. “Impact of imagenet model selection on domain adaptation”. In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision Workshops*. 2020, pp. 173–182.

[78] Jindong Wang et al. “Generalizing to Unseen Domains: A Survey on Domain Generalization”. In: *International Joint Conference on Artificial Intelligence*. Aug. 2021, pp. 4627–4635.

[79] Xinyu Guan et al. “Polycentric clustering and structural regularization for source-free unsupervised domain adaptation”. In: *arXiv preprint arXiv:2210.07463* (2022).

[80] Fan Wang et al. “Exploring domain-invariant parameters for source free domain adaptation”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 7151–7160.

[81] Yujie Liu et al. “A source free domain adaptation model based on adversarial learning for image classification”. In: *Applied Intelligence* 53.9 (2023), pp. 11389–11402.

[82] Guanglei Yang et al. “Transformer-based source-free domain adaptation”. In: *arXiv preprint arXiv:2105.14138* (2021).

[83] Jiahua Dong et al. “Confident anchor-induced multi-source free domain adaptation”. In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 2848–2860.

[84] Kurt Hornik et al. “Spherical k-means clustering”. In: *Journal of statistical software* 50 (2012), pp. 1–22.

[85] Hongyi Zhang et al. “mixup: Beyond Empirical Risk Minimization”. In: *International Conference on Learning Representations*. 2018.

[86] Hongyu Guo, Yongyi Mao, and Richong Zhang. "Mixup as locally linear out-of-manifold regularization". In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 33. 01. 2019, pp. 3714–3722.

[87] Luigi Carratino et al. "On mixup regularization". In: *The Journal of Machine Learning Research* 23.1 (2022), pp. 14632–14662.

[88] Ross Wightman. *PyTorch Image Models*. https://github.com/rwightman/pytorch-image-models. 2019. DOI: 10.5281/zenodo.4414861.

[89] Kate Saenko et al. "Adapting visual category models to new domains". In: *Computer Vision–ECCV 2010: 11th European Conference on Computer Vision, Heraklion, Crete, Greece, September 5-11, 2010, Proceedings, Part IV 11*. Springer. 2010, pp. 213–226.

[90] Hemanth Venkateswara et al. "Deep hashing network for unsupervised domain adaptation". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 5018–5027.

[91] Xingchao Peng et al. "Moment matching for multi-source domain adaptation". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 1406–1415.

[92] Mingsheng Long et al. "Deep transfer learning with joint adaptation networks". In: *International conference on machine learning*. PMLR. 2017, pp. 2208–2217.

[93] Tobias Ringwald and Rainer Stiefelhagen. "Adaptiope: A modern benchmark for unsupervised domain adaptation". In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 2021, pp. 101–110.

[94] Karen Simonyan and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition". In: *International Conference on Learning Representations*. 2015.

[95] Kaiming He et al. "Deep residual learning for image recognition". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2016, pp. 770–778.

[96] Mingxing Tan and Quoc Le. "Efficientnet: Rethinking model scaling for convolutional neural networks". In: *International Conference on Machine Learning*. 2019, pp. 6105–6114.

[97] Zhuang Liu et al. "A ConvNet for the 2020s". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2022, pp. 11976–11986.

[98] Ze Liu et al. "Swin transformer: Hierarchical vision transformer using shifted windows". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 10012–10022.

[99] Hugo Touvron et al. "Training data-efficient image transformers & distillation through attention". In: *International Conference on Machine Learning*. 2021, pp. 10347–10357.

[100] Alaaeldin Ali et al. "Xcit: Cross-covariance image transformers". In: *Advances in Neural Information Processing Systems*. 2021.

[101] TorchVision maintainers and contributors. *TorchVision: PyTorch's Computer Vision library*. https://github.com/pytorch/vision. 2016.

[102] Wouter M Kouw and Marco Loog. "A review of domain adaptation without target labels". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 43.3 (2019), pp. 766–785.

[103] Kuniaki Saito et al. "Tune it the right way: Unsupervised validation of domain adaptation via soft neighborhood density". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 9184–9193.

[104] Mathilde Caron et al. "Emerging properties in self-supervised vision transformers". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 9650–9660.

[105] Yanghao Li et al. "Revisiting batch normalization for practical domain adaptation". In: *International Conference on Learning Representations*. 2017.

[106] Andy Brock et al. "High-performance large-scale image recognition without normalization". In: *International Conference on Machine Learning*. PMLR. 2021, pp. 1059–1071.

[107] Junyi Chai et al. "Deep learning in computer vision: A critical review of emerging techniques and application scenarios". In: *Machine Learning with Applications* 6 (2021), p. 100134.

[108] Dengyong Zhou et al. "Learning with local and global consistency". In: *Advances in neural information processing systems* 16 (2003).

[109] Jiayuan Huang et al. "Correcting sample selection bias by unlabeled data". In: *Advances in neural information processing systems* 19 (2006).

[110] Sinno Jialin Pan et al. "Domain adaptation via transfer component analysis". In: *IEEE transactions on neural networks* 22.2 (2010), pp. 199–210.

[111] Chen Yang et al. "Source free domain adaptation for medical image segmentation with fourier style mining". In: *Medical Image Analysis* 79 (2022), p. 102457.

[112] Qing Tian et al. "Source-free unsupervised domain adaptation with sample transport learning". In: *Journal of Computer Science and Technology* 36.3 (2021), pp. 606–616.

[113] Antti Tarvainen and Harri Valpola. "Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results". In: *Advances in neural information processing systems* 30 (2017).

[114] Xinyu Liu and Yixuan Yuan. "A source-free domain adaptive polyp detection framework with style diversification flow". In: *IEEE Transactions on Medical Imaging* 41.7 (2022), pp. 1897–1908.

[115] Xuejun Zhao et al. "Adaptive contrastive learning with label consistency for source data free unsupervised domain adaptation". In: *Sensors* 22.11 (2022), p. 4238.

[116] Bo Han et al. "Co-teaching: Robust training of deep neural networks with extremely noisy labels". In: *Advances in neural information processing systems* 31 (2018).

[117] Youngdong Kim et al. "Nlnl: Negative learning for noisy labels". In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2019, pp. 101–110.

[118] Erik Englesson and Hossein Azizpour. “Generalized jensen-shannon divergence loss for learning with noisy labels”. In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 30284–30297.

[119] Haojian Zhang et al. “Unsupervised domain adaptation of black-box source models”. In: *arXiv preprint arXiv:2101.02839* (2021).

[120] Xin Luo et al. “Exploiting negative learning for implicit pseudo label rectification in source-free domain adaptive semantic segmentation”. In: *arXiv preprint arXiv:2106.12123* (2021).

[121] Jianfei Yang et al. “Divide to adapt: Mitigating confirmation bias for domain adaptation of black-box predictors”. In: *arXiv preprint arXiv:2205.14467* (2022).

[122] Mattia Litrico, Alessio Del Bue, and Pietro Morerio. “Guiding Pseudo-Labels With Uncertainty Estimation for Source-Free Unsupervised Domain Adaptation”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 7640–7650.

[123] Andrea Maracani et al. “Key Design Choices for Double-Transfer in Source-Free Unsupervised Domain Adaptation”. In: *arXiv preprint arXiv:2302.05379* (2023).

[124] Kaiming He et al. “Deep residual learning for image recognition”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[125] Adam Paszke et al. “Pytorch: An imperative style, high-performance deep learning library”. In: *Advances in neural information processing systems* 32 (2019).

[126] Shuhao Cui et al. “Fast batch nuclear-norm maximization and minimization for robust domain adaptation”. In: *arXiv preprint arXiv:2107.06154* (2021).

[127] Shiqi Yang et al. “Generalized source-free domain adaptation”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 8978–8987.

[128] Li Yi et al. “When source-free domain adaptation meets learning with noisy labels”. In: *arXiv preprint arXiv:2301.13381* (2023).

[129] Michael J Behrenfeld et al. “Biospheric primary production during an ENSO transition”. In: *Science* 291.5513 (2001), pp. 2594–2597.

[130] Daniel Boyce, Marlon Lewis, and Boris Worm. “Global phytoplankton decline over the past century”. In: *Nature* 466 (July 2010), pp. 591–6. DOI: 10.1038/nature09268.

[131] Vito P Pastore et al. “Establishing the baseline for using plankton as biosensor”. In: *Imaging, Manipulation, and Analysis of Biomolecules, Cells, and Tissues XVII*. Vol. 10881. International Society for Optics and Photonics. 2019, 108810H.

[132] Vito Paolo Pastore, Nimrod Megiddo, and Simone Bianco. “An Anomaly Detection Approach for Plankton Species Discovery”. In: *Image Analysis and Processing – ICIAP 2022*. Ed. by Stan Sclaroff et al. Cham: Springer International Publishing, 2022, pp. 599–609.

[133] Paolo Didier Alfano et al. “Efficient Unsupervised Learning for Plankton Images”. In: *2022 26th International Conference on Pattern Recognition (ICPR)*. 2022, pp. 1314–1321. DOI: 10.1109/ICPR56361.2022.9956360.

[134] Robert J. Olson and Heidi M. Sosik. “A submersible imaging-in-flow instrument to analyze nano-and microplankton: Imaging FlowCytobot”. In: *Limnology and Oceanography: Methods* 5.6 (2007), pp. 195–203. DOI: https://doi.org/10.4319/lom.2007.5.195. eprint: https://aslopubs.onlinelibrary.wiley.com/doi/pdf/10.4319/lom.2007.5.195. URL: https://aslopubs.onlinelibrary.wiley.com/doi/abs/10.4319/lom.2007.5.195.

[135] HM Sosik, EE Peacock, and EF Brownlee. *WHOI-Plankton: Annotated Plankton Images-Dataset for Developing and Evaluating Classification Methods (Woods Hole Open Access Server)*. 2021.

[136] Robert K. Cowen et al. *PlanktonSet 1.0: Plankton imagery data collected from F.G. Walton Smith in Straits of Florida from 2014-06-03 to 2014-06-06 and used in the 2015 National Data Science Bowl (NCEI Accession 0127422)*. 2015. DOI: 10.7289/V5D21VJD. URL: https://www.ncei.noaa.gov/archive/accession/0127422.

[137] Gaby Gorsky et al. “Digital zooplankton image analysis using the ZooScan integrated system”. In: *Journal of Plankton Research* 32.3 (Mar. 2010), pp. 285–303. ISSN: 0142-7873. DOI: 10.1093/plankt/fbp124. eprint: https://academic.oup.com/plankt/article-pdf/32/3/285/4394627/fbp124.pdf. URL: https://doi.org/10.1093/plankt/fbp124.

[138] Simon-Martin Schröder, Rainer Kiko, and Reinhard Koch. “Morphocluster: efficient annotation of Plankton images by clustering”. In: *Sensors* 20.11 (2020), p. 3060.

[139] Vito P. Pastore et al. “Annotation-free learning of plankton for classification and anomaly detection”. In: *Scientific Reports* 10.1 (July 2020), p. 12142. ISSN: 2045-2322. DOI: 10.1038/s41598-020-68662-3. URL: https://doi.org/10.1038/s41598-020-68662-3.

[140] Haiyong Zheng et al. “Automatic plankton image classification combining multiple view features via multiple kernel learning”. In: *BMC Bioinformatics* 18.16 (Dec. 2017), p. 570. ISSN: 1471-2105. DOI: 10.1186/s12859-017-1954-8. URL: https://doi.org/10.1186/s12859-017-1954-8.

[141] PF Culverhouse et al. “Automatic categorisation of five species of Cymatocylis (Protozoa, Tintinnida) by artificial neural network”. In: *Marine Ecology Progress Series* (1994), pp. 273–280.

[142] Qiao Hu and Cabell Davis. “Automatic plankton image recognition with co-occurrence matrices and support vector machine”. In: *Marine Ecology Progress Series* 295 (2005), pp. 21–31.

[143] Alessandra Lumini and Loris Nanni. “Deep learning and transfer learning features for plankton classification”. In: *Ecological Informatics* 51 (2019), pp. 33–43. ISSN: 1574-9541. DOI: https://doi.org/10.1016/j.ecoinf.2019.02.007. URL: https://www.sciencedirect.com/science/article/pii/S1574954118303054.

[144] Pablo González et al. “Automatic plankton quantification using deep features”. In: *Journal of Plankton Research* 41.4 (July 2019), pp. 449–463. ISSN: 0142-7873. DOI: 10.1093/plankt/fbz023. eprint: https://academic.oup.com/plankt/article-pdf/41/4/449/30279440/fbz023.pdf. URL: https://doi.org/10.1093/plankt/fbz023.

[145] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. “Deep learning”. In: *nature* 521.7553 (2015), pp. 436–444.

[146] Simon-Martin Schröder et al. "Low-Shot Learning of Plankton Categories". In: *Pattern Recognition*. Ed. by Thomas Brox, Andrés Bruhn, and Mario Fritz. Cham: Springer International Publishing, 2019, pp. 391–404. ISBN: 978-3-030-12939-2.

[147] Jialun Dai et al. "ZooplanktoNet: Deep convolutional network for zooplankton classification". In: *OCEANS 2016 - Shanghai* (2016), pp. 1–6.

[148] Alessandra Lumini, Loris Nanni, and Gianluca Maguolo. "Deep learning for plankton and coral classification". In: *Applied Computing and Informatics* ahead-of-print.ahead-of-print (Jan. 2020). ISSN: 2210-8327. DOI: 10.1016/j.aci.2019.11.004. URL: https://doi.org/10.1016/j.aci.2019.11.004.

[149] Sreenath P Kyathanahally et al. "Deep Learning Classification of Lake Zooplankton". In: *Frontiers in Microbiology* 12 (Nov. 2021). DOI: 10.3389/fmicb.2021.746297.

[150] Heidi M. Sosik and Robert J. Olson. "Automated taxonomic classification of phytoplankton sampled with imaging-in-flow cytometry". In: *Limnology and Oceanography: Methods* 5.6 (2007), pp. 204–216. DOI: https://doi.org/10.4319/lom.2007.5.204. eprint: https://aslopubs.onlinelibrary.wiley.com/doi/pdf/10.4319/lom.2007.5.204. URL: https://aslopubs.onlinelibrary.wiley.com/doi/abs/10.4319/lom.2007.5.204.

[151] Olga Russakovsky et al. "ImageNet Large Scale Visual Recognition Challenge". In: *International Journal of Computer Vision (IJCV)* 115.3 (2015), pp. 211–252. DOI: 10.1007/s11263-015-0816-y.

[152] Jia Deng et al. "Imagenet: A large-scale hierarchical image database". In: *2009 IEEE conference on computer vision and pattern recognition*. Ieee. 2009, pp. 248–255.

[153] Alexey Dosovitskiy et al. *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*. 2020. DOI: 10.48550/ARXIV.2010.11929. URL: https://arxiv.org/abs/2010.11929.

[154] Ze Liu et al. "Swin Transformer: Hierarchical Vision Transformer Using Shifted Windows". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. Oct. 2021, pp. 10012–10022.

[155] Hangbo Bao, Li Dong, and Furu Wei. *BEiT: BERT Pre-Training of Image Transformers*. 2021. DOI: 10.48550/ARXIV.2106.08254. URL: https://arxiv.org/abs/2106.08254.

[156] Xiu Li and Zuoying Cui. "Deep residual networks for plankton classification". In: *OCEANS 2016 MTS/IEEE Monterey*. 2016, pp. 1–4. DOI: 10.1109/OCEANS.2016.7761223.

[157] Ouyang Py, Hu Hong, and Shi Zhongzhi. "Plankton classification with deep convolutional neural networks". In: *2016 IEEE Information Technology, Networking, Electronic and Automation Control Conference*. IEEE. 2016, pp. 132–136.

[158] Buyu Guo et al. "Automated plankton classification from holographic imagery with deep convolutional neural networks". In: *Limnology and Oceanography: Methods* 19.1 (2021), pp. 21–36.

[159] Hansang Lee, Minseok Park, and Junmo Kim. "Plankton classification on imbalanced large scale database via convolutional neural networks with transfer learning". In: *2016 IEEE international conference on image processing (ICIP)*. IEEE. 2016, pp. 3713–3717.

[160] Francisco Caio Maia Rodrigues et al. “Evaluation of Transfer Learning Scenarios in Plankton Image Classification”. In: *Proceedings of the 13th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 5: VISAPP,* INSTICC. SciTePress, 2018, pp. 359–366. ISBN: 978-989-758-290-5. DOI: 10.5220/0006626703590366.

[161] Eric C. Orenstein and Oscar Beijbom. “Transfer Learning and Deep Feature Extraction for Planktonic Image Data Sets”. In: *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*. 2017, pp. 1082–1088. DOI: 10.1109/WACV.2017.125.

[162] Joseph L Walker and Eric C Orenstein. “Improving rare-class recognition of marine plankton with hard negative mining”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 3672–3682.

[163] Yuming Kuang. *Deep neural network for deep sea plankton classification*. Tech. rep. Technical Report 2015. Available online: https://pdfs. semanticscholar.org, 2015.

[164] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. “ImageNet Classification with Deep Convolutional Neural Networks”. In: *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*. NIPS’12. Lake Tahoe, Nevada: Curran Associates Inc., 2012, pp. 1097–1105.

[165] Gao Huang et al. “Densely Connected Convolutional Networks”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. July 2017.

[166] Mingxing Tan and Quoc Le. “EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks”. In: *Proceedings of the 36th International Conference on Machine Learning*. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. PMLR, Sept. 2019, pp. 6105–6114. URL: https://proceedings.mlr.press/v97/tan19a.html.

[167] Sang-Soo Baek et al. “Identification and enumeration of cyanobacteria species using a deep neural network”. In: *Ecological Indicators* 115 (2020), p. 106395.

[168] Jiawei Zhang et al. “SEM-RCNN: a squeeze-and-excitation-based mask region convolutional neural network for multi-class environmental microorganism detection”. In: *Applied Sciences* 12.19 (2022), p. 9902.

[169] Qiong Li et al. “Developing a microscopic image dataset in support of intelligent phytoplankton detection using deep learning”. In: *ICES Journal of Marine Science* 77.4 (2020), pp. 1427–1439.

[170] David Rivas-Villar et al. “Fully automatic detection and classification of phytoplankton specimens in digital microscopy images”. In: *Computer Methods and Programs in Biomedicine* 200 (2021), p. 105923.

[171] Amanda Elineau et al. *ZooScanNet: plankton images captured with the ZooScan*. 2018. DOI: 10.17882/55741.

[172] Kaiming He et al. “Deep Residual Learning for Image Recognition”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2016.

[173] Tal Ridnik et al. "ImageNet-21K Pretraining for the Masses". In: *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*. Ed. by J. Vanschoren and S. Yeung. Vol. 1. 2021. URL: https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/file/98f13708210194c475687be6106a3b84-Paper-round1.pdf.

[174] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. "Layer normalization". In: *arXiv preprint arXiv:1607.06450* (2016).

[175] Sergey Ioffe and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift". In: *International conference on machine learning*. pmlr. 2015, pp. 448–456.

[176] Tim Salimans and Diederik P. Kingma. "Weight Normalization: A Simple Reparameterization to Accelerate Training of Deep Neural Networks". In: *Proceedings of the 30th International Conference on Neural Information Processing Systems*. NIPS'16. Barcelona, Spain: Curran Associates Inc., 2016, pp. 901–909. ISBN: 9781510838819.

[177] Sebastian Ruder. "An overview of gradient descent optimization algorithms". In: *arXiv preprint arXiv:1609.04747* (2016).

[178] Mike Lambeta et al. "DIGIT: A Novel Design for a Low-Cost Compact High-Resolution Tactile Sensor With Application to In-Hand Manipulation". In: *IEEE Robotics and Automation Letters* 5.3 (July 2020), pp. 3838–3845. DOI: 10.1109/lra.2020.2977257.

[179] Wenzhen Yuan, Siyuan Dong, and Edward H Adelson. "Gelsight: High-resolution robot tactile sensors for estimating geometry and force". In: *Sensors* 17.12 (2017), p. 2762.

[180] Shaoxiong Wang et al. "Tacto: A fast, flexible, and open-source simulator for high-resolution vision-based tactile sensors". In: *IEEE Robotics and Automation Letters* 7.2 (2022), pp. 3930–3937.

[181] Zilin Si and Wenzhen Yuan. "Taxim: An example-based simulation model for gelsight tactile sensors". In: *IEEE Robotics and Automation Letters* 7.2 (2022), pp. 2361–2368.

[182] Berk Calli et al. "The ycb object and model set: Towards common benchmarks for manipulation research". In: *2015 international conference on advanced robotics (ICAR)*. IEEE. 2015, pp. 510–517.

[183] Carolina Higuera, Byron Boots, and Mustafa Mukadam. *Learning to Read Braille: Bridging the Tactile Reality Gap with Diffusion Models*. 2023. arXiv: 2304.01182 [cs.RO].

[184] Gabriele M. Caddeo et al. "Collision-aware In-hand 6D Object Pose Estimation using Multiple Vision-based Tactile Sensors". In: *2023 IEEE International Conference on Robotics and Automation (ICRA)*. 2023, pp. 719–725. DOI: 10.1109/ICRA48891.2023.10160359.

[185] Daniel Fernandes Gomes, Paolo Paoletti, and Shan Luo. "Generation of gelsight tactile images for sim2real learning". In: *IEEE Robotics and Automation Letters* 6.2 (2021), pp. 4177–4184.

[186] Francois R Hogan et al. “Seeing through your skin: Recognizing objects with a novel visuotactile sensor”. In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 2021, pp. 1218–1227.

[187] Weihang Chen et al. “Bidirectional sim-to-real transfer for gelsight tactile sensors with cyclegan”. In: *IEEE Robotics and Automation Letters* 7.3 (2022), pp. 6187–6194.

[188] Jun-Yan Zhu et al. “Unpaired image-to-image translation using cycle-consistent adversarial networks”. In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2223–2232.

[189] Sudharshan Suresh et al. “MidasTouch: Monte-Carlo inference over distributions across sliding touch”. In: *Conference on Robot Learning*. PMLR. 2023, pp. 319–331.

[190] Wenqiang Xu et al. “Visual-Tactile Sensing for In-Hand Object Reconstruction”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 8803–8812.

[191] Yiting Chen et al. “Sliding Touch-based Exploration for Modeling Unknown Object Shape with Multi-fingered Hands”. In: *arXiv preprint arXiv:2308.00576* (2023).

[192] Paloma Sodhi et al. “Patchgraph: In-hand tactile tracking with learned surface normals”. In: *2022 International Conference on Robotics and Automation (ICRA)*. IEEE. 2022, pp. 2164–2170.

[193] K. B. White, D. Cline, and P. K. Egbert. “Poisson Disk Point Sets by Hierarchical Dart Throwing”. In: *2007 IEEE Symposium on Interactive Ray Tracing*. Sept. 2007, pp. 129–132. DOI: 10.1109/RT.2007.4342600.

[194] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. “U-net: Convolutional networks for biomedical image segmentation”. In: *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*. Springer. 2015, pp. 234–241.

[195] Jascha Sohl-Dickstein et al. “Deep unsupervised learning using nonequilibrium thermodynamics”. In: *International conference on machine learning*. PMLR. 2015, pp. 2256–2265.

[196] Chenlin Meng et al. “SDEdit: Guided Image Synthesis and Editing with Stochastic Differential Equations”. In: *International Conference on Learning Representations*. 2021.

[197] Dan Hendrycks and Kevin Gimpel. “Gaussian error linear units (gelus)”. In: *arXiv preprint arXiv:1606.08415* (2016).

[198] Yu Xiang et al. “PoseCNN: A Convolutional Neural Network for 6D Object Pose Estimation in Cluttered Scenes”. In: *Proceedings of Robotics: Science and Systems*. Pittsburgh, Pennsylvania, June 2018. DOI: 10.15607/RSS.2018.XIV.019.

[199] Kaiming He et al. “Mask R-CNN”. In: *2017 IEEE International Conference on Computer Vision (ICCV)* (2017), pp. 2980–2988.

[200] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. “Yolov4: Optimal speed and accuracy of object detection”. In: *arXiv preprint arXiv:2004.10934* (2020).

[201] Giulia Pasquale et al. “Are we done with object recognition? The iCub robot’s perspective”. In: *Robotics and Autonomous Systems* 112 (2019), pp. 260–281. ISSN: 0921-8890. DOI: https://doi.org/10.1016/j.robot.2018.11.001. URL: http://www.sciencedirect.com/science/article/pii/S0921889018300332.

[202] E. Maiettini et al. “Interactive data collection for deep learning object detectors on humanoid robots”. In: *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*. Nov. 2017, pp. 862–868. DOI: 10.1109/HUMANOIDS.2017.8246973.

[203] E. Maiettini et al. “A Weakly Supervised Strategy for Learning Object Detection on a Humanoid Robot”. In: *2019 IEEE-RAS 19th International Conference on Humanoid Robotics (Humanoids)*. Nov. 2019.

[204] Zhi-Hua Zhou. “A brief introduction to weakly supervised learning”. en. In: *National Science Review* 5.1 (Jan. 2018). Publisher: Oxford Academic, pp. 44–53. ISSN: 2095-5138. DOI: 10.1093/nsr/nwx106. URL: https://academic.oup.com/nsr/article/5/1/44/4093912 (visited on 10/25/2020).

[205] S. Jamieson, J. P. How, and Y. Girdhar. “Active Reward Learning for Co-Robotic Vision Based Exploration in Bandwidth Limited Environments”. In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. 2020, pp. 1806–1812. DOI: 10.1109/ICRA40945.2020.9196922.

[206] Burr Settles. *Active Learning*. en. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool, 2012.

[207] Jerónimo Hernández-González, Iñaki Inza, and Jose A. Lozano. “Weak supervision and other non-standard classification problems: A taxonomy”. en. In: *Pattern Recognition Letters* 69 (Jan. 2016), pp. 49–55. ISSN: 0167-8655. DOI: 10.1016/j.patrec.2015.10.008. URL: http://www.sciencedirect.com/science/article/pii/S0167865515003505 (visited on 10/25/2020).

[208] C.P. Papageorgiou, M. Oren, and T. Poggio. “A general framework for object detection”. In: *Sixth International Conference on Computer Vision (IEEE Cat. No.98CH36271)*. Jan. 1998, pp. 555–562. DOI: 10.1109/ICCV.1998.710772.

[209] Paul Viola and Michael Jones. *Rapid object detection using a boosted cascade of simple features*. 2001.

[210] Shaoqing Ren et al. “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”. In: *Advances in Neural Information Processing Systems 28*. Ed. by C. Cortes et al. Curran Associates, Inc., 2015, pp. 91–99. (Visited on 10/20/2020).

[211] Jifeng Dai et al. “R-FCN: Object Detection via Region-based Fully Convolutional Networks”. In: *Advances in Neural Information Processing Systems 29*. Ed. by D. D. Lee et al. Curran Associates, Inc., 2016, pp. 379–387. (Visited on 10/26/2020).

[212] Jifeng Dai et al. “Deformable convolutional networks”. In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 764–773.

[213] Jiangmiao Pang et al. “Libra R-CNN: Towards Balanced Learning for Object Detection”. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. ISSN: 2575-7075. June 2019, pp. 821–830. DOI: 10.1109/CVPR.2019.00091.

[214] Wei Liu et al. “SSD: Single Shot MultiBox Detector”. en. In: *Computer Vision – ECCV 2016*. Ed. by Bastian Leibe et al. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2016, pp. 21–37. ISBN: 978-3-319-46448-0. DOI: 10.1007/978-3-319-46448-0_2.

[215] Sheping Zhai et al. “DF-SSD: An Improved SSD Object Detection Algorithm Based on DenseNet and Feature Fusion”. In: *IEEE Access* 8 (2020), pp. 24344–24357.

[216] T. Lin et al. “Focal Loss for Dense Object Detection”. In: *2017 IEEE International Conference on Computer Vision (ICCV)*. ISSN: 2380-7504. Oct. 2017, pp. 2999–3007. DOI: 10.1109/ICCV.2017.324.

[217] Ian J Goodfellow et al. “An empirical investigation of catastrophic forgetting in gradient-based neural networks”. In: *arXiv preprint arXiv:1312.6211* (2013).

[218] E. Maiettini et al. “Speeding-up Object Detection Training for Robotics with FALKON”. In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Oct. 2018.

[219] Elisa Maiettini et al. “On-line object detection: a robotics challenge”. In: *Autonomous Robots* (Nov. 2019). ISSN: 1573-7527. DOI: 10.1007/s10514-019-09894-9. URL: https://doi.org/10.1007/s10514-019-09894-9.

[220] Alessandro Rudi, Luigi Carratino, and Lorenzo Rosasco. “Falkon: An optimal large scale kernel method”. In: *Advances in Neural Information Processing Systems*. 2017, pp. 3888–3898.

[221] Giacomo Meanti et al. “Kernel methods through the roof: handling billions of points efficiently”. In: _eprint: 2006.10350. 2020.

[222] Alessandro Rudi, Raffaello Camoriano, and Lorenzo Rosasco. “Less is more: Nyström computational regularization”. In: *Advances in Neural Information Processing Systems*. 2015, pp. 1657–1665.

[223] Andreas Kirsch, Joost van Amersfoort, and Yarin Gal. “BatchBALD: Efficient and Diverse Batch Acquisition for Deep Bayesian Active Learning”. In: *arXiv:1906.08158 [cs, stat]* (Oct. 2019). arXiv: 1906.08158. URL: http://arxiv.org/abs/1906.08158 (visited on 02/25/2020).

[224] Fedor Zhdanov. “Diverse mini-batch Active Learning”. In: *arXiv:1901.05954 [cs, stat]* (Jan. 2019). arXiv: 1901.05954. URL: http://arxiv.org/abs/1901.05954 (visited on 05/25/2020).

[225] Jordan T. Ash et al. “Deep Batch Active Learning by Diverse, Uncertain Gradient Lower Bounds”. en. In: Jan. 2020. URL: https://openreview.net/forum?id=0HjEAtQNNWD (visited on 10/26/2020).

[226] Hamed H. Aghdam et al. “Active Learning for Deep Detection Neural Networks”. en. In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. Seoul, Korea (South): IEEE, Oct. 2019, pp. 3671–3679. ISBN: 978-1-72814-803-8. DOI: 10.1109/ICCV.2019.00377. URL: https://ieeexplore.ieee.org/document/9009535/ (visited on 06/16/2020).

[227] Elmar Haussmann et al. “Scalable Active Learning for Object Detection”. In: *arXiv:2004.04699 [cs]* (Apr. 2020). arXiv: 2004.04699. URL: http://arxiv.org/abs/2004.04699 (visited on 06/16/2020).

[228] Sai Vikas Desai et al. “An Adaptive Supervision Framework for Active Learning in Object Detection”. In: 2019.

[229] Chieh-Chi Kao et al. “Localization-Aware Active Learning for Object Detection”. en. In: *Computer Vision – ACCV 2018*. Ed. by C.V. Jawahar et al. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2019, pp. 506–522. ISBN: 978-3-030-20876-9. DOI: 10.1007/978-3-030-20876-9_32.

[230] Yandong Li et al. “Improving Object Detection with Selective Self-supervised Self-training”. In: *European Conference on Computer Vision*. Springer, 2020, pp. 589–607.

[231] Keze Wang et al. “Towards Human-Machine Cooperation: Self-supervised Sample Mining for Object Detection”. In: *arXiv:1803.09867 [cs]* (Mar. 2018). arXiv: 1803.09867. URL: http://arxiv.org/abs/1803.09867 (visited on 10/01/2019).

[232] Keze Wang et al. “Cost-effective object detection: Active sample mining with switchable selection criteria”. In: *IEEE transactions on neural networks and learning systems* 30.3 (2019). Publisher: IEEE, pp. 834–850.

[233] Poojan Oza et al. “Unsupervised domain adaptation of object detectors: A survey”. In: *arXiv preprint arXiv:2105.13502* (2021).

[234] Xianfeng Li et al. “A Free Lunch for Unsupervised Domain Adaptive Object Detection without Source Data”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 35. 10. 2021, pp. 8474–8481.

[235] Aruni RoyChowdhury et al. “Automatic Adaptation of Object Detectors to New Domains Using Self-Training”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2019.

[236] Federico Ceola et al. “Fast Region Proposal Learning for Object Detection for Robotics”. In: *arXiv preprint arXiv:2011.12790* (2021). arXiv: 2011.12790 [cs.CV].

[237] Federico Ceola et al. “Fast Object Segmentation Learning with Kernel-based Methods for Robotics”. In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. 2021.

[238] Shaoqing Ren et al. “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”. In: *Neural Information Processing Systems (NIPS)*. 2015.

[239] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning: data mining, inference, and prediction*. Springer Science & Business Media, 2009.

[240] Kemal Oksuz et al. “Imbalance problems in object detection: A review”. In: *IEEE transactions on pattern analysis and machine intelligence* (2020).

[241] Hamed H. Aghdam et al. “Active Learning for Deep Detection Neural Networks”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. Oct. 2019.

[242] Tsung-Yi Lin et al. “Microsoft COCO: Common Objects in Context”. In: *European Conference on Computer Vision (ECCV)*. Oral. Zürich, Jan. 1, 2014. URL: /se3/wp-content/uploads/2014/09/coco_eccv.pdf,%20http://mscoco.org.

[243] Kaiming He et al. “Deep Residual Learning for Image Recognition”. In: *arXiv preprint arXiv:1512.03385* (2015).

[244] Shreyas Hampali et al. “HOnnotate: A method for 3D annotation of hand and object poses”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 3196–3206.

[245] Yu Xiang et al. “PoseCNN: A Convolutional Neural Network for 6D Object Pose Estimation in Cluttered Scenes”. In: 2018.

[246] Berk Calli et al. “The YCB object and model set: Towards common benchmarks for manipulation research”. In: *2015 international conference on advanced robotics (ICAR)*. IEEE. 2015, pp. 510–517.

[247] Mark Everingham et al. “The Pascal Visual Object Classes (VOC) Challenge”. en. In: *International Journal of Computer Vision* 88.2 (June 2010), pp. 303–338. ISSN: 1573-1405. DOI: 10.1007/s11263-009-0275-4. URL: https://doi.org/10.1007/s11263-009-0275-4 (visited on 10/29/2020).

[248] Shengyu Zhang et al. “Instruction tuning for large language models: A survey”. In: *arXiv preprint arXiv:2308.10792* (2023).

[249] Rafael Müller, Simon Kornblith, and Geoffrey E Hinton. “When does label smoothing help?” In: *Advances in neural information processing systems* 32 (2019).

[250] F. Pedregosa et al. “Scikit-learn: Machine Learning in Python”. In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

Appendix A

Trust And Balance (Chapter 4)

A.1 Training Details

As outlined in Sec. 4.6.1, we employed standard procedures and hyperparameters drawn from existing literature.

Pre-training. For the ResNet50 architecture, we utilized weights pre-trained on ImageNet as provided by TorchVision [101]. For the ViT-Large architecture, weights pre-trained on ImageNet21k from the timm library [88] were used.

Input pipeline and augmentations. Initially, images are resized to dimensions 256×256. For training purposes, a random crop yielding a dimension of 224×224 is executed along with the application of a random horizontal flip. During evaluation, a centered crop is applied. Subsequent to these transformations, images undergo normalization based on the mean and standard deviation values from ImageNet, consistent with the pre-training setup of the backbones.

Source fine-tuning. The fine-tuning on the source domain employs SGD with Nesterov Momentum of 0.9 and an $L2$ penalty of 10^{-3}. A batch size of 64 is used, with learning rates initialized to 10^{-3} for the pre-trained backbone and 10^{-2} for the additional bottleneck and classifier with freshly initialized weights. We employ a Cross-entropy objective with label smoothing [249] using a factor of 0.1 and clip gradients at 5.0. The learning rate adjustment follows the equation:

$$lr(t) = lr(0) * \left(1 + 10 \cdot \frac{t}{T}\right)^{-0.75} \tag{A.1}$$

where $lr(0)$ represents the initial learning rate, T the total training steps and t is the current training step. The dataset is partitioned into training (85%) and validation (15%)

subsets. Training continues for 100 epochs, and after each epoch, validation accuracy is assessed. Model weights achieving the highest validation accuracy are retained. Distributed fine-tuning is executed on 4 Nvidia V100 16GB GPUs.
Target adaptation. This phase uses a batch size of 64 with similar learning rates and scheduling as source fine-tuning for 15 epochs, but the classifier γ remains fixed (lr set to 0). At every epoch, FTSP computes pseudo-labels, and by default, Multinomial Logistic Regression serves as the classifier. We employ the MLR classifier from Scikit-Learn [250] with default settings and minimal strength $L2$ regularization ($C = \frac{1}{\lambda} = 1000$). We subsequently apply our Pseudo-label Refinement (PR): we remove the 20% least confident pseudo-labels from each class based on predicted probabilities via the **label deletion** step. In the **label completion** phase, Label Spreading [108] (from Scikit-Learn) with default hyperparameters (RBF kernel with $gamma = 20$) is used. The TSAL objective, described in the main text, is then minimized. Additionally, MixUp [51] regularization is utilized, where the mixing ratio is sourced from a Beta distribution with parameters $\alpha = \beta = 0.3$. Since no label is available in this training phase the adapted model is the one obtained after the whole 15 epochs training that is directly tested with labels. Target adaptation takes place on a singular GPU: NVIDIA V100 16GB for ResNet50 and NVIDIA A100 80GB for ViT-Large.

A.2 Few Trusted Samples Pseudo-labeling: an ablation study

In this section, we present an ablation study centered around the FTSP methodology. It is crucial to clarify that the hyperparameters used for the experiments presented in the main text were kept fixed. The ablation experiments described herein have been conducted subsequently to those in the main text to prevent potential evaluation biases. As we explore further, it becomes evident that the hyperparameters used in the main text could be sub-optimal for Office31 and Office-Home benchmarks. In particular the value of trusted samples (K) and also the classifier (MLR) used.
Number of Trusted Samples and Pseudo-label Refinement. Throughout the experiments, we consistently set the number K of trusted samples (per class) to 3 for ResNet50 and 7 for ViT. We selected $K = 3$ for ResNet50 aiming for a constrained set of trusted samples, ensuring these samples had accurate labels for the classifier training in FTSP. This approach stems from the understanding that a limited number of samples can adequately train a classifier, which then generalizes effectively across the entire target domain, as our empirical results

confirm. Given that (according to the ImageNet benchmark) ViT-Large is a better performing model compared to ResNet50, it also exhibits superior out-of-distribution generalization as shown in [123]. Consequently, we followed this intuition and we increased K to 7 for ViT-L, anticipating enhanced predictions and greater model reliability. In Table A.1, we provide an ablation study focusing on the value of K for ResNet50 and on the effects of the Pseudo-labeling Refinement phase that we propose. As it is possible to observe in the table both for Office31 and Office-Home the algorithm is robust to the choice of K in the range considered and the Pseudo-Labeling refinement provides in almost all cases benefits in terms of accuracy. Additionally the value of $K = 5$ seems to be the best choice for these datasets considering our classifier (MLR), increasing marginally the results reported in the main text (with $K = 3$). For completeness we report in Table A.2 and A.5 the comparison of the best performing SOTA methods, with our proposed method with optimal K.

Classifier. We present an ablation study about the choice of the classifier for FTSP. In particular, in this study, we focused on the FTSP methodology **without Pseudo-label Refinement** and we optimized the TSAL objective as stated in the main text. We evaluated the following classifiers: Support Vector Machine Classifier with RBF (SVC-R) and Linear (SVC-L) kernels, Multinomial Logistic Regression (MLR), and Linear Discriminant Analysis (LDA). For these classifiers, we examined a variety of L2 regularization strengths, regulated by the C hyperparameter in Scikit-Learn (where the value of C is inversely proportional to the strength of regularization). We also assessed different shrinkage values for LDA.

Results for the Office-Home dataset are provided in table A.3, and the outcomes for the Office31 dataset are presented in table A.4. For both datasets, it is evident that MLR, particularly with weaker regularization, outperforms SVC. Furthermore, the Linear Discriminant Analysis Classifier surpasses MLR in performance for the datasets examined.

Table A.1 **Ablation on K with and without Pseudo-label Refinement**: Average accuracy of FTSP+TSAL using MLR with C=1000 for Office31 and Office-Home using different values of K.

Classifier	K=2	K=3	K=5	K=7	K=10
Office31 (w/o PR)	89.0	89.6	**89.7**	89.2	88.9
Office-Home (w/o PR)	72.3	72.4	**72.9**	72.5	72.3
Office31 (with PR)	89.6	89.9	**90.3**	89.9	89.8
Office-Home (with PR)	72.6	72.8	**72.9**	72.8	72.5

Table A.2 Comparison of TAB (with $K = 3$ and $K = 5$) with two state-of-the-art methods on the **Office31** dataset using ResNet50. The top results are highlighted in **bold**, while the runners-up are <u>underlined</u>.

Method	A→D	A→W	D→A	D→W	W→A	W→D	**Avg**
DIPE$_{\text{R50}}$ [80]	**96.6**	93.1	75.5	98.4	<u>77.2</u>	99.6	<u>90.1</u>
A^{2}Net$_{\text{R50}}$ [59]	94.5	94.0	76.7	**99.2**	76.1	**100.0**	<u>90.1</u>
TAB$_{K=3}$	94.4	**94.7**	<u>76.9</u>	97.4	76.0	<u>99.8</u>	89.9
TAB$_{K=5}$	<u>94.6</u>	<u>94.1</u>	**77.5**	<u>98.7</u>	**77.3**	99.4	**90.3**

Table A.3 **Ablation on Office-Home**: Evaluation of SVM Classifiers using RBF (SVC-R) and Linear (SVC-L) kernels, and Multinomial Logistic Regression (MLR) across varying L2 regularization strengths. Here, the C hyperparameter is inversely proportional to regularization strength. Linear Discriminant Analysis with different shrinkage values (S) is also assessed. The value of trusted samples K is set to 3.

Classifier	C=0.1	C=1.0	C=1000	S=0.5	S=0.99
SVC-R	68.9	71.5	71.3	—	—
SVC-L	70.0	72.0	71.7	—	—
MLR	72.3	72.3	72.4	—	—
LDA	—	—	—	72.6	**72.7**

Table A.4 **Ablation on Office31**: Evaluation of SVM Classifiers using RBF (SVC-R) and Linear (SVC-L) kernels, and Multinomial Logistic Regression (MLR) across varying L2 regularization strengths. Here, the C hyperparameter is inversely proportional to regularization strength. Linear Discriminant Analysis with different shrinkage values (S) is also assessed.The value of trusted samples K is set to 3.

Classifier	C=0.1	C=1.0	C=1000	S=0.5	S=0.99
SVC-R	87.4	89.2	89.4	—	—
SVC-L	88.2	89.1	89.3	—	—
MLR	89.0	89.2	89.6	—	—
LDA	—	—	—	89.2	**89.8**

Table A.5 Comparison of TAB (with $K = 3$ and $K = 5$) with two state-of-the-art methods on the **Office-Home** dataset using ResNet50. The top results are highlighted in **bold**, while the runners-up are <u>underlined</u>.

Method	A→C	A→P	A→R	C→A	C→P	C→R	P→A	P→C	P→R	R→A	R→C	R→P	**Avg**
AAD$_{\text{R50}}$ [57]	**59.3**	79.3	<u>82.1</u>	**68.9**	<u>79.8</u>	<u>79.5</u>	67.2	<u>57.4</u>	<u>83.1</u>	72.1	58.5	**85.4**	72.7
A^{2}Net$_{\text{R50}}$ [59]	58.4	79.0	**82.4**	67.5	79.3	78.9	68.0	56.2	82.9	**74.1**	**60.5**	85.0	<u>72.8</u>
TAB$_{K=3}$	<u>58.9</u>	<u>79.6</u>	81.5	<u>68.6</u>	78.0	**79.8**	<u>69.3</u>	56.8	**83.7**	<u>73.2</u>	59.5	84.7	<u>72.8</u>
TAB$_{K=5}$	58.3	**80.3**	81.5	67.3	**81.0**	78.4	**69.8**	**57.8**	83.0	72.0	<u>60.1</u>	<u>85.3</u>	**72.9**

A.3 Efficiency of the Proposed Approach

The computational demands for executing the optimization of TSAL objective during the target adaptation stage are comparable to those encountered in standard supervised learning. These demands are influenced by several factors, including the chosen model architecture (backbone), the use of hardware accelerators, the software implementation, and the volume of data being processed.

The additional computational steps introduced by TAB at each adaptation epoch are primarily for generating pseudo-labels, which involves the following processes:

1. Extraction of features and prediction probabilities for images from the target domain.
2. Selection of a Few-Trusted Samples dataset based on model predictions.
3. Training of a classifier on the Few-Trusted Samples dataset.
4. Use of the trained classifier to generate pseudo-labels for the target domain.
5. Optional application of Label Spreading to further refine the pseudo-labels.

The majority of the computational effort and time is allocated to the first step, which is a common requirement across many SF-UDA algorithms. Steps 2 through 5, which are unique to the TAB approach, require comparatively minimal time relative to feature extraction. For context, in our experiments, steps 2 through 5, implemented using Scikit-learn algorithms running on CPU, took only a few seconds, whereas feature extraction could take minutes, especially with large architectures, extensive datasets, or in the absence of hardware acceleration. Our publicly available code includes functions that facilitate feature extraction (and model optimization) through distributed computing across multiple GPUs, significantly enhancing time-efficiency.

In summary, the computational times for our approach are on par with other methodologies documented in the literature, such as SHOT, AAD, and NRC when models are adapted using a single hardware accelerator. Moreover, our distributed computing implementation further optimizes the performance of TAB, especially when additional GPUs are utilized, enabling experiments with larger models and datasets.