Domain Adaptation for Urban Scene Segmentation

Antoine Saporta

▶ **To cite this version:**

Antoine Saporta. Domain Adaptation for Urban Scene Segmentation. Artificial Intelligence [cs.AI]. Sorbonne Université, 2022. English. NNT : 2022SORUS115 . tel-03886201

HAL Id: tel-03886201
https://theses.hal.science/tel-03886201

Submitted on 6 Dec 2022

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

THÈSE DE DOCTORAT DE SORBONNE UNIVERSITÉ
Spécialité **Informatique**
École Doctorale Informatique, Télécommunications et Électronique (Paris)

Domain Adaptation for Urban Scene Segmentation

Adaptation de Domaine pour la Segmentation de Scènes Urbaines

Présentée par
Antoine SAPORTA

Dirigée par
Matthieu CORD

Pour obtenir le grade de
DOCTEUR de SORBONNE UNIVERSITÉ

Présentée et soutenue publiquement le 14 avril 2022

Devant le jury composé de :

M. Alain Rakotomamonjy *Professeur, Université de Rouen – LITIS*	Rapporteur
Mme. Gabriela Csurka *Research Director, Naver Labs*	Rapportrice
M. Frédéric Precioso *Professeur, Université Côte d'Azur – I3S*	Examinateur
M. Patrick Gallinari *Professeur, Sorbonne Université – ISIR*	Examinateur
M. Patrick Pérez *Research Director, Valeo.ai*	Co-directeur de thèse
M. Matthieu Cord *Professeur, Sorbonne Université – ISIR & Senior Scientist, Valeo.ai*	Directeur de thèse
M. Tuan-Hung Vu *Research Scientist, Valeo.ai*	Examinateur

Antoine Saporta: *Domain Adaptation for Urban Scene Segmentation,* © 2022

ABSTRACT

This thesis tackles some of the scientific locks of perception systems based on neural networks for autonomous vehicles. This dissertation discusses domain adaptation, a class of tools aiming at minimizing the need for labeled data. Domain adaptation allows generalization to so-called target data that share structures with the labeled so-called source data allowing supervision but nevertheless following a different statistical distribution.

First, we study the introduction of privileged information in the source data, for instance, depth labels. The proposed strategy, BerMuDA, bases its domain adaptation on a multimodal representation obtained by bilinear fusion, modeling complex interactions between segmentation and depth.

Next, we examine self-supervised learning strategies in domain adaptation, relying on selecting predictions on the unlabeled target data, serving as pseudo-labels. We propose two new selection criteria: first, an entropic criterion with ESL; then, with ConDA, using an estimate of the true class probability.

Finally, the extension of adaptation scenarios to several target domains as well as in a continual learning framework is proposed. Two approaches are presented to extend traditional adversarial methods to multi-target domain adaptation: Multi-Dis. and MTKT. In a continual learning setting for which the target domains are discovered sequentially and without rehearsal, the proposed CTKT approach adapts MTKT to this new problem to tackle catastrophic forgetting.

RÉSUMÉ

Cette thèse attaque certains des verrous scientifiques des systèmes de perception à base de réseaux de neurones des véhicules autonomes. Une classe d'outils abordée dans cette thèse pour limiter les besoins de données étiquetées est celle de l'adaptation de domaine. Celle-ci permet la généralisation à des données dites cibles qui partagent des structures avec les données annotées dites sources permettant la supervision mais qui suivent néanmoins une distribution statistique différente.

D'abord, nous étudions l'introduction d'information privilégiée dans les données sources, par exemple des annotations de profondeur. La stratégie proposée BerMuDA appuie son adaptation de domaine sur une représentation multimodale obtenue par fusion bilinéaire, modélisant des interactions complexes entre segmentation et profondeur.

Ensuite, nous examinons les stratégies d'auto-apprentissage en adaptation de domaine, reposant sur la sélection de prédictions sur les données cibles non étiquetées, servant de pseudo-étiquettes. Nous proposons deux nouveaux critères de sélection : d'abord, un critère entropique avec ESL ; puis, avec ConDA, utilisant une estimation de la probabilité de la vraie classe.

Enfin, l'extension des scénarios d'adaptation à plusieurs domaines cibles ainsi que dans un cadre d'apprentissage continu est proposée. Deux approches sont présentées pour étendre les méthodes adversaires traditionnelles à l'adaptation de domaine multi-cible : Multi-Dis. et MTKT. Dans un cadre d'apprentissage continu, les domaines cibles sont découverts séquentiellement et sans répétition. L'approche proposée CTKT adapte MTKT à ce nouveau problème pour lutter contre l'oubli catastrophique.

REMERCIEMENTS

J'aimerai remercier les nombreuses personnes qui ont concouru à l'existence de ce manuscrit.

J'aimerais tout d'abord remercier Matthieu Cord qui m'a accueilli à MLIA et m'a encadré, aidé et guidé dans le domaine complexe et compétitif du deep learning pendant plus de 4 ans, du stage de master jusqu'à aujourd'hui. Je remercie également Patrick Pérez qui m'a accueilli à Valeo.ai et a été source de conseils et d'inspiration pendant ces 3 années de thèse, ainsi que Tuan-Hung Vu, chercheur à Valeo.ai, pour les discussions enrichissantes et les collaborations fructueuses pendant ma thèse.

Je remercie bien évidemment aussi l'ensemble des membres du jury pour l'intérêt qu'ils ont porté à mes travaux.

Je tiens ensuite à remercier l'ensemble de l'équipe de Matthieu Cord au MLIA, les "Chordettes", avec qui j'ai passé d'excellents moments : Thibaut Durand, Michael Blot, Taylor Mordan, Micael Carvalho, Thomas Robert, Hedi Ben-Younès, Rémi Cadène, Yifu Chen, Arnaud Dapogny, Corentin Dancette, Arthur Douillard, Rémy Sun, Alexandre Ramé, Asya Grechka, Guillaume Couairon, Mustafa Shukor. Je remercie également l'équipe de doctorants de Valeo.ai : Maximilian Jaritz, Simon Roburin, Arthur Ouaknine, Charles Corbière, Huy Van Vo, Laura Calem, Florent Bartoccioni, Léon Zheng, Antonín Vobecký. Il est évident que les discussions avec toutes ces personnes sont une part importante de ce que ce doctorat m'a apporté.

Pour les nombreux instants de convivalité et leur soutien pendant ces années au sein de l'équipe MLIA, je tiens aussi à remercier Ahmed Mazari, Clara Gainon de Forsan de Gabriac, Jean-Yves Franseschi, Yuan Yin.

Je remercie l'intégralité des doctorants et professeurs de l'équipe MLIA et tous les chercheurs de Valeo.ai, trop nombreux pour être tous cités, ainsi que les personnes qui ont pu m'apporter une aide précieuse sur les plans administratifs et techniques, en particulier Nadine Taniou et Christophe Bouder.

Pour finir, je remercie bien sûr ma famille et mes amis pour leur soutien durant ce marathon, et tout particulièrement Alice pour sa présence et son support précieux au quotidien.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACRONYMS

AdvEnt	Adversarial Entropy Minimization
AI	Artificial Intelligence
ASPP	Atrous Spatial Pyramidal Pooling
BCE	Binary Cross-Entropy
BDL	Bidirectional Learning
BerMuDA	Bilinear Multimodal Domain Adaptation
CE	Cross-Entropy
ConDA	Confidence Learning for Domain Adaptation
ConfidNet	Confidence Network
CNN	Convolutional Neural Network
CORAL	CORrelation ALignment
CTKT	Continual Target Knowledge Transfer
CV	Computer Vision
DA	Domain Adaptation
DADA	Depth-Aware Domain Adaptation
DL	Deep Learning
DISE	Domain Invariant Structure Extraction
DNN	Deep Neural Network
ESL	Entropy-based Self-supervised Learning
FCN	Fully-Convolutional Network
GAN	Generative Adversarial Network
GPU	Graphics Processing Unit
IoU	Intersection over Union
IT	Image Translation
JDOT	Joint distribution Optimal Transport
KL	Kullback-Leibler
MCP	Maximum Class Probability
MinEnt	Minimizing Entropy
MMD	Maximum Mean Discrepancy
Multi-Dis.	Multi-Discriminator
mIoU	mean Intersection over Union
ML	Machine Learning

MTKT	Multi-Target Knowledge Transfer
MTL	Multi-Task Learning
NN	Neural Network
OCDA	Open-Compound Domain Adaptation
PI	Privileged Information
ReLU	Rectified Linear Unit
SGD	Stochastic Gradient Descent
SPIGAN	Simulator Privileged Information and Generative Adversarial Network
SSL	Softmax-based Self-supervised Learning
ST	Self-Training
TCP	True Class Probability
TL	Transfer Learning
UDA	Unsupervised Domain Adaptation
VRU	Vulnerable Road User

CHAPTER 1

INTRODUCTION

1.1 Context

Autonomous vehicles have recently returned to the front of the stage with the dazzling progress of Artificial Intelligence (AI) and the emergence of many actors around new forms of mobility (connected, shared, autonomous and electric), tech giants, like Google-Waymo and Uber, and countless start-ups. As a major French automotive supplier and world leader in automotive sensors, Valeo is positioned at the heart of these upcoming revolutions, especially regarding autonomous vehicles. Resolutely involved in improving advanced driving assistance systems (ADAS) deployed in commercial cars, the group has been developing major research and development activity on autonomous driving for several years. Two experimental programs, Valeo Cruise4U for autonomous driving on motorways and Valeo Drive4U for driving in urban environments, were set up in 2012. The first gave rise to several large-scale experiments, including crossing Europe and the United States, or 24 hours on the Parisian ring road. During the Paris Motor Show 2018, the latter came to be the first-ever experience of autonomous driving in the center of Paris.

The recent progress in AI brought by Deep Learning (DL) (a branch of Machine Learning (ML) relying on Neural Networks (NNs) with multiple layers) largely explains the spectacular resurgence of driverless cars. Thanks to the new generation of Convolutional Neural Networks (CNNs), cameras may be embedded to understand, in real-time, crucial aspects of the environment: nature and position of vehicles, pedestrians and stationary objects; position and meaning of lane markings, signs, traffic lights; drivable area; etc. Nevertheless, the richness and versatility of visual signals have their drawbacks: accessing the underlying information is often indirect and complex. The projective nature of the camera makes it especially challenging when considering three-dimensional information. To increase the extent, the robustness, and the quality of automated perception, other sensors are thus necessary, in particular radar and LiDAR (3D laser scanner). Both of these sensors directly measure a sparse representation of the three-dimensionality and dynamic of their environment. Thus, autonomous vehicles

generally conjointly leverage cameras, radars, and LiDARs in their perception system (Figure 1.1). The fusion of their information is often late, after the decision function attached to each sensor, but may also be early, either on the raw signal or after a few transformations. Sensors and their fusions are the heart of the first step of perception, followed in succession by the localization, prediction, planning, and finally, vehicle control.

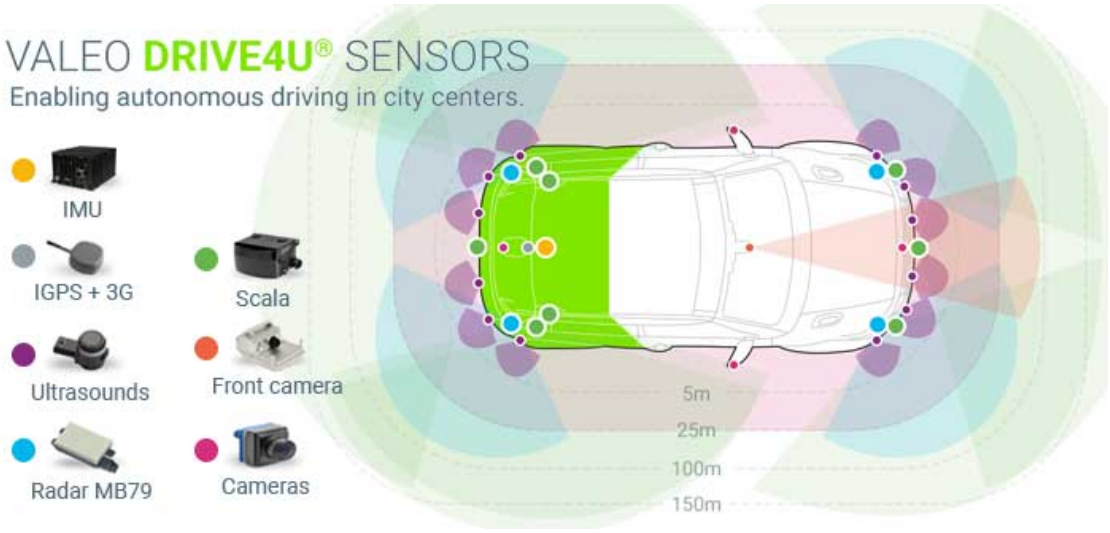


Figure 1.1 – **Sensors of the Drive4U Valeo car.** The Drive4U Valeo car leverages multiple types of sensors (mass-produced by Valeo Comfort and Driving Assistance) all around the vehicle in order to acquire complete knowledge of its environment in real-time. Illustration from Valeo.

1.2 Motivation

Despite these recent advances, there is still a long way to go before we see widespread deployment of automated taxis, let alone the launch of fully self-driving vehicles on the market to private individuals. One of the major obstacles is the capacity of autonomous systems to handle unforeseen situations in new environments. Although these systems are now extremely effective, they remain vulnerable and subject to critical errors even in situation which a human would handle. For example, the perception system may fail to recognize a new animal or vehicle it has not been trained to detect, catastrophically misleading the rest of the decision stack. Critical failures of autonomous cars could lead to disastrous accidents with other vehicles or Vulnerable Road Users (VRUs). These problems have to be solved before any advanced autonomous vehicle may be approved by regulators and adopted by users.

Many scientific obstacles remain for autonomous driving systems. This dissertation specifically focuses on the perception layers based on NN. Today, these are trained in a fully-supervised fashion, requiring massive amounts of annotated data. While powerful, this form of training raises major issues. Gathering annotated datasets that are large and varied enough for supervised learning is a complex and costly enterprise. For instance, the semantic segmentation of urban scenes is a task that aims at predicting a category label for each pixel of an image, illustrated in Figure 1.2. Thus, this task requires thousands of high-definition urban scene images and manually annotated semantic segmentation maps. While capturing large datasets of urban scene images is made easy by car-embedded cameras, the manual annotation of their semantic segmentation maps costs a lot of time and human energy: for instance, a single image of the Cityscapes dataset (Cordts et al. 2016b) required more than 1.5 hours on average in annotation and quality control.

Moreover, such datasets remain limited, considering the diversity, complexity, and unpredictability of environments a vehicle may encounter. For safety reasons, it is much more complex to acquire data for rarer critical scenarios, for example, car accidents, for which a system failure may have catastrophic consequences. Conceiving perception systems that generalize better, even from limited training situations, that may identify novel environments they cannot understand, and that may adapt to changes and continue training, are some of the many challenges facing autonomous driving, and AI in general.

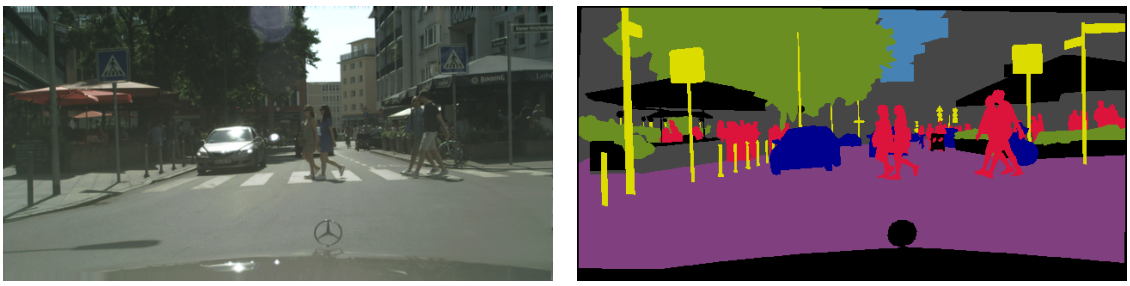


Figure 1.2 – **Semantic segmentation of urban scene.** Each pixel of the image is associated with a category label, represented in different colors on the semantic segmentation map. Image and manually annotated ground-truth semantic segmentation map from the Cityscapes dataset (Cordts et al. 2016b).

In this context, a prime class of tools in the ML community is *frugal learning*. The goal of frugal learning is to drastically reduce the need for data and annotation of fully-supervised learning to train performing models. More specifically, this dissertation focuses on Domain Adaptation (DA). Its goal is to generalize, or adapt, to a set of data, the *target domain*, sharing structures with another annotated dataset, the *source domain*, allowing supervised training though having

some statistical distribution differences. Practically, one could, for example, use DA to train a model on nighttime data as the target domain by taking advantage of annotated daytime data. Similarly, a promising practice in DL is to use DA to leverage synthetic data as the source domain, on which annotation is automatic and cheap, to train models on real-world data as the target domain.

1.3 Outline and Contributions

Many different problems of DA remain open, especially when considering Unsupervised Domain Adaptation (UDA), the setting in which the target domain is completely unlabeled. My Ph.D. work focuses on developing UDA algorithms and architectures that better leverage available data to build more robust models, specifically for the task of semantic segmentation of urban scene images.

While semantic segmentation annotation is essential in the source domain as supervision for the target domain, more knowledge may be leveraged from the source data. For instance, synthetic imagery may produce more modalities, such as depth map or object bounding boxes, that a UDA algorithm may exploit as Privileged Information (PI) (Vapnik and Izmailov 2015) to improve its performance on the semantic segmentation task on the target domain (Chapter 3). Furthermore, semi-supervised learning, another frugal learning paradigm, may shed some light on ways to better use the unlabeled target data of UDA. In particular, Self-Training (ST) (D.-H. Lee 2013) is a strategy that produces pseudo-annotation for unlabeled data based on a pre-trained model's predictions. Building a ST framework in the context of UDA could help to improve existing algorithms (Chapter 4). Finally, due to the variety of scenarios an autonomous vehicle may encounter in the wild, restricting UDA to a single target domain may limit the use cases of these perception models. Finding UDA architectures and algorithms that may adapt to more than one target domain would considerably increase their practicality (Chapter 5).

Outline. This dissertation is structured around these diverse themes of UDA for semantic segmentation I tackled during my Ph.D.:

- Chapter 2 introduces the basics of ML, DL and Transfer Learning (TL), while giving an overview of standard UDA approaches in the literature and making a first focus on UDA for urban scene segmentation;

- Chapter 3 covers UDA scenarios in presence of PI and how to effectively use this additional knowledge to improve UDA methods. The work in this chapter has led to the publication of a conference paper:

  - Taylor Mordan, Antoine Saporta, Alexandre Alahi, Matthieu Cord, and Patrick Pérez (2020). "Bilinear Multimodal Discriminator for Adversarial Domain Adaptation with Privileged Information". In: *Symposium of the European Association for Research in Transportation (hEART)*.

- Chapter 4 discusses how to estimate confident pseudo-labels from a semantic segmentation model trained with UDA techniques and how to effectively improve the performance of such a model with ST using these new annotations. The work in this chapter has led to the publication of a conference workshop paper and a journal paper:

  - Antoine Saporta, Tuan-Hung Vu, Matthieu Cord, and Patrick Pérez (2020). "ESL: Entropy-guided self-supervised learning for domain adaptation in semantic segmentation". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshop on Scalability in Autonomous Driving*.

  - Charles Corbière, Nicolas Thome, Antoine Saporta, Tuan-Hung Vu, Matthieu Cord, and Patrick Perez (2021). "Confidence Estimation via Auxiliary Models". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*.

- Chapter 5 extends the traditional UDA scenario to multiple target domains rather than a single one, both in a multi-target learning context and a continual learning context. The work in this chapter has led to the publication of a conference paper:

  - Antoine Saporta, Tuan-Hung Vu, Matthieu Cord, and Patrick Pérez (2021). "Multi-Target Adversarial Frameworks for Domain Adaptation in Semantic Segmentation". In: *IEEE/CVF International Conference on Computer Vision (ICCV)*.

- Finally, Chapter 6 concludes and proposes some perspectives.

CHAPTER 2

UNSUPERVISED DOMAIN ADAPTATION

Chapter abstract

Transfer Learning (TL) is a well-known research problem in the Machine Learning (ML) community whose focus is to leverage the knowledge acquired by solving one problem to solve another different though related problem. More specifically, Unsupervised Domain Adaptation (UDA) concerns the class of TL problems on which data from the source includes annotations for the considered task, while the target data available is not labeled. This research topic has received increasing attention with the rise of Deep Learning (DL) due to the need for large amounts of training data while an increasing amount of raw data is available, especially in the Computer Vision (CV) community. This chapter starts by laying essential definitions and notations of ML and DL for CV as well as TL and develops some particular cases of TL depending on the parameters of the problem. Then, the chapter goes further into the details of the specific TL scenario of UDA in CV and reviews the multiple core approaches to this problem found in the literature. This review is non-exhaustive and the following chapters go into more details in some more specific UDA topics. Finally, a focus on UDA for urban scene segmentation is made, which introduces the work carried out during the thesis.

Contents

2.1 Deep Learning for Computer Vision

2.1.1 Machine Learning Basics

Machine Learning (ML) (Bishop 2006; Hastie 2009) is the domain that aims at training models to solve various tasks based on given examples. In the context of Computer Vision (CV), ML models are generally given pictures, real or synthetic, and are trained to solve a large variety of problems: classification, semantic segmentation, object detection, etc. (Cord and Cunningham 2008; Szeliski 2010)

Before going further into the details, let's start with a few definitions and notations that will be used in all that follows.

A *domain* $\mathfrak{D}$ is composed of:

- a multi-dimensional space $\mathfrak{X}$, generally some subset of a real d-space $\mathbb{R}^d$;
- a marginal distribution $\mathcal{P}(X)$ on $\mathfrak{X}$.

Moreover, for a given domain $\mathfrak{D}$, a *task* $\mathfrak{T}$ is defined by:

- a label space $\mathfrak{Y}$, which, for example, can be described as some subset of natural numbers $\mathbb{N}$ for classification tasks, a subset of a d-space of natural numbers $\mathbb{N}^d$ for semantic segmentation tasks, or a subset of a real k-space $\mathbb{R}^k$ for regression tasks;

- a conditional probability distribution $\mathcal{P}(Y|X)$.

ML trains a model F with parameters $\mathbf{w}$, having as input elements $\boldsymbol{x} \in \mathfrak{X}$, to produce predictions $F(\boldsymbol{x}) = \hat{\boldsymbol{y}} \in \mathfrak{Y}$ for a given task $\mathfrak{T}$. The objective of the model F is to solve the task $\mathfrak{T}$, i.e. estimating the conditional probability distribution $\mathcal{P}(Y|X)$. Knowing the *ground-truth label* $\boldsymbol{y} \in \mathfrak{Y}$ for a sample $\boldsymbol{x} \in \mathfrak{X}$, the prediction error of the model is quantified by the definition of a *loss function* for the task $\mathfrak{T}$ written $\mathcal{L}_{\mathfrak{T}}(\hat{\boldsymbol{y}}, \boldsymbol{y})$. For instance, for classification tasks with C classes, the usual loss function used is the Cross-Entropy (CE):

$$\mathcal{L}_{\text{CE}}(\hat{\boldsymbol{y}}, \boldsymbol{y}) = -\sum_{c=1}^{C} y_c \log(\hat{y}_c), \tag{2.1}$$

performed after a *softmax* activation in order to convert the C-sized output of the model $\widetilde{\boldsymbol{y}}$ into a vector of probabilities:

$$\forall c \in [C], \quad \hat{y}_c = \text{softmax}(\widetilde{\boldsymbol{y}})_c = \frac{\exp(\widetilde{y}_c)}{\sum_{j=1}^{C} \exp(\widetilde{y}_j)}, \tag{2.2}$$

where $[K]$ is defined as the set of all natural numbers between 1 and K:

$$[K] = \{k \in \mathbb{N} \mid 1 \leq k \leq K\}. \tag{2.3}$$

The output of the model $\hat{\boldsymbol{y}}$ is thus the vector of probabilities of $\boldsymbol{x}$ belonging to each of the C classes while the ground-truth $\boldsymbol{y}$ is a *one-hot* vector, i.e. the value of the coordinate for the true class is 1 and the other values are 0.

In the specific two-class classification case ($C = 2$), the loss is changed. The output of the model $\hat{y}$ has only one dimension and the loss becomes the Binary Cross-Entropy (BCE):

$$\mathcal{L}_{\text{BCE}}(\hat{y}, y) = -y \log(\hat{y}) - (1 - y) \log(1 - \hat{y}), \tag{2.4}$$

where the ground-truth y takes the value 0 for the first class and 1 for the second class.

Semantic segmentation tasks usually extends the CE to a pixel-wise CE as segmentation loss:

$$\mathcal{L}_{\text{seg}}(\hat{\boldsymbol{y}}, \boldsymbol{y}) = \sum_{h=1}^{H} \sum_{w=1}^{W} \mathcal{L}_{\text{CE}}(\hat{\boldsymbol{y}}_{[h,w,\cdot]}, \boldsymbol{y}_{[h,w,\cdot]}), \tag{2.5}$$

where $H \times W$ are the dimensions of the input image $\boldsymbol{x}$ (and thus are also the dimensions of its ground-truth segmentation map $\boldsymbol{y}$), $\boldsymbol{u}_{[i,j,k]}$ is the (i, j, k) compo-

nent of a three-dimensional $\boldsymbol{u}$ and $\boldsymbol{u}_{[i,j,\cdot]}$ is the vector over the last coordinate with fixed two first coordinates $(i, j) \in [I] \times [J]$, defined as:

$$\boldsymbol{u}_{[i,j,\cdot]} = (\boldsymbol{u}_{[i,j,k]})_{k \in [K]}. \tag{2.6}$$

With the task loss defined, the goal of ML is to find the optimal parameters $\mathbf{w}^*$ of F that minimize the expectation of the loss function for the domain and task couple:

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \mathbb{E}_{(X,Y)}[\mathcal{L}_{\mathfrak{T}}(\hat{Y}, Y)] = \arg\min_{\mathbf{w}} \mathbb{E}_{(X,Y)}[\mathcal{L}_{\mathfrak{T}}(F(X), Y)]. \tag{2.7}$$

Practically, supervised ML exploits a training dataset $(\mathcal{X}, \mathcal{Y}) = \{(\boldsymbol{x}^{(n)}, \boldsymbol{y}^{(n)})\}$, $n \in [N_{\text{train}}]$ on which the expectation above is empirically estimated with Monte-Carlo sampling. An optimization algorithm is then employed to minimize the empirical loss function over the training dataset:

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \sum_{n=1}^{N_{\text{train}}} \left[\mathcal{L}_{\mathfrak{T}}(F(\boldsymbol{x}^{(n)}), \boldsymbol{y}^{(n)})\right]. \tag{2.8}$$

In general, after training, the performance of ML models are then evaluated on a test dataset using various metrics depending on the considered task. Moreover, a dedicated validation dataset is often used beforehand to optimize any hyperparameter or architectural detail which was chosen prior to the training to hope for the best performance possible on the test dataset.

2.1.2 Deep Learning and Architectures

Deep Architectures. Deep Learning (DL) (Goodfellow et al. 2016) represents the subset of ML models based on Deep Neural Networks (DNNs). Usually, the Neural Networks (NNs) considered are *feed-forward*: the model F is a succession of transformations, called *layers*, transforming the input $\boldsymbol{x}$ in a sequence of intermediate *representations* or *features* $\boldsymbol{h}_l$ after each layer l.

One of the most common layer is the *dense* or *fully-connected* layer, often noted `fc`. These layers consist in a linear transformation of their input by a weight matrix $\mathbf{w}_l$ and bias $\boldsymbol{b}_l$:

$$\widetilde{\boldsymbol{h}_l} = \mathbf{w}_l \boldsymbol{h}_{l-1} + \boldsymbol{b}_l, \tag{2.9}$$

followed by a non-linear activation function, the most commonly used nowadays being the Rectified Linear Unit (ReLU) activation:

$$\boldsymbol{h}_l = \text{ReLU}(\widetilde{\boldsymbol{h}}_l) = \max(0, \widetilde{\boldsymbol{h}}_l). \tag{2.10}$$

In the CV community, another layer, though introduced decades ago by LeCun et al. (1989), has been popularized by the AlexNet architecture (Krizhevsky et al. 2012) due to the outstanding performance of the model at the time on the ImageNet Large Scale Visualization Challenge in 2012: *convolutional* layers, generally noted `conv`, apply 2D convolutions instead of dense linear functions, allowing to capture local and spatially-aware features, regardless of the spatial position in the image. DNN architectures based on convolutional layers are usually called Convolutional Neural Networks (CNNs).

Finally, *pooling* layers are used to aggregate spatial information and effectively reduce the size of the intermediate representations. The most commonly used pooling strategies are the average pooling, which outputs the average of each patch of the input, and the maximum pooling, which outputs the maximum of each patch of the input.

For CV applications, CNNs have become a staple architecture for most tasks, may it be classification, object detection, semantic segmentation, etc. Models such as VGG-16 (Simonyan and Zisserman 2014), Inception V1 or V3 (Szegedy et al. 2015; Szegedy et al. 2016) or ResNet-50, 101 or 152 (K. He et al. 2016) are still used today as backbone networks for many architectures in order to extract deep features before a task-dedicated NN head.

For instance, popular semantic segmentation models such as PSPNet (H. Zhao et al. 2017) or DeepLabs (L.-C. Chen et al. 2014; L.-C. Chen et al. 2017; L.-C. Chen et al. 2018b) base their architecture on such baseline CNN. For the semantic segmentation task, the CNNs must output a segmentation map of the size of its input. Thus, fully-connected layers are not well suited for this task since they discard the spatiality of their input. In general, semantic segmentation models replace them with convolutional layers, making them Fully-Convolutional Networks (FCNs). For example, DeepLabV2 (L.-C. Chen et al. 2017) uses Atrous Spatial Pyramidal Pooling (ASPP), illustrated in Figure 2.1, to upsample features extracted from either VGG-16 (Simonyan and Zisserman 2014) or ResNet-101 (K. He et al. 2016). ASPP uses in parallel multiple *atrous* convolutional layers: these convolutions use a dilation rate defining a spacing between the values in a kernel, effectively enlarging the field of view of the filter. By employing atrous convolutions with different dilation rates, ASPP can capture objects or image context at multiple scales.

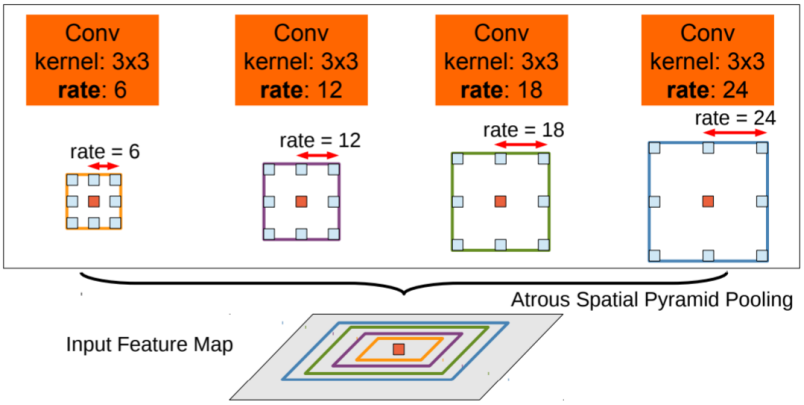


Figure 2.1 – **Atrous Spatial Pyramid Pooling overview**. The center pixel (orange) representation after the ASPP exploits multi-scale features thanks to multiple convolutional filters with different dilation rates in parallel. Illustration from (L.-C. Chen et al. 2017).

Training DNNs. DNNs are trained by using *gradient back-propagation* (Rumelhart et al. 1995). This method exploits the feed-forward property and the chain-rule to compute progressively the gradient of the loss $\nabla_{\mathbf{w}}\mathcal{L}$. The weights are then updated iteratively using a gradient descent algorithm to decrease the value of the loss until a minimum of the objective function is reached:

$$\mathbf{w} \leftarrow \mathbf{w} - \eta \nabla_{\mathbf{w}}\mathcal{L}, \tag{2.11}$$

where the parameter η, called *learning rate*, modulates the step size of each iteration of gradient descent.

Exactly estimating the gradient of the loss over the training dataset is usually impractical due to the amount of training data and the size of the DNNs. Moreover, DNN training losses are complex and non-convex, leading to the existence of many local minima. Numerous gradient descent algorithms have been developed to exploit a stochastic estimation of the loss, the simplest one being Stochastic Gradient Descent (SGD) (Bottou 2010), with many variants designed to improve the stability, training speed, and the local minimum reached after training. The most used algorithms include SGD with momentum (Qian 1999), RMSProp (Tieleman, Hinton, et al. 2012) or Adam (Diederik P. Kingma 2015).

Adversarial Training. Generative Adversarial Networks (GANs) (Goodfellow et al. 2014) introduces a new training framework for generative models via an adversarial process. They train two models in parallel: first, a generative model that aims at capturing the distribution of the data and generates close-to-data outputs; second, a discriminator model that estimates the probability of a sample belonging to the training data rather than being generated by the generative

model. In this adversarial training framework, the generative model seeks to maximize the probability of the discriminator model making a mistake on the generated data. This adversarial training strategy has become popular in the DL community both to train generative models and, more generally, to bring closer together features coming from different domains.

2.1.3 Transfer Learning

In a Transfer Learning (TL) (C. Tan et al. 2018; Zhuang et al. 2020) setting, let's consider two couples of domain and task: the *source*, described as $(\mathfrak{D}_s, \mathfrak{T}_s)$; and the *target*, described as $(\mathfrak{D}_t, \mathfrak{T}_t)$. The core objective of TL is to solve the target problem, i.e. learn to estimate $\mathcal{P}(Y_t|X_t)$, by transferring the knowledge accumulated by learning to estimate $\mathcal{P}(Y_s|X_s)$, considering that the source and the target share some common ground. Generally, it is also assumed that source and target are different, which is formalized as either $\mathfrak{D}_s \neq \mathfrak{D}_t$ or $\mathfrak{T}_s \neq \mathfrak{T}_t$, otherwise one could resort to traditional learning instead. As an analogy, one will have an easier time learning to play the violin if they have already learned to play the cello; and one will learn more easily how to drive a truck if they know how to drive a car, or how to drive on snow if they know how to drive on dry roads. In short, TL aims at leveraging the knowledge of the source domain to solve the problem of the related target domain. Ideally, the advantages of TL compared to traditional learning on the target domain are a faster learning process, more accurate models or a limited need for training data.

Following this general setting and depending on the availability of annotation on the source or target, the next paragraphs describe a few, non-exhaustive, typical scenarios of TL (C. Tan et al. 2018; Zhuang et al. 2020).

Pre-Training and Fine-Tuning. When considering a target problem $(\mathfrak{D}_t, \mathfrak{T}_t)$ with annotated data, a common strategy to boost the efficiency of a supervised learning algorithm is to employ a *pre-training* strategy: instead of initializing the weights of the model at random, one may use those of a model pre-trained on a different source problem $(\mathfrak{D}_s, \mathfrak{T}_s)$ to start the training procedure on the target with already meaningful features. Training such a model on the new target problem is usually referred to as *fine-tuning*. This pre-training and fine-tuning strategy is particularly helpful when the target training data is scarce, such as in *few-shot learning* (Wang et al. n.d.). Nonetheless, this approach may be extended to most DL problems and is almost always employed by the community when using existing DNN as backbone of the models.

Multi-Task Learning. Very similar as above, when both the source data and target data are labeled and from a same domain $\mathfrak{D}_s = \mathfrak{D}_t$ but the tasks are different ($\mathfrak{T}_s \neq \mathfrak{T}_t$), the scenario is generally called *Multi-Task Learning (MTL)* (Caruana 1997). The core principle of MTL is that, by simultaneously training both tasks with shared representations, what is learned for each task should help the model generalize better on other tasks. Note also that due to the symmetry between source and target in this scenario, MTL may consider an arbitrary number of tasks. MTL often considers the different tasks equally, meaning the objective is to perform on every task, though one may only care about performance on one task and do not evaluate the other auxiliary tasks at run-time.

Domain Adaptation. Let's finally consider a last specific setting. When $\mathcal{P}(X_s) \neq \mathcal{P}(X_t)$ but the feature spaces and the tasks are the same ($\mathfrak{T}_s = \mathfrak{T}_t$), the scenario is called *Domain Adaptation (DA)* (Csurka 2017). In this setting, the source is always annotated and the core idea is to transfer the predictive function that can be learned in a supervised fashion on the source to the target domain. Different assumptions may be made on the presence of annotations on the target domain: if the target is fully, though generally scarce, the scenario is supervised DA (Motiian et al. 2017; Morsing et al. 2021); if the target is partially annotated, semi-supervised DA (Saito et al. 2019; B. Li et al. 2021); and finally, if the target is not annotated, Unsupervised Domain Adaptation (UDA), which is the most studied in the literature and the focus of this dissertation. On a side note, while DA traditionnally assumes that the tasks are identical between source and target, specific DA settings may consider some marginally different class labels between source and target ($\mathfrak{Y}_s \neq \mathfrak{Y}_t$) – e.g. class-incremental DA (Kundu et al. 2020) or boundless DA (Bucher et al. 2020) – or unbalance between source and target with regards to their classes ($\mathcal{P}(Y_s|X_s) \neq \mathcal{P}(Y_t|X_t)$) – e.g. class-imbalanced DA (S. Tan et al. 2020). Also, in what follows and for convenience, $\mathfrak{D}_s = (\mathfrak{X}_s, \mathcal{P}(X_s))$ and $\mathfrak{D}_t = (\mathfrak{X}_t, \mathcal{P}(X_t))$ will usually simply be referenced as $\mathfrak{X}_s$ and $\mathfrak{X}_t$ with the initial assumptions in mind and still be called domains.

2.2 Unsupervised Domain Adaptation in the Computer Vision Literature

Knowing that one can train a model to estimate $\mathcal{P}(Y_s|X_s)$ with supervised learning methods thanks to the availability of annotated source data, the crux of DA is to adapt this model to give a good estimation of $\mathcal{P}(Y_t|X_t)$. DA generally rely either on aligning in some way the source and target features of the model to make them indistinguishable or on finding a transformation from the source

domain to the target domain and using this transformation to train the model entirely in the target domain. In this section, we describe multiple UDA approaches to highlight the different paradigms and strategies found in the literature to tackle the adaptation problem on CV tasks and categorize them in three core approaches: direct distribution alignment approaches; image-level approaches to learn transformation between domains, and adversarial approaches to indirectly align the source and target distributions. The objective is to give a global view of the UDA literature by presenting principally architecture designs and some losses that impacted the community by their novelty and performance compared to the state of the art at the time of their publication. We claim by no means to be exhaustive, only a few representative examples of works are presented among the sizeable number of contributions in the field. We also want to mention that, though we identify three core approaches to UDA, they are not mutually exclusive, and some works may adopt more than one. Finally, note that this section mixes UDA approaches for classification and semantic segmentation tasks, each approach may not be directly applicable to any other task than the one it has been designed to solve.

2.2.1 Direct Distribution Alignment Approaches

The first class of approaches consists in performing this distribution alignment by directly minimizing some distance or measure of discrepancy between the two domains. We describe in what follows common metrics and strategies for UDA with direct distribution alignment.

2.2.1.1 Maximum Mean Discrepancy

As far as distances between distribution go, the Maximum Mean Discrepancy (MMD) (Gretton et al. 2012) is a standard metric that has been widely used in UDA applications. Very generally, for a given feature map $\phi(\cdot)$ on a reproducing kernel Hilbert space $\mathcal{H}$, the MMD is empirically estimated on source data $\mathcal{X}_s$ and target data $\mathcal{X}_t$ by the formula:

$$\widehat{\mathrm{MMD}}(\mathcal{X}_s, \mathcal{X}_t) = \left\| \frac{1}{|\mathcal{X}_s|} \sum_{x_s \in \mathcal{X}_s} \phi(x_s) - \frac{1}{|\mathcal{X}_t|} \sum_{x_t \in \mathcal{X}_t} \phi(x_t) \right\|_{\mathcal{H}}. \tag{2.12}$$

In NN applications, the feature map $\phi(\cdot)$ is often applied on the output activation of a chosen layer of the NN instead of the input images and $\mathcal{H}$ is usually a real multi-dimensional space $\mathbb{R}^d$. While the theory behind MMD depending on the

chosen feature map won't be developed further there, we present some UDA approaches exploiting this metric.

Deep Domain Confusion. Tzeng et al. (2014) adapt an AlexNet (Krizhevsky et al. 2012) classification model to the UDA task. Along with the classification loss on the source domain, their architecture minimizes the square of the MMD on an added bottleneck fully-connected layer "fc adapt". This additional loss, dubbed *domain loss*, aligns the distributions coming out of "fc adapt", thus ensuring that source and target domains are indistinguishable, i.e. maximizing the domain confusion. Their architecture is illustrated in Figure 2.2.

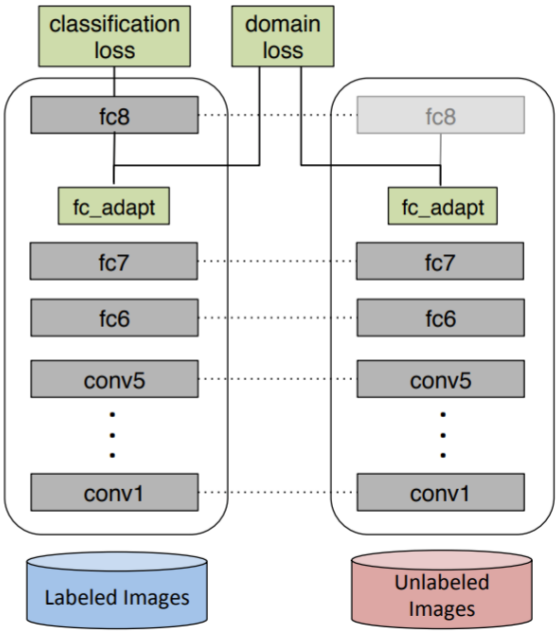


Figure 2.2 – **Deep Domain Confusion network overview**. The CNN architecture adapts AlexNet (Krizhevsky et al. 2012) to enforce domain confusion through an adaptation layer "fc adapt" as well as an additional "domain loss", based on MMD, that minimizes the distance between source features (coming from labeled images, in blue) and target features (coming from unlabeled images, in red). The weights of the models are shared between the source CNN (left) and the target CNN (right), highlighted by the dotted lines between the two models. Illustration from (Tzeng et al. 2014).

Deep Adaptation Networks. Long et al. (2015) go even further than Tzeng et al. (2014). First, instead of focusing on a single layer, their architecture directly aligns the features between source and target on each fully-connected layer of AlexNet

(Krizhevsky et al. 2012). Secondly, they exploit a multi-kernel MMD, allowing for a more discriminative metric. Figure 2.3 illustrates their architecture.

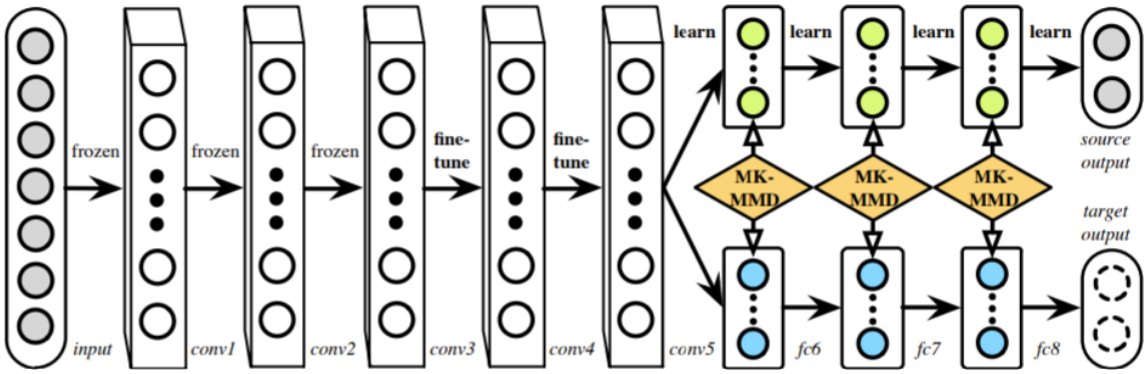


Figure 2.3 – **Deep Adaptation Network overview**. From a partially frozen AlexNet, pre-trained on ImageNet (Krizhevsky et al. 2012), the architecture adapts the fully-connected layers of the model with multi-kernel MMD. Illustration from (Long et al. 2015).

Joint Adaptation Networks. Long et al. (2017) improves over Long et al. (2015) by aligning the joint distribution over the fully-connected layers of AlexNet with a single joint MMD. Figure 2.4 illustrates their architecture.

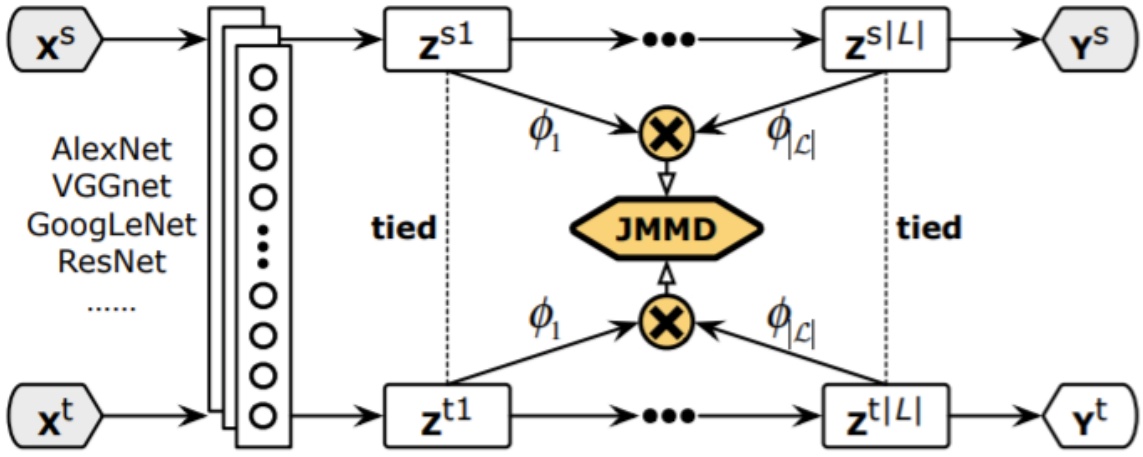


Figure 2.4 – **Joint Adaptation Networks overview**. The architecture aligns the joint distribution of the fully-connected layers of the AlexNet (Krizhevsky et al. 2012) with a single joint MMD: each fully-connected layer i, with output activations $\boldsymbol{z}^{si}$ and $\boldsymbol{z}^{ti}$ for source input $\boldsymbol{x}^s$ and target input $\boldsymbol{x}^s$, respectively, is given a dedicated kernel ϕ_i for the joint MMD. Illustration from (Long et al. 2017).

2.2.1.2 Correlation Alignment

CORAL. First proposed by Sun et al. (2016) specifically for UDA purposes, CORrelation ALignment (CORAL) aims at matching the distributions of source and target by aligning the second-order statistics, i.e. the covariance. Practically, CORAL performs a linear transformation A on the source representations and minimizes the distance between the covariance C_s of the transformed source representations and the covariance C_t of the target representations:

$$\min_A ||A^\top C_s A - C_t||_F^2, \tag{2.13}$$

where $||\cdot||_F^2$ is the Frobenius norm.

Deep CORAL. Sun and Saenko (2016) extend their CORAL loss to DNNs. They define the CORAL loss directly as the distance between the source features and the target features on a given layer of the DNN. The CORAL loss can be minimized directly with the supervised classification loss on the source domain. Figure 2.5 illustrates their approach with the CORAL loss acting at the last layer of the NN.

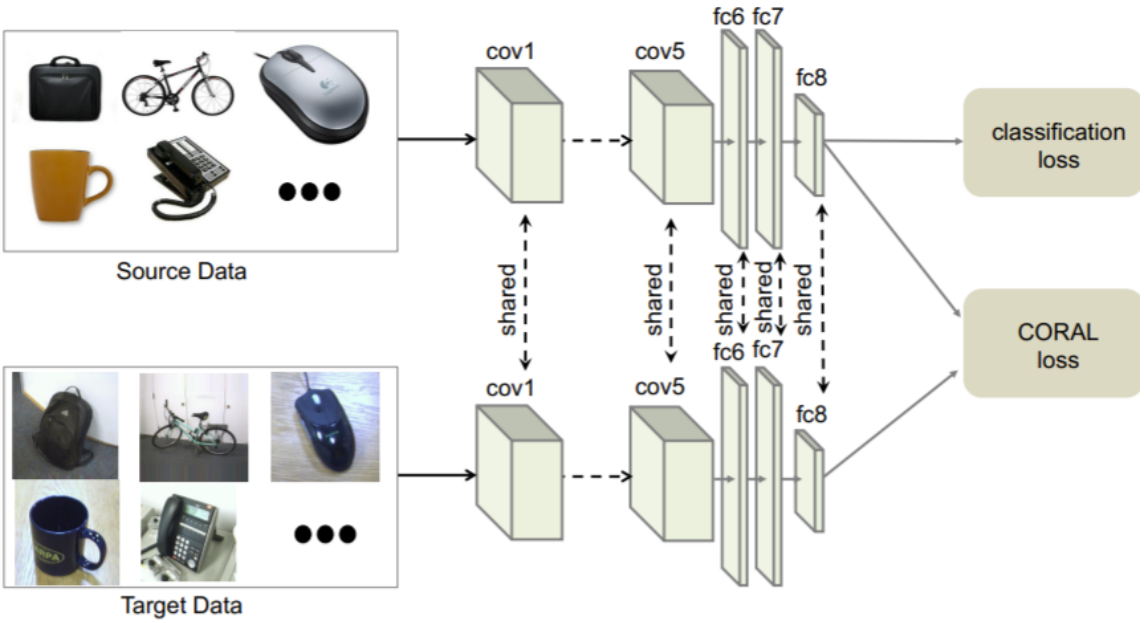


Figure 2.5 – **Deep CORAL network overview**. In this example, the CORAL loss is applied on the output of the NN, the last fully-connected of AlexNet (Krizhevsky et al. 2012). Illustration from (Tzeng et al. 2014).

This approach can be used on multiple layers and different DNN architectures.

2.2.1.3 Optimal Transport for Domain Adaptation

The theory of optimal transport is intuitively simple: the goal is to find the transportation map of minimal cost between two distributions. Optimal transport is an interesting strategy for UDA: using the optimal transportation map from the source domain to the target domain is an elegant way to learn meaningful representations in the target domain by making them closer to those of the source data samples.

Courty et al. (2016) apply optimal transport theory to UDA for image classification. Their method proposes a regularized unsupervised optimal transportation model on the density functions of source and target images which constrains source samples of the same class to remain close during transport, thus ensuring that the transport retains the original semantic. Courty et al. (2017) go further and propose Joint distribution Optimal Transport (JDOT). This method uses optimal transportation to find a transformation between the joint feature space and label space distributions of source and target domains, thus allowing to jointly bring both the source feature distribution and classification distribution closer to those of the target domain.

DeepJDOT. Damodaran et al. (2018) extend the theory developed by Courty et al. (2017) and apply it on deep neural networks. Their method DeepJDOT is illustrated in Figure 2.6. They perform optimal transport on the joint distribution of latent representation and label spaces using an optimal transport solver, giving a coupling matrix γ of size $|\mathcal{X}^s| \times |\mathcal{X}^t|$. This coupling matrix represents the optimal transport function from the source data to the target data: γ_{ij} is close to 1 if the source image $\boldsymbol{x}_i^s$ is transported onto $\boldsymbol{x}_j^t$ with optimal transport. This coupling matrix serves as a weight to the loss function on the target data for the deep neural network training. For each source-target image pair $(\boldsymbol{x}_i^s, \boldsymbol{x}_j^t)$, the authors apply a constraint on the extracted features from g in order to bring closer the features of $\boldsymbol{x}_i^s$ and $\boldsymbol{x}_j^t$ if γ_{ij} is close to 1. Moreover, they apply the classification loss on the unlabeled target data image $\boldsymbol{x}_j^t$ by using the ground-truth of the source image $\boldsymbol{y}_i^s$, yet again weighted by γ_{ij} which is close to 1 if $\boldsymbol{x}_i^s$ is close to $\boldsymbol{x}_j^t$ after optimal transport.

2.2.2 Image-Level Approaches

The second class of approaches consists in building transformations at the image level, either by directly translating the source image into the target domain before the NN model dedicated to solving the task ("task model") or by finding representations that allow reconstructing of images in either domain, indepen-

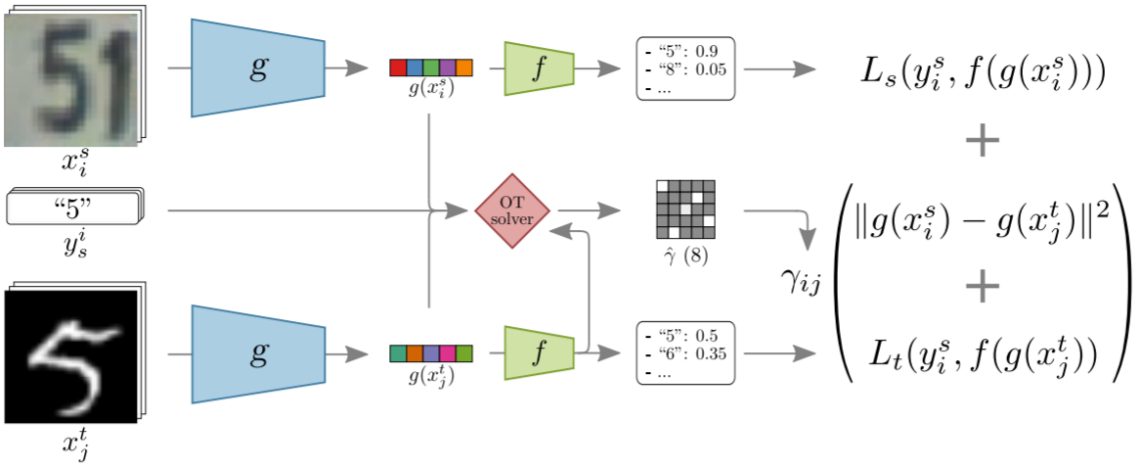


Figure 2.6 – **DeepJDOT network overview**. The latent representations and labels are used to compute with optimal transport a coupling matrix γ, used as weight on the loss function to align the representations of target images to source images which are close to them after transportion. A classification loss on the target domain is also added by using the source labels of close source images, still weighted by the coupling matrix γ. Illustration from (Damodaran et al. 2018).

dently of their domain of origin. We describe in what follows common image-level strategies for UDA.

2.2.2.1 Image-to-Image Translation

In image-to-image translation approaches, the source images are translated to the target domain before training the task model. Such approaches usually make the additional assumption that this transformation does not alter the labels of the source data. This way, one can train the model in the target domain using the supervision of this translated source data with their original annotations.

Pixel Domain Adaptation. Bousmalis et al. (2017) use a GAN (Goodfellow et al. 2014) to translate source domain images into the target domain. The target domain images (real) are only used to train an image generator G: G takes as input a source image (synthetic) and a random noise to produce "fake" images, which are constrained to resemble target domain images (real) by adversarially training a discriminator network D that distinguishes between real target images and generated images. In parallel, the classifier model T is trained on both the source images (synthetic) and the generated target-like images from these source images (fake) paired with the original source annotations. Figure 2.7 illustrates their architecture.

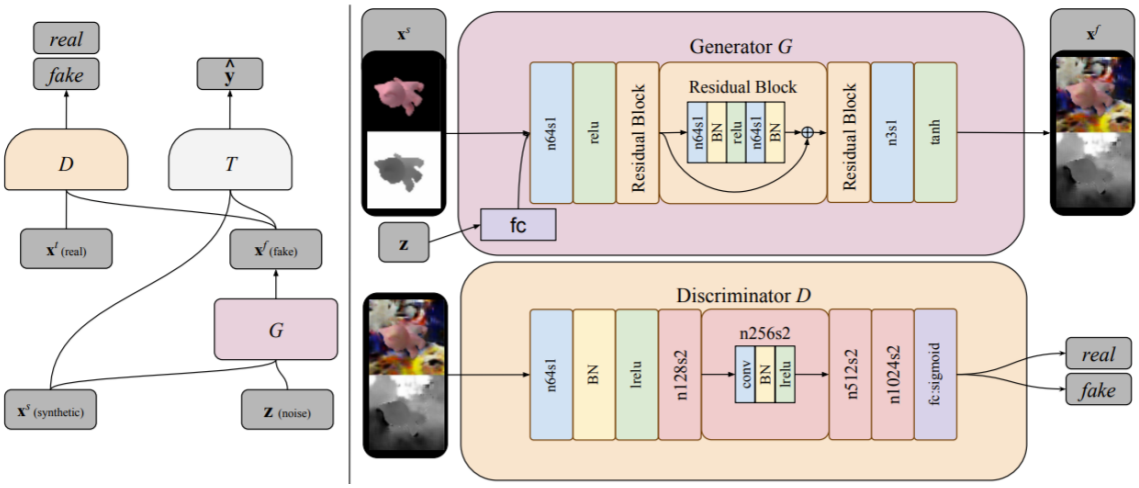


Figure 2.7 – **PixelDAN network overview**. A generator network G learns to transform source images (synthetic) into target-style images (fake). A discriminator D serves as adversary to train the generator G in a GAN approach. The classification model T is trained both on source images (synthetic) and generated images (fake) by consdering that the labels are not impacted by the translation. Illustration from (Bousmalis et al. 2017).

Cycle-Consistent Adaptation. Similarly, Hoffman et al. (2018) use CycleGAN (Zhu et al. 2017) to translate source images into the target domain style, style gaps coming, for instance, from considering a synthetic domain and a real domain, or different weathers such as clear and foggy. Their cycle-consistent stylization strategy is illustrated in red on Figure 2.8. A source-to-target generator $G_{S \to T}$ is trained to stylize source images into the target style, which are then used to train the task model with the original labels. In green on Figure 2.8, an adversary discriminator D_T ensures that the generated images are indistinguishable from the target images. To finalize the cycle-consistent stylization, still in red on Figure 2.8, as in CycleGAN (Zhu et al. 2017), a target-to-source generator $G_{T \to S}$ is also trained to reconstruct the original source images from the translated target-style images. A similar cycle is performed during training on the target images to the source style (not illustrated on Figure 2.8). Additionally, represented in grey on Figure 2.8, a semantic consistency loss, based on a segmentation model pre-trained on source data, is added between images and their generated stylized images to ensure that the semantics are not impacted by the transformation. Finally, the authors' method CyCADA also performs feature-space adversarial alignment, in orange on the Figure 2.8, which is described in more detail in Section 2.2.3.1.

Dual Channel-wise Alignment. Wu et al. (2018) also have a similar idea but with a different approach. Figure 2.9 illustrates their architecture. Instead of opt-

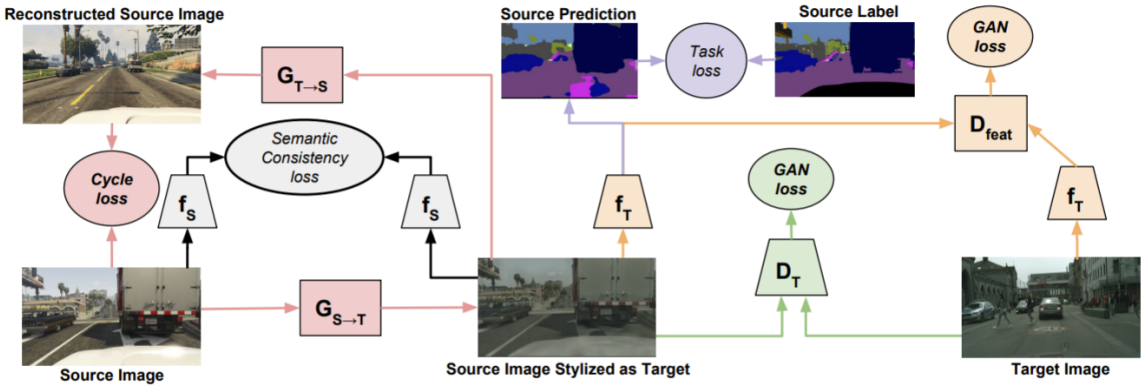


Figure 2.8 – **CyCADA network overview**. Inspired by CycleGAN (Zhu et al. 2017), the architecture performs cycle-consistent adversarial stylization of source image into target style (in red and green) and trains the semantic segmentation model on this stylized data (in yellow and purple). Illustration from (Hoffman et al. 2018).

ing to use GANs for image translation, their method DCAN uses an image generator based on adaptive instance normalization (Huang and Belongie 2017): given an arbitrary target image (orange arrows on Figure 2.9), the intermediate feature means and standard deviations of the source image being translated (blue arrows on Figure 2.9) are matched to the statistics of a target image. Then, the segmentation network (green arrows on Figure 2.9) is trained using these translated source images, considering that the label maps are unchanged by the stylization.

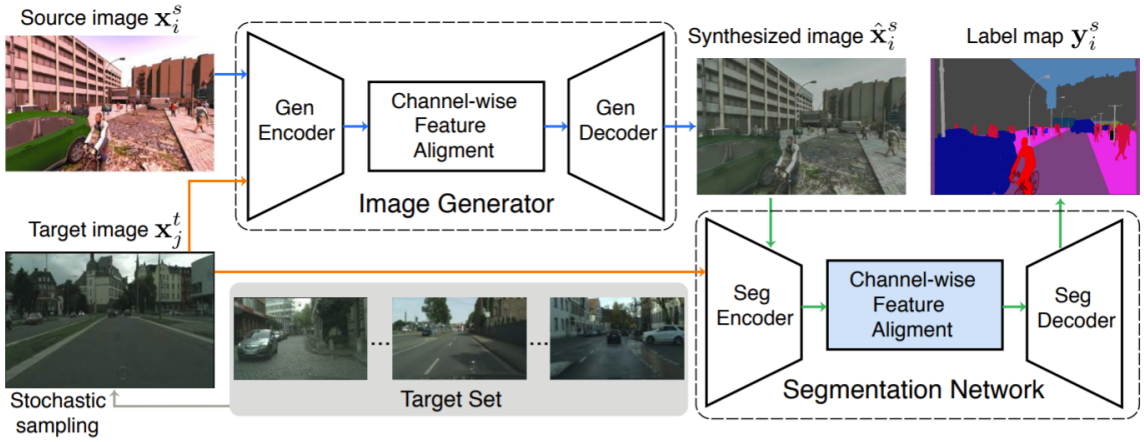


Figure 2.9 – **Dual Channel-wise Alignment Network overview**. The model adapts the style of source images (blue arrows) into target style through an image generator conditionned on a target image (orange arrows) using adaptive instance normalization (Huang and Belongie 2017). The segmentation network is then trained on these synthesized images (green arrows) while considering that the label maps are untouched by the stylization. Illustration from (Wu et al. 2018).

Fourier Domain Adaptation. Y. Yang and Soatto (2020) adopt a radically different strategy to image-to-image translation. Indeed, their method FDA relies on "spectral transfer": the low-frequency components of the Fourier amplitude of the source image are swapped with the ones of the target image. They argue that the low-frequencies are responsible for the style of the image and can be changed significantly without affecting the semantics, thus allowing to keep the original annotations on the stylized source. Their approach is illustrated in Figure 2.10.

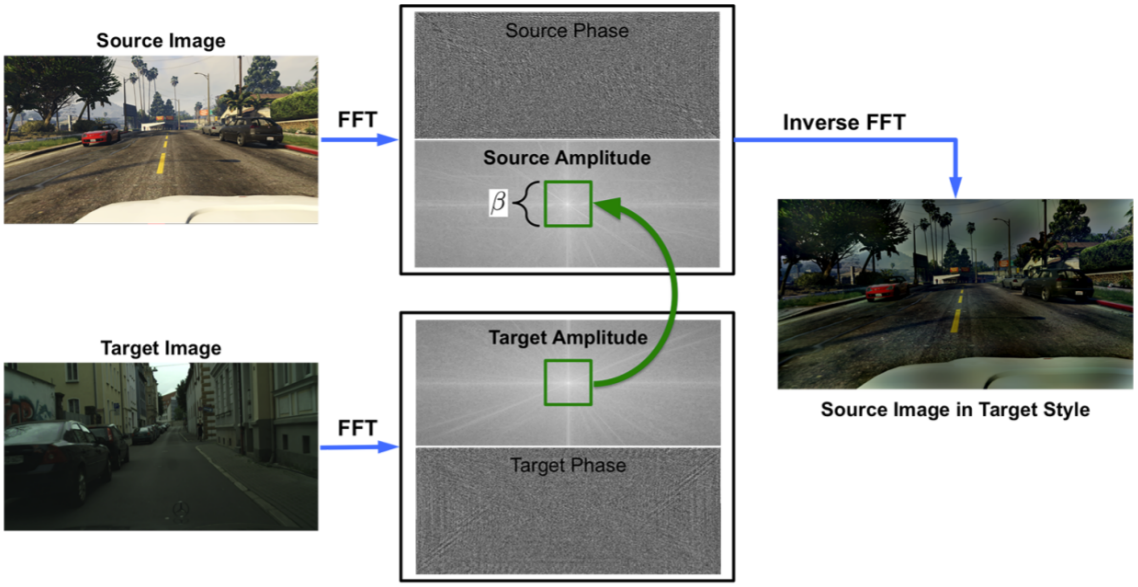


Figure 2.10 – **Fourier Domain Adaptation overview**. *Spectral transfer* is performed to transfer the target style onto the source image without impacting its semantic content. The low-frequency components of the Fourier amplitude of the source image are swapped with the ones of a target image. Illustration from (Y. Yang and Soatto 2020).

Style-swap. M. Kim and Byun (2020) propose to use another style transfer method: Style-swap (T. Q. Chen and Schmidt 2016). This style transfer method is an image synthesis technique that reproduces the content of an image while applying the style of another. The approach of M. Kim and Byun (2020) to UDA for semantic segmentation using Style-swap is illustrated in Figure 2.11. On the left part of Figure 2.11, they use Style-swap to stylize source images with random styles from the *Painter by Numbers* dataset [1] (Stylized source data in Figure 2.11) and to stylize source images as target domain data from a sampled target image (Translated source data in Figure 2.11). Yet again, it is assumed that those transformations do not change the label maps of the images. An additional alignment

1. Painting dataset sourced primarily from `wikiart.org` by Small Yellow Duck (Kiri Nichol) and hosted by Kaggle.

is then performed with output-space discrimination (right part of Figure 2.11), described in more details in Section 2.2.3.2.

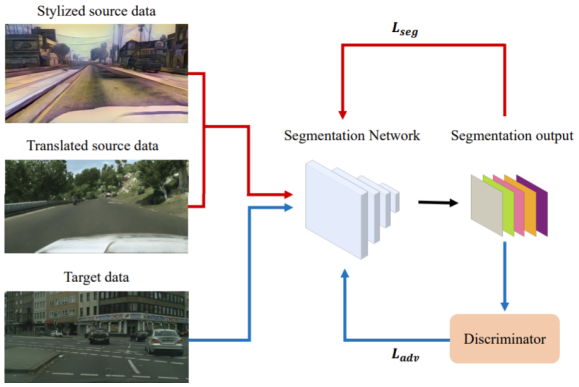


Figure 2.11 – **Style-swap approach overview**. On the left part, the architecture uses Style-swap (T. Q. Chen and Schmidt 2016) to stylize source images with random styles from the *Painter by Numbers* dataset (Stylized source data) and to translate source images into target style (Translated source data). On the right part, the segmentation network is then trained with output-space adversarial alignment (Section 2.2.3.2). Illustration from (M. Kim and Byun 2020).

LAB-color space stylization. J. He et al. (2021) use yet another strategy for image translation. Closer to (Y. Yang and Soatto 2020), their work introduces LAB-color space stylization: to convert a source image into target-style given a target image, after converting the source image and the target image into the LAB color-space, they swap the mean and standard deviation of the source image with the ones of the target image and finally re-convert the source image into RGB-color space.

2.2.2.2 Reconstruction Approaches

Reconstruction-based approaches usually focus on finding representations that allow for image reconstruction in either source or target domain. With the ability to reconstruct images to either domain, such representation only encompass the semantics of the input data.

Domain Separation Network. Bousmalis et al. (2016) construct different representations for both source and target data. Figure 2.12 illustrates their architecture. First, a shared encoder E_c is used to extract both domain representations in a common space that ideally encompasses the content of the images. These common representations are the ones used by the classifier G. Secondly, two pri-

vate encoders, E_p^s and E_p^p for source and target domains, respectively, extract style representations that are used, when summed with the content representations, to reconstruct the original images through a shared decoder D. Some orthogonality constraints are applied between private and shared representations to ensure that both encoders encode different aspects of their inputs. Moreover, a similarity constraint is also applied between the shared representations of source and target with MMD, similarly to what is discussed in Section 2.2.1.1.

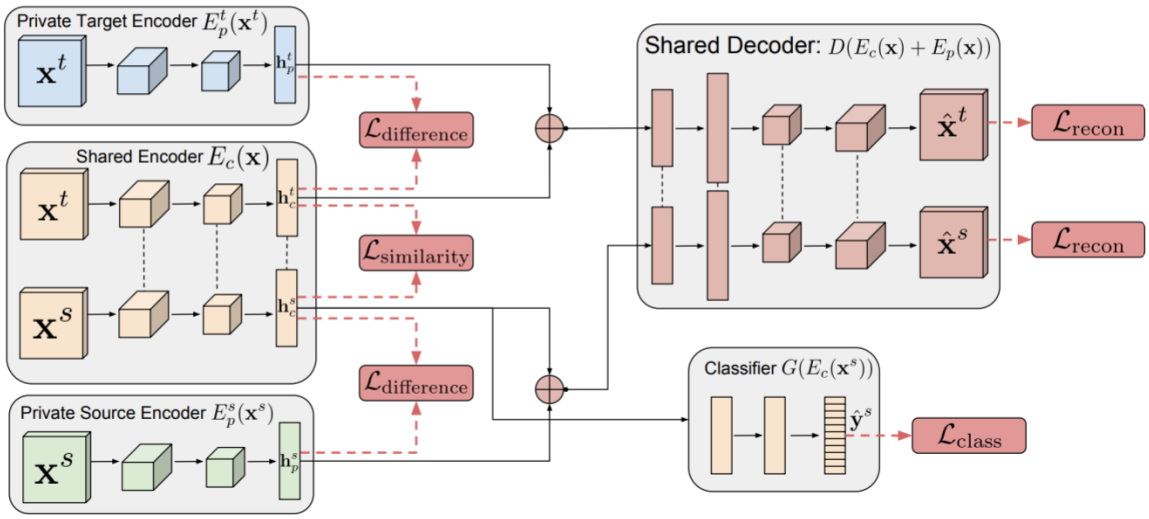


Figure 2.12 – **Domain Separation Network overview**. The architecture uses a shared encoder E_c and private encoders E_p^s and E_p^t to disentangle the representations of source and target domains into components used for the classification task (representation of the *semantics* of the input) and components used for reconstruction by a shared decoder D (representation of the *style* of the input). Illustration from (Bousmalis et al. 2016).

Domain Invariant Structure Extraction. Chang et al. (2019) base their architecture on a feature disentangling strategy, as in (Bousmalis et al. 2016), but train it with different objectives. As Bousmalis et al. (2016), their model DISE has a shared encoder extracting structure content representations that are used for solving the task. It also features two domain-specific encoders extracting texture appearance representations. Moreover, they train a single decoder that takes as input a structure content representation and a texture appearance representation to output an image. Their approach uses this decoder to reconstruct the source and target images, but also, following ideas from Section 2.2.2.1, to translate the source image into target-style by using the source content and the target texture representations, and similarly to translate the target image into source style. Finally, the task

model is also trained on the source image translated into target style. Figure 2.13 illustrates their architecture.

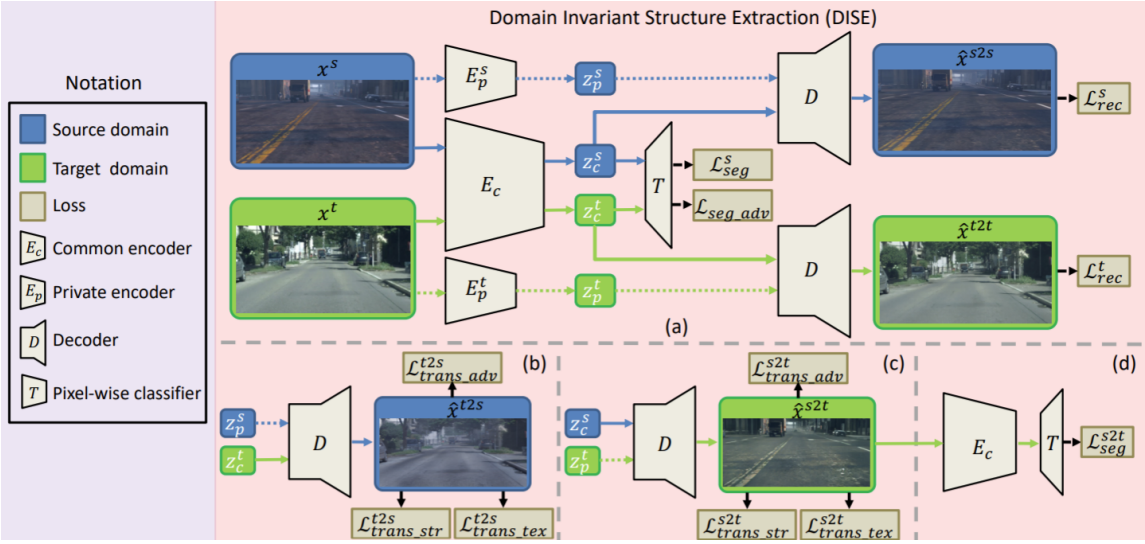


Figure 2.13 – **Domain Invariant Structure Extraction overview**. The architecture uses a shared encoder E_c to extract structure content components and private encoders E_p^s and E_p^t to extract texture appearance components from the source and target images. While only the structure content part is used by the segmentor T, a single shared decoder D is used to construct and reconstruct images using the multiple structure content representation z_c and texture appearance representation z_p pairs. Illustration from (Chang et al. 2019).

Image-to-Image Adaptation. Murez et al. (2018) mix ideas from reconstruction and image-to-image translation approaches. Their method consists in encoding both source and target domains into a single shared embedding space from which one can reconstruct images in either source or target style with a double CycleGAN approach. The task is then solved from this shared embedding space. Their method is illustrated in Figure 2.14.

2.2.3 Adversarial Approaches

Finally, the last class of UDA approaches is based on adversarial training to align the distributions of some representations of source and target domains. Usually, the representations selected for adversarial alignment are either deep features extracted after the DNN backbone of the model right before the last layers dedicated to the task or directly close-to-prediction representations at the output of the task model. In such approaches, besides the task model F, an additional

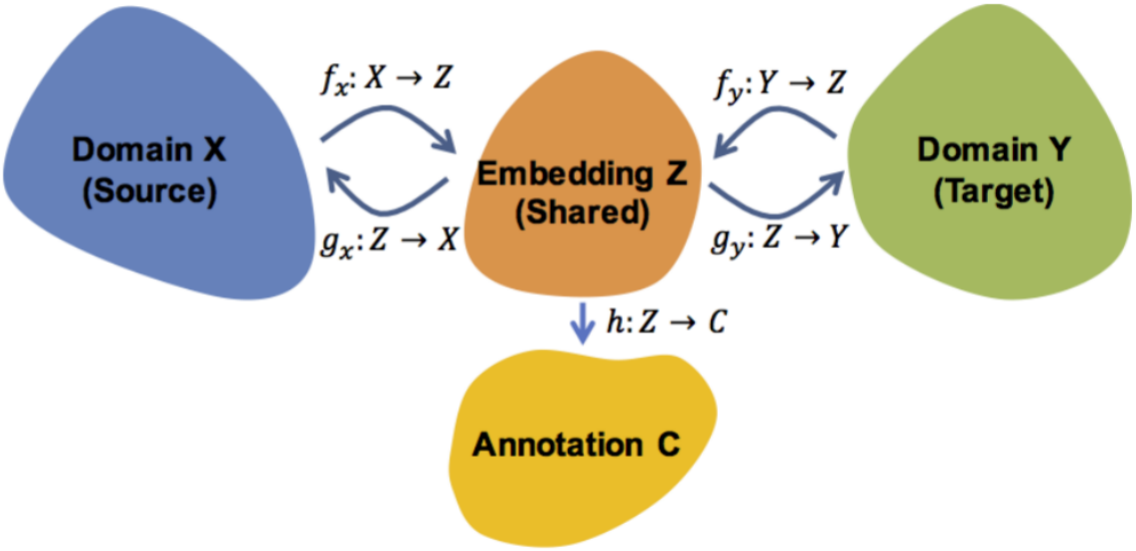


Figure 2.14 – **Image-to-Image Adaptation overview**. Both source and target images are embedded into a common shared space from which source or target images can be reconstructed. Illustration from (Murez et al. 2018).

network D, called discriminator, is trained to play the model's "adversary": D is learned to predict the domain of input from the features extracted from F. Concurrently, F tries to produce results that can fool D into wrong discrimination.

2.2.3.1 Feature-Space Adversarial Alignment

Gradient Reversal Layer. Ganin and Lempitsky (2015), whose method is illustrated in Figure 2.15, introduce a domain classifier G_d (in pink on Figure 2.15) attached to intermediate features of the classification NN, after the feature extractor G_f (in green on Figure 2.15). As its name suggests, it is trained in parallel with the main model to classify the domain of the input. Furthermore, a "gradient reversal layer" changes the sign of the gradients coming from the domain classifier during back-propagation and flowing into the feature extractor, effectively changing the objective of the feature extractor over this domain classification loss: the feature extractor tries to fool the domain classifier.

Adversarial Discriminative Domain Adaptation. Tzeng et al. (2017) adopt an adversarial approach inspired by GANs, illustrated in Figure 2.16. Their strategy is decomposed into three steps. First, they pre-train a source feature extractor (Source CNN) and a classifier on the source. Both models are frozen after this pre-training. Second, using the frozen Source CNN, both a target feature extractor (Target CNN) and a discriminator are trained. While this discriminator must

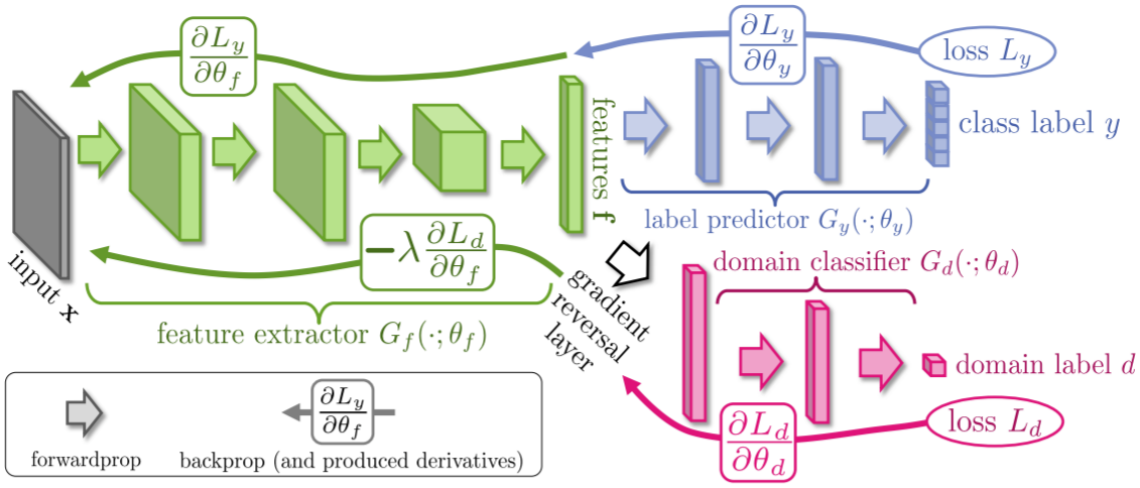


Figure 2.15 – **Gradient reversal layer overview**. A domain classifier G_d, in pink, takes as input the intermediate features of the model, after the feature extractor G_f, in green. Its gradient is backpropagated into the feature extractor through a gradient reversal layer, making the two networks adversaries. Illustration from (Ganin and Lempitsky 2015).

distinguish between source features and target features, the target extractor must fool this discriminator and make indistinguishable the target and source features. Third, for testing on the target domain, the final model consists of this Target CNN and pre-trained classifier from the first step.

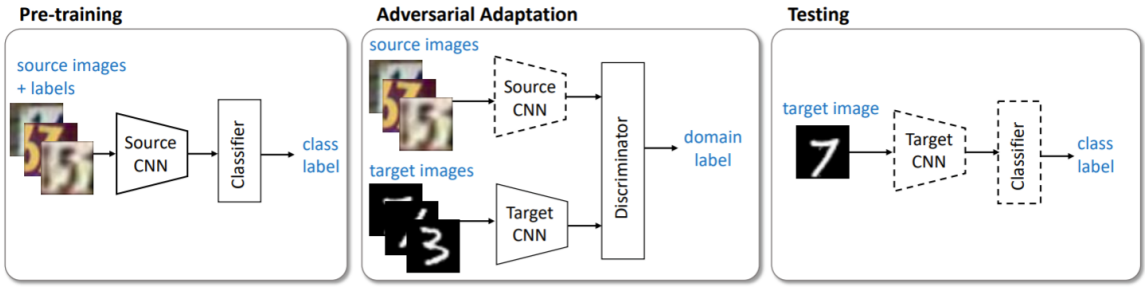


Figure 2.16 – **Adversarial Discriminative Domain Adaptation overview**. The GAN-inspired approach consists in three steps. First, a classification model (Source CNN + Classifier) is pre-trained on the source data. Second, both a Target CNN and a discriminator are trained in an adversarial fashion using the frozen Source CNN in order to produce indistinguishable features between source and target domains. Third, for testing on target data, the classification model consists in the Target CNN and the classifier pre-trained in the first step. Illustration from (Tzeng et al. 2017).

FCNs in the Wild. Hoffman et al. (2016), illustrated in Figure 2.17, train iteratively and alternately a semantic segmentation network, shared between source and target domains, and a discriminator on the extracted features to perform domain adversarial training. Moreover, to further constrain the constructed segmentation maps in the target domain, the label statistics from the source domain are enforced on the target domain with an additional constrained multiple instance loss (constrained MI loss), encouraging the predictions of the target domains to have a label distribution within the expected range observed in the source domain.

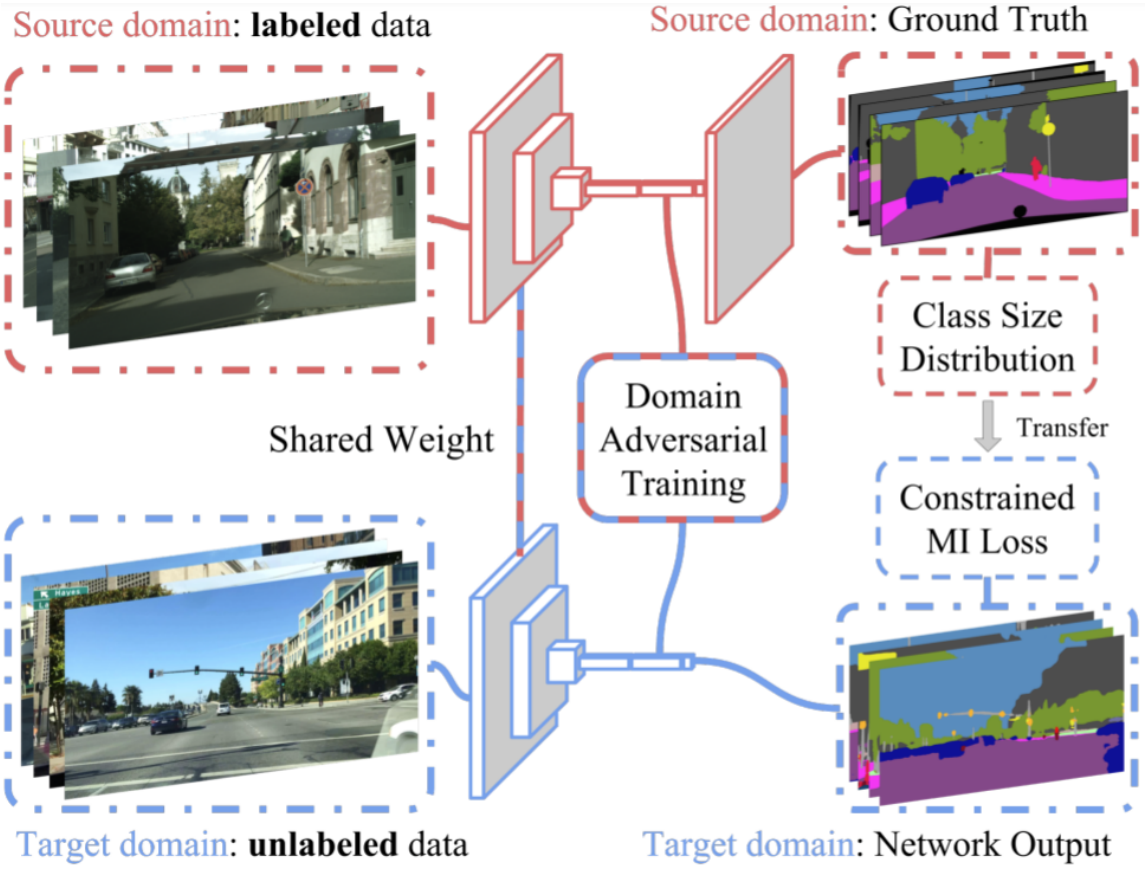


Figure 2.17 – **FCNs in the Wild network overview**. Domain adversarial training is performed on intermediate features of the segmentation network, shared between the two domains. Additionally, the class distribution of the target domain are aligned onto the one of the source data with what the authors call constrained multiple instance loss, encouraging the predictions on the target domain to have a label distribution within the expected range observed in the source domain. Illustration from (Hoffman et al. 2016).

2.2.3.2 Output-space Adversarial Alignment

In semantic segmentation, adversarial approaches operating on close-to-prediction representations have had the most success. We present two particular state-of-the-art approaches following this paradigm.

AdaptSegNet. Adversarial-based approaches to UDA focus on enforcing that features extracted at a particular level of the neural network are made indistinguishable between source and target domains. To this aim, Tsai et al. (2018) consider semantic segmentation maps as structured outputs containing spatial similarities between source and target domains and propose to perform adversarial alignment on the segmentation softmax outputs of the model. Moreover, to improve further the performance of the adapted model, their model AdaptSegNet constructs a multi-level adversarial network with domain adaptation modules at multiple feature levels, each computing segmentation outputs and aligning them adversarially between source and target domains. Figure 2.18 describes the architecture of this network.

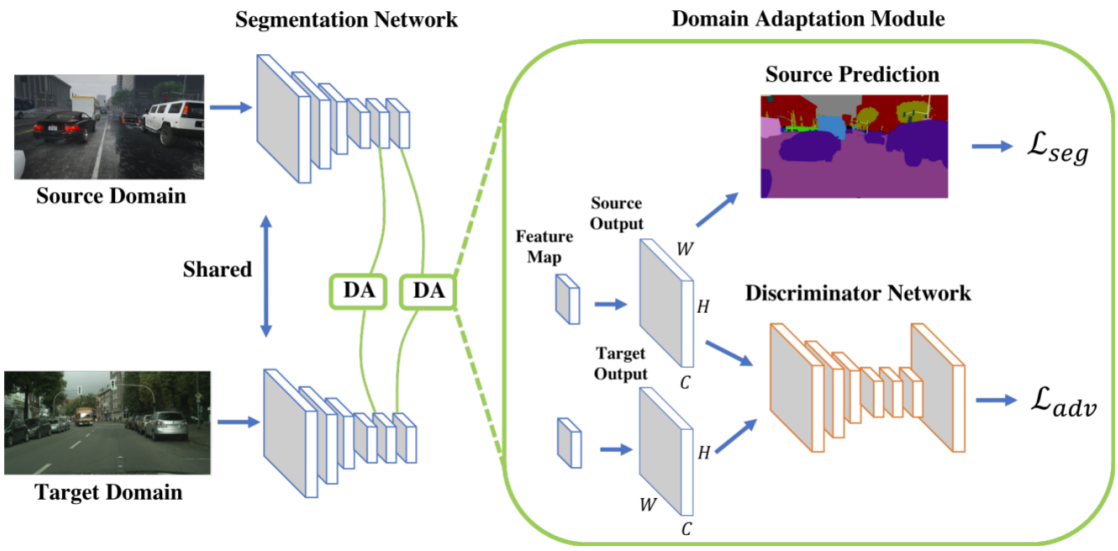


Figure 2.18 – **AdaptSegNet architecture overview**. Images are passed through the segmentation network to obtain output predictions into a DA module. Source predictions only are used to compute a segmentation loss $\mathcal{L}_{seg}$ based on the source ground-truth. Target predictions are brought closer to the source ones thanks to an adversarial loss $\mathcal{L}_{adv}$ on a discriminator network trained to distinguish whether the input comes from the source or the target domain. Illustration from (Tsai et al. 2018).

AdvEnt. Inspired by (Tsai et al. 2018), Vu et al. (2019a) base their adaptation strategy on output space features. However, instead of considering the segmentation softmax outputs, they develop two UDA strategies based on the entropy of the pixel-wise predictions, or more specifically on the weighted self-information maps of the outputs. The first one, Minimizing Entropy (MinEnt), proposes to directly minimize the entropy on the target domain, which can be seen as a soft-assignment version of a pseudo-label cross-entropy loss (cf. Chapter 4 for more details on training with pseudo-labels). The second one, Adversarial Entropy Minimization (AdvEnt), proposes to minimize the entropy by performing adversarial training on the weighted self-information maps between source and target domains. Both approaches are illustrated in Figure 2.19.

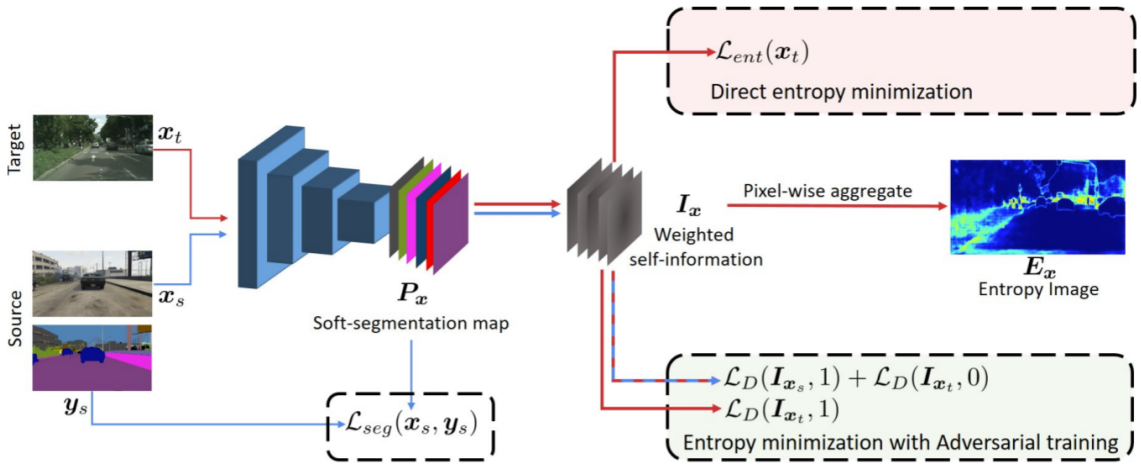


Figure 2.19 – **MinEnt and AdvEnt architecture overview**. For each input image, the weighted self-information is computed from the prediction output of the segmentation network and used to minimize the entropy of the minimzation with two approaches. In MinEnt, the entropy of the predictions is directly minimized by minimizing the weighted self-information maps. In AdvEnt, adversarial training is performed to enforce the consistency of the weighted self-information maps between domains. Red arrows are used for target domain and blue arrows for source domain. Illustration from (Vu et al. 2019a).

2.3 Unsupervised Domain Adaptation for Urban Scene Segmentation

2.3.1 Context

2.3.1.1 Specific Issues of Semantic Segmentation

While classification requires to produce image-level predictions and object detection requires producing bounding-box localization and bounding-box-level categories, semantic segmentation involves pixel-level prediction, which poses a unique set of challenges while addressing UDA. With the need to produce input-sized outputs, semantic segmentation models need to keep a large amount of information in their features to produce precise enough pixel-wise predictions, leading to very large feature spaces. Furthermore, due to the co-occurence of categories in semantic segmentation compared to classification, UDA may have to deal with uneven domain shifts for each category, coexisting within individual target inputs. Practically, having to deal with networks with large feature spaces automatically requires a lot of computing power and large Graphics Processing Unit (GPU) memory to store the features for each image of the batch for SGD. With limited GPU memory, one has to consider small mini-batches of a couple of images during training. All in all, training semantic segmentation models is tricky and demands task-specific methodologies and dedicated UDA architectures.

2.3.1.2 Output-Space Adversarial Baselines

Over the past few years, output-space adversarial alignment (Section 2.2.3.2) has been used by many methods as a baseline UDA strategy for semantic segmentation. AdaptSegnet (Tsai et al. 2018) proposes to have adversarial learning on top of the soft-segmentation predictions $F(\boldsymbol{x}) = \boldsymbol{P}_{\boldsymbol{x}}$. AdvEnt (Vu et al. 2019a) uses the weighted self-information maps $\boldsymbol{I}_{\boldsymbol{x}}$:

$$\boldsymbol{I}_{\boldsymbol{x}} = -\boldsymbol{P}_{\boldsymbol{x}} \log \boldsymbol{P}_{\boldsymbol{x}}, \tag{2.14}$$

with entrywise multiplication and logarithm. More generally, we denote $\boldsymbol{Q}_{\boldsymbol{x}}$ the used representation, which stands for either $\boldsymbol{P}_{\boldsymbol{x}}$ in (Tsai et al. 2018) or $\boldsymbol{I}_{\boldsymbol{x}}$ in (Vu et al. 2019a). Such adversarial frameworks serve as the building block on top of which we developed most of our work. Note nevertheless that many of our contributions may be extended to other UDA architectures and approaches, which will be specified in the following chapters when necessary.

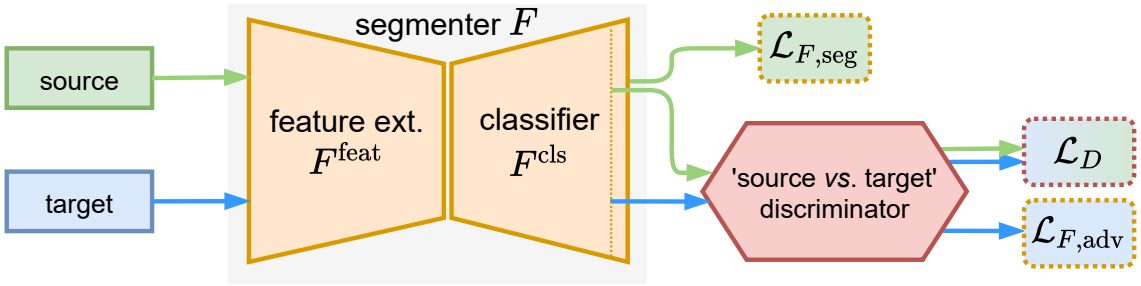


Figure 2.20 – **Output-space adversarial alignment approach to UDA**. The segmentation model under training ingests source-domain (green) and target-domain (blue) data. The former contribute to the segmentation loss, the latter to the adversarial loss, and both to the discriminator's loss. The three losses (dotted boxes) are defined in Equation 2.15 and Equation 2.16.

In practice, the discriminator D is a fully-convolutional binary classifier with parameters ϕ. It classifies segmenter's output $\boldsymbol{Q}_{\boldsymbol{x}}$ into either class 1 (source) or 0 (target). To train the discriminator on the source dataset $\mathcal{X}_s$ and target dataset $\mathcal{X}_t$, one minimizes the classification loss:

$$\mathcal{L}_D = \frac{1}{|\mathcal{X}_s|} \sum_{\boldsymbol{x}_s \in \mathcal{X}_s} \mathcal{L}_{\text{BCE}}(D(\boldsymbol{Q}_{\boldsymbol{x}_s}), 1) + \frac{1}{|\mathcal{X}_t|} \sum_{\boldsymbol{x}_t \in \mathcal{X}_t} \mathcal{L}_{\text{BCE}}(D(\boldsymbol{Q}_{\boldsymbol{x}_t}), 0), \tag{2.15}$$

where $\mathcal{L}_{\text{BCE}}$ is the BCE loss defined in Equation 2.4.

Concurrently, the semantic segmentation model, or segmenter, F is trained over its parameters θ not only to minimize the supervised segmentation loss $\mathcal{L}_{F,\text{seg}}$ on source-domain data, but also to fool the discriminator D via minimizing an adversarial loss $\mathcal{L}_{F,\text{adv}}$. The final objective reads:

$$\mathcal{L}_F = \underbrace{\frac{1}{|\mathcal{X}_s|} \sum_{\boldsymbol{x}_s \in \mathcal{X}_s} \mathcal{L}_{\text{seg}}(\boldsymbol{P}_{\boldsymbol{x}_s}, \boldsymbol{y}_s)}_{\mathcal{L}_{F,\text{seg}}} + \lambda_{\text{adv}} \underbrace{\frac{1}{|\mathcal{X}_t|} \sum_{\boldsymbol{x}_t \in \mathcal{X}_t} \mathcal{L}_{\text{BCE}}(D(\boldsymbol{Q}_{\boldsymbol{x}_t}), 1)}_{\mathcal{L}_{F,\text{adv}}}, \tag{2.16}$$

with a weight λ_{adv} balancing the two terms; $\mathcal{L}_{\text{seg}}$ is the pixel-wise CE defined in Equation 2.5. During training, one alternately minimizes the two losses $\mathcal{L}_D$ and $\mathcal{L}_F$.

Figure 2.20 provides a high-level view of the training flow in output-space adversarial alignment UDA approaches. For more details, we refer the readers to (Tsai et al. 2018) or (Vu et al. 2019a).

To later facilitate the presentation of some of the proposed strategies, the segmenter F may be decoupled into a feature extractor, F^{feat}, followed by a pixel-wise classifier, F^{cls}.

2.3.2 Urban Scene Semantic Segmentation Datasets

Historically, UDA in CV was mostly popular in classification applications and the methods were generally evaluated on the office environment object classification datasets like Office-31 (Saenko et al. 2010) or on numbers classification datasets, such as MNIST (LeCun et al. 1989), SVHN (Netzer et al. 2011) or USPS (Hull 1994). With the growing interest in autonomous driving applications of DL, urban scene semantic segmentation datasets have become popular in the community. In this real-world context with many sources of domain gaps and critical consequences on safety when failing to handle them, UDA in multiple specific and practical scenarios have been studied, the most prominent being synthetic-to-real and city-to-city. This section describes the datasets that will be used in the experiments of the following chapters.

2.3.2.1 Cityscapes

Cityscapes (Cordts et al. 2016a) contains 2048×1024-sized labeled urban scene images from cities around Germany, split in training and validation sets of 2,975 and 500 samples respectively. Human annotations are based on 34 classes, divided in 7 high-level categories (or super classes). While the segmentation maps for training examples are available, note that they are not used in UDA applications where Cityscapes is used as the target domain. Nonetheless, Cityscapes may also be used as a source domain in some experiments. Figure 2.21 shows example images from the Cityscapes dataset. Table 2.1 describes the class labels and mappings of the dataset.

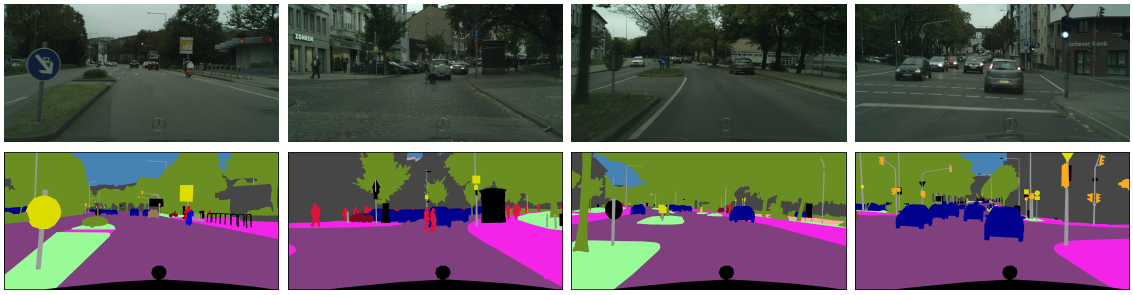


Figure 2.21 – **Example of images from the Cityscapes dataset.** First row: Real European urban scene image; Second row: Associated 19-class semantic segmentation map provided by human annotators (colormap defined in Figure 2.22).

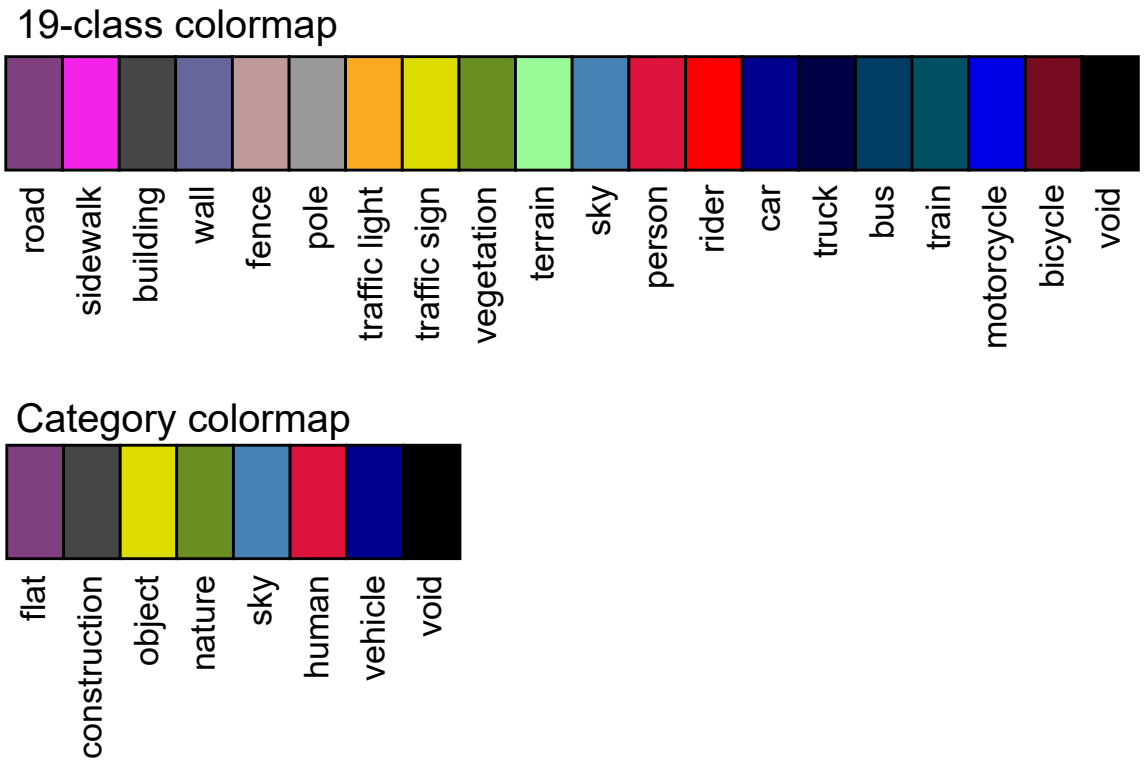


Figure 2.22 – **Urban scene segmentation colormaps.** Top: colormap for the 19-class semantic segmentation setting of Cityscapes; Bottom: colormap for the 7 high-level categories of classes for urban scene datasets. "void" refers to both unlabeled pixels and irrelevant or ignored classes.

2.3.2.2 GTA5

GTA5 (Richter et al. 2016) is a dataset of 24,966 synthetic images of size 1920×1080 rendered using the eponymous open-world video game. The urban scenes are all from the car perspective and mimic the streets of American cities. Every image is annotated with a pixel-wise semantic segmentation map with labels over 19 classes compatible with Cityscapes. This dataset is generally used as a source domain for UDA experiments. As a consequence, it only includes a fully-labeled train set and no validation or test set. Figure 2.23 shows example images from the GTA5 dataset. Table 2.2 describes the class labels and mappings of the dataset.

2.3.2.3 SYNTHIA

The SYNTHIA dataset (Ros et al. 2016) is a synthetic dataset of urban scenes that proposes a split designed for domain adaptation to Cityscapes, SYNTHIA-RAND-CITYSCAPES. It is composed of 9,400 synthetic images of size $1280 \times$

Class Name	Orig. Id	Std. Id	Category	Used?
unlabeled	0		void	
ego vehicle	1		void	
rectification border	2		void	
out of roi	3		void	
static	4		void	
dynamic	5		void	
ground	6		void	
road	7	0	flat	✓
sidewalk	8	1	flat	✓
parking	9		flat	
rail track	10		flat	
building	11	2	construction	✓
wall	12	3	construction	✓
fence	13	4	construction	✓
guard rail	14		construction	
bridge	15		construction	
tunnel	16		construction	
pole	17	5	object	✓
polegroup	18		object	
traffic light	19	6	object	✓
traffic sign	20	7	object	✓
vegetation	21	8	nature	✓
terrain	22	9	nature	✓
sky	23	10	sky	✓
person	24	11	human	✓
rider	25	12	human	✓
car	26	13	vehicle	✓
truck	27	14	vehicle	✓
bus	28	15	vehicle	✓
caravan	29		vehicle	
trailer	30		vehicle	
train	31	16	vehicle	✓
motorcycle	32	17	vehicle	✓
bicycle	33	18	vehicle	✓
license plate	-1		vehicle	

Table 2.1 – **Classes of Cityscapes.** First two columns: names and ids of original classes; Third column: ids of the classes standardly used in the 19-class semantic segmentation setting of Cityscapes. Not numbered classes are not used; Fourth column: high-level categories (or super classes) of the class (color-code as defined in Figure 2.22); Fifth column: ✓ indicates that the class is considered during training and test with super classes, the other classes are ignored.

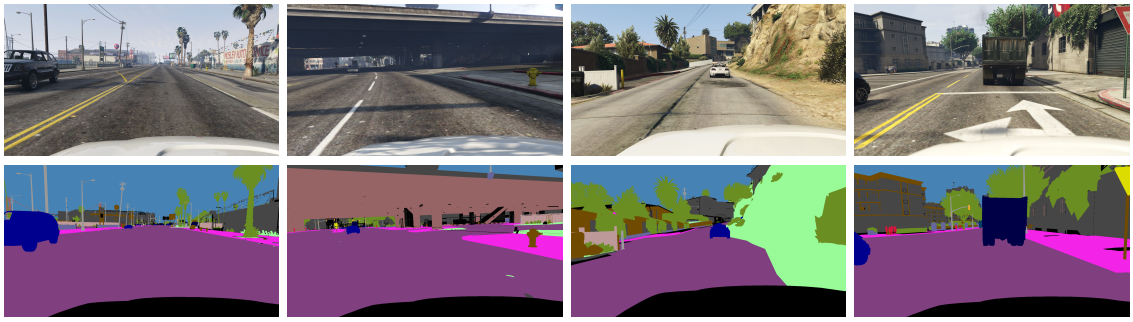


Figure 2.23 – **Example of images from the GTA5 dataset.** First row: Synthetic urban scene image; Second row: Associated 19-class semantic segmentation map provided by simulation (colormap defined in Figure 2.22).

Class Name	Id	Category	Used?
road	0	flat	✓
sidewalk	1	flat	✓
building	2	construction	✓
wall	3	construction	✓
fence	4	construction	✓
pole	5	object	✓
traffic light	6	object	✓
traffic sign	7	object	✓
vegetation	8	nature	✓
terrain	9	nature	✓
sky	10	sky	✓
person	11	human	✓
rider	12	human	✓
car	13	vehicle	✓
truck	14	vehicle	✓
bus	15	vehicle	✓
train	16	vehicle	✓
motorcycle	17	vehicle	✓
bicycle	18	vehicle	✓
unlabeled	-1	void	

Table 2.2 – **Classes of GTA5.** First two columns: names and ids of the classes. Note that these classes are common with Cityscapes; Third column: high-level categories (or super classes) of the class; Fourth column: ✓ indicates that the class is considered during training and test both with the standard 19-class setting and with super classes, the other classes are ignored.

760, fully annotated with pixel-wise semantic labels over 16 classes in common with Cityscapes as well as other ground-truth modalities, for instance, depth

information. Compared to GTA5 whose images are all from the car perspective, SYNTHIA features photo-realistic frames from multiple viewpoints. Usually, this dataset is only used for training purposes as a source domain. Thus, it doesn't include either validation or test sets. Figure 2.24 shows example images from the SYNTHIA dataset. Table 2.3 describes the class labels and mappings of the dataset.

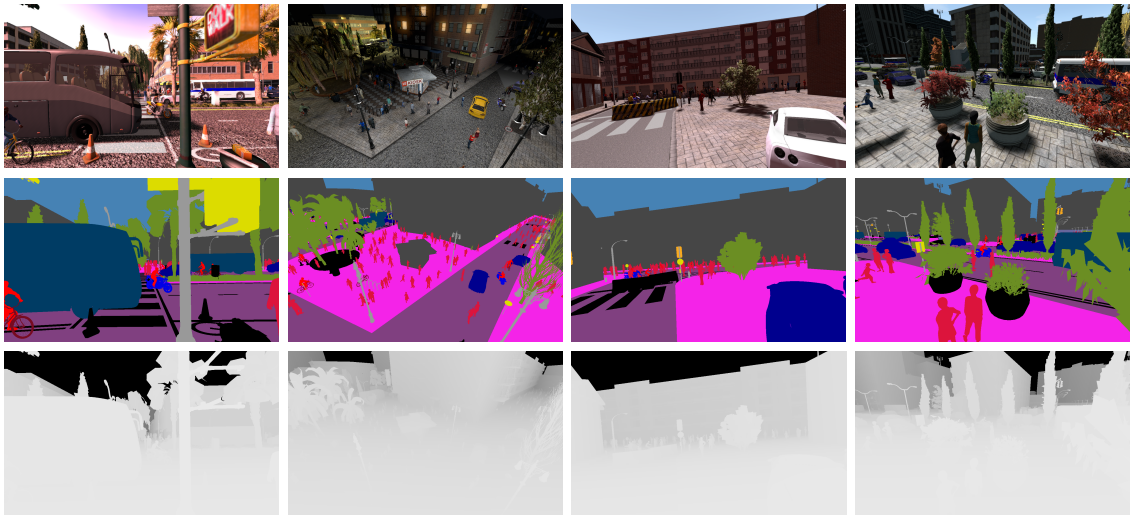


Figure 2.24 – **Example of images from the SYNTHIA dataset.** First row: Synthetic urban scene image; Second row: Associated 16-class semantic segmentation map provided by the simulation (colormap defined in Figure 2.22); Third row: Associated depth map provided by the simulation (brighter means closer).

2.3.2.4 Mapillary Vistas

Mapillary Vistas (Neuhold et al. 2017), Mapillary in short, is a dataset collected in multiple cities around the world, which is composed of 18,000 training and 2,000 validation labeled scenes of varying sizes. Figure 2.25 shows example images from the Mapillary dataset. Table 2.4 describes the class labels and mappings of the dataset.

2.3.2.5 IDD

IDD (Varma et al. 2019) is an Indian urban dataset having 6,993 training and 981 validation labeled scenes of size 1280×720. Figure 2.26 shows example images from the IDD dataset. Table 2.5 describes the class labels and mappings of the dataset.

Class Name	Orig. Id	16-c. Id	13-c. Id	Category	Used?
void	0			void	
sky	1	9	6	sky	✓
building	2	2	2	construction	✓
road	3	0	0	flat	✓
sidewalk	4	1	1	flat	✓
fence	5	4		construction	✓
vegetation	6	8	5	nature	✓
pole	7	5		object	✓
car	8	12	9	vehicle	✓
traffic sign	9	7	4	object	✓
pedestrian	10	10	7	human	✓
bicycle	11	15	12	vehicle	✓
motorcycle	12	14	11	vehicle	✓
parking-slot	13			flat	
road-work	14			flat	
traffic light	15	6	3	object	✓
terrain	16			nature	✓
rider	17	11	8	human	✓
truck	18			vehicle	✓
bus	19	13	10	vehicle	✓
train	20			vehicle	✓
wall	21	3		construction	✓
lanemarking	22			flat	

Table 2.3 – **Classes of SYNTHIA.** The reported classes correspond to the SYNTHIA-RAND-CITYSCAPES split. First two columns: names and ids of the classes; Third and fourth columns: ids of the classes standardly used in the 16-class and 13-class semantic segmentation setting of SYNTHIA. Not numbered classes are not used; Fifth column: high-level categories (or super classes) of the class; Sixth column: ✓ indicates that the class is considered during training and test with super classes, the other classes are ignored.

2.3.2.6 Evaluation

Semantic segmentation models are generally evaluated in terms of Intersection over Union (IoU), also known as Jaccard index, per class or mean Intersection over Union (mIoU) over all classes, expressed in percentage. The IoU is a distance between two sets A and B defined as the ratio between the intersection of the sets and their union:

$$\text{IoU}(A, B) = \frac{|A \cap B|}{|A \cup B|}. \tag{2.17}$$

When evaluating the performance of a semantic segmentation model on a specific class, the IoU metric is computed between the ground-truth and the predicted set

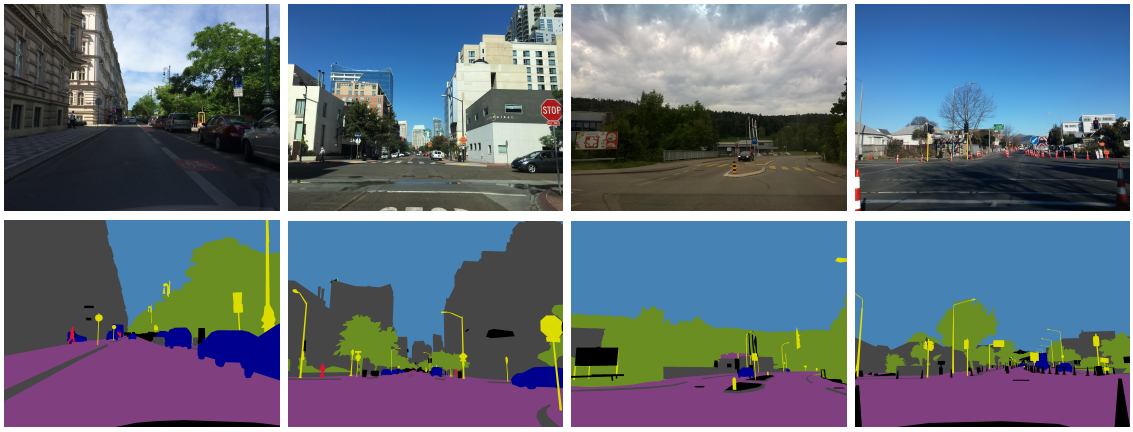


Figure 2.25 – **Example of images from the Mapillary dataset.** First row: Real urban scene image; Second row: Associated high-level category semantic segmentation map (colormap defined in Figure 2.22).

Figure 2.26 – **Example of images from the IDD dataset.** First row: Real indian urban scene image; Second row: Associated high-level category semantic segmentation map (colormap defined in Figure 2.22).

of pixels of this class. The higher this percentage, the better. The mIoU is computed as the average of the IoU of all the considered classes, sometimes excluding some irrelevant or problematic classes.

Since the goal of UDA is to train a model to perform on the target domain, the performance of UDA models are only evaluated on the target dataset. Moreover, since the source and target domain must be associated with the same task, the semantic labels of source and target must also be shared. Thus, "mIoU-C" may refer to the mIoU over the C classes shared between the source and target datasets.

2.3.3 Positioning

The literature on UDA is rich, with a variety of models and strategies to tackle the problematics of the task, especially in the complex semantic segmentation

Class Name	Id	Category	Used?
bird	0	other	
ground animal	1	other	
curb	2	construction	✓
fence	3	construction	✓
guard rail	4	construction	✓
barrier	5	construction	✓
wall	6	construction	✓
bike lane	7	flat	✓
crosswalk - plain	8	flat	✓
curb cut	9	flat	✓
parking	10	flat	✓
pedestrian area	11	flat	✓
rail track	12	flat	✓
road	13	flat	✓
service lane	14	flat	✓
sidewalk	15	flat	✓
bridge	16	construction	✓
building	17	construction	✓
tunnel	18	construction	✓
person	19	human	✓
bicyclist	20	human	✓
motorcyclist	21	human	✓
other rider	22	human	✓
lane marking - crosswalk	23	flat	✓
lane marking - general	24	flat	✓
mountain	25	nature	
sand	26	nature	
sky	27	sky	✓
snow	28	nature	
terrain	29	flat	✓
vegetation	30	nature	✓
water	31	nature	
banner	32	object	

Class Name	Id	Category	Used?
bench	33	object	
bike rack	34	object	
billboard	35	object	
catch basin	36	object	
cctv camera	37	object	
fire hydrant	38	object	
junction box	39	object	
mailbox	40	object	
manhole	41	object	
phone booth	42	object	
pothole	43	object	✓
street light	44	object	✓
pole	45	object	✓
traffic sign frame	46	object	✓
utility pole	47	object	✓
traffic light	48	object	✓
traffic sign (back)	49	object	✓
traffic sign (front)	50	object	✓
trash can	51	object	
bicycle	52	vehicle	✓
boat	53	vehicle	✓
bus	54	vehicle	✓
car	55	vehicle	✓
caravan	56	vehicle	✓
motorcycle	57	vehicle	✓
on rails	58	vehicle	✓
other vehicle	59	vehicle	✓
trailer	60	vehicle	✓
truck	61	vehicle	✓
wheeled slow	62	vehicle	✓
car mount	63	void	
ego vehicle	64	void	
unlabeled	-1	void	

Table 2.4 – **Classes of Mapillary Vistas.** Due to its size, the table is split in two. The organization of both is the same. First two columns: names and ids of the classes; Third column: high-level categories (or super classes) of the class; Fourth column: ✓ indicates that the class is considered during training and test with super classes, the other classes are ignored.

problem. However, the gap in performance between fully supervised learning and UDA is still large, leading to hopes of finding new better performing UDA techniques to bridge this gap. Moreover, the classical UDA scenario is very rigid: its scope is limited to the adaptation of a single source domain to a single target domain, sharing a single task, on which the source data is fully annotated while the target data is completely unannotated. Direct distribution alignment approaches, image-level approaches and adversarial approaches of the literature constitute a powerful toolbox for closing the gap between two domains based on a given task. However, they can neither directly leverage additional data from

Class Name	Id	Category	Used?
road	0	flat	✓
parking	1	flat	
drivable fallback	2	flat	✓
sidewalk	3	flat	✓
rail track	4	flat	
non-drivable fallback	5	flat	
person	6	human	✓
animal	7	other	
rider	8	human	✓
motorcycle	9	vehicle	✓
bicycle	10	vehicle	✓
autorickshaw	11	vehicle	✓
car	12	vehicle	✓
truck	13	vehicle	✓
bus	14	vehicle	✓
caravan	15	vehicle	✓
trailer	16	vehicle	✓
train	17	vehicle	✓
vehicle fallback	18	vehicle	✓
curb	19	construction	✓

Class Name	Id	Category	Used?
wall	20	construction	✓
fence	21	construction	✓
guard rail	22	construction	
billboard	23	object	✓
traffic sign	24	object	✓
traffic light	25	object	✓
pole	26	object	✓
polegroup	27	object	
obs-str-bar-fallback	28	object	
building	29	construction	✓
bridge	30	construction	
tunnel	31	construction	
vegetation	32	nature	✓
sky	33	sky	✓
fallback background	34	object	
unlabeled	35	void	
ego vehicle	36	void	
rectification border	37	void	
out of roi	38	void	
license plate	39	vehicle	

Table 2.5 – **Classes of IDD.** Due to its size, the table is split in two. The organization of both is the same. First two columns: names and ids of the classes; Third column: high-level categories (or super classes) of the class; Fourth column:✓ indicates that the class is considered during training and test with super classes, the other classes are ignored.

the source or target domain, nor be trivially extended to more than two domains. Based on the UDA approaches of this literature, my Ph.D. work aims at broadening the classical UDA scenario to more practical cases depending on its applications, especially in an autonomous driving context.

For instance, even though the main objective is to train a semantic segmentation model for the target domain using unlabeled target data, the source data used by the UDA model does not need to be restricted to pairs of images and semantic segmentation maps from a practical standpoint. Indeed, especially when synthetic source data is considered, more modalities could be acquired and included in the training process as Privileged Information (PI): instance segmentation, 2D bounding boxes, or even depth information, also more and more often sparsely available in real datasets thanks to joint scanning with LiDAR. Chapter 3 studies how to effectively use additional knowledge in the source domain as PI, especially depth information, and presents, in particular, the architecture Bilinear Multimodal Domain Adaptation (BerMuDA) we propose to improve adversarial UDA approaches in the presence of PI.

Moreover, some parallels can be made between UDA and semi-supervised learning: indeed, while in UDA two different domains are considered with the source labeled and the target unlabeled, semi-supervised learning aims at training mod-

els using both few labeled data and a larger unlabeled dataset. Analyzing the UDA problem from a semi-supervised learning point of view may shed new light on ways to better leverage the unlabeled target data. In particular, Self-Training (ST) is a semi-supervised learning method that uses a pre-trained classifier on the unlabeled data to generate more labeled samples, which can be used by a supervised learning method. Generally, in this kind of approach, only the most confident predictions are used as *pseudo-labels*. Chapter 4 studies how to improve UDA methods by using this semi-supervised learning ST strategy. In particular, it discusses how to properly estimate the confidence of the predictions used to generate pseudo-labels and introduces two novel methods to this end: Entropy-based Self-supervised Learning (ESL), which exploits the entropy as a measure of confidence, and Confidence Learning for Domain Adaptation (ConDA), which learns the confidence with an auxiliary NN.

Finally, while the standard UDA scenario is useful for autonomous driving applications, especially in the synthetic-to-real context, it is still very limited when considering practical use-cases. Indeed, autonomous vehicles may encounter a large variety of urban scene scenarios in the wild, such as varying weather conditions, lighting conditions, or different cities, each of those representing a specific domain. While standard UDA allows us to train a model on a particular unlabeled target domain, it may not generalize to this wide variety of scenarios. Thus, Chapter 5 extends the single source-single target UDA setting to UDA to multiple target domains. More specifically, we introduce two adversarial frameworks for multi-target UDA, Multi-Discriminator (Multi-Dis.) and Multi-Target Knowledge Transfer (MTKT), as well as an adversarial method for continual learning UDA, Continual Target Knowledge Transfer (CTKT).

CHAPTER 3

LEVERAGING PRIVILEGED INFORMATION FOR UNSUPERVISED DOMAIN ADAPTATION

Chapter abstract

One advantage of synthetic imagery is the ease with which additional ground-truth information may be obtained, e.g. other modalities, which can be considered as Privileged Information (PI). This chapter studies the scenario of Unsupervised Domain Adaptation (UDA) from synthetic images for semantic segmentation with depth as PI, in the context of autonomous driving. It introduces our contribution, Bilinear Multimodal Domain Adaptation (BerMuDA), which exploits this PI by adversarial training with a multimodal discriminator. More specifically, BerMuDA proposes to efficiently learn a bilinear fusion between the two modalities to optimize the ability of the discriminator to distinguish between the source and target domains.

The work in this chapter has led to the publication of a conference paper:

- Taylor Mordan, Antoine Saporta, Alexandre Alahi, Matthieu Cord, and Patrick Pérez (2020). "Bilinear Multimodal Discriminator for Adversarial Domain Adaptation with Privileged Information". In: *Symposium of the European Association for Research in Transportation (hEART)*.

Contents

The Unsupervised Domain Adaptation (UDA) methods for semantic segmentation presented in Chapter 2, such as (Tsai et al. 2018) or (Vu et al. 2019a), proved to be particularly effective in synthetic-to-real scenarios. Resorting to synthetic imagery to produce the labeled source dataset while being able to adapt to the real target domain has many advantages. In addition to being substantially cheaper to annotate than real images, the synthetic image generator is significantly less prone to labeling errors compared to human-annotated data. Furthermore, synthetic imagery has the freedom to generate urban scenes from virtually any conceivable situation, including unusually-rare scenarios in real life. These properties of synthetically generated datasets can help prediction models trained with Domain Adaptation (DA) to be more robust and better adjust to the unpredictability of urban environments.

Additionally, synthetic urban scene generators may create ground-truth annotations for other modalities than semantic segmentation maps, for instance, depth maps, without great additional costs. This extra data could be used as Privileged Information (PI) (Vapnik and Izmailov 2015) to further improve the UDA training for semantic segmentation. To leverage this PI, the UDA strategies need to be adapted both in terms of architectures and in terms of learning.

This chapter first extends the literature review of Chapter 2 by discussing UDA approaches revolving on multiple modalities, both in Multi-Task Learning (MTL) settings and PI settings. Then, it presents our contribution Bilinear Multimodal Domain Adaptation (BerMuDA) which proposes a new way to adapt the discriminator of adversarial UDA strategies when dealing with PI such as depth.

3.1 Domain Adaptation with Multiple Modalities

This section reviews UDA approaches dealing with multiple modalities in the literature, first developing UDA in an MTL context, then in a PI scenario. It finally dicusses how to more generally leverage multiple modalities in Deep Neural Network (DNN) architectures.

3.1.1 Multi-Task Unsupervised Domain Adaptation

The methods discussed in Chapter 2 consider UDA in the traditional context where both source and target domains deal with a single task. Nonetheless, if data for other tasks were to be accessible for the source domain, one may extend the standard UDA scenario to an MTL context (Caruana 1997). By training the UDA model on multiple tasks simultaneously with shared representations, the model should be able to generalize better on each of the tasks on the one hand and, more importantly, on the target domain on the other hand.

Cross-Domain Multi-Task Feature Learning. Ren and Y. J. Lee (2018) propose to apply an adversarial domain adaptation strategy (see Section 2.2.3) in an MTL setting. More specifically, their approach considers a single base network, a bottleneck consisting of dilated convolution layers and multiple task heads, one for edge detection, one for surface normal estimation, and one for depth prediction. A discriminator, placed at the end of the base network, ensures that the learned features are domain invariant. Figure 3.1 illustrates their architecture.

Cross-Task Distillation. Kundu et al. (2019) build an architecture relying on a good balance between the multiple tasks to ensure an easy adaptation to the target domain. Figure 3.2 illustrates their architecture. The base model is first trained on the source domain along with a cross-task distillation module (blue block on Figure 3.2): this module aims at predicting representation of each task from the combined representations of all the other tasks. This module effectively balances the importance given to each task during the training since the least performing task will exhibit higher discrepancy between its direct prediction and indirect prediction through cross-task distillation, thus giving it more importance in the next iterations. In the second training phase, the base model is adapted to the target domain. The whole model is frozen except for the single last residual block (in yellow on Figure 3.2), which is trained by minimizing the cross-task distillation module discrepancy losses. While the source-target alignment is performed with an adversarial approach in (Ren and Y. J. Lee 2018), Kundu et al. (2019) leverage

the MTL framework for the UDA by learning on the source domain to distill knowledge from each task to the other tasks.

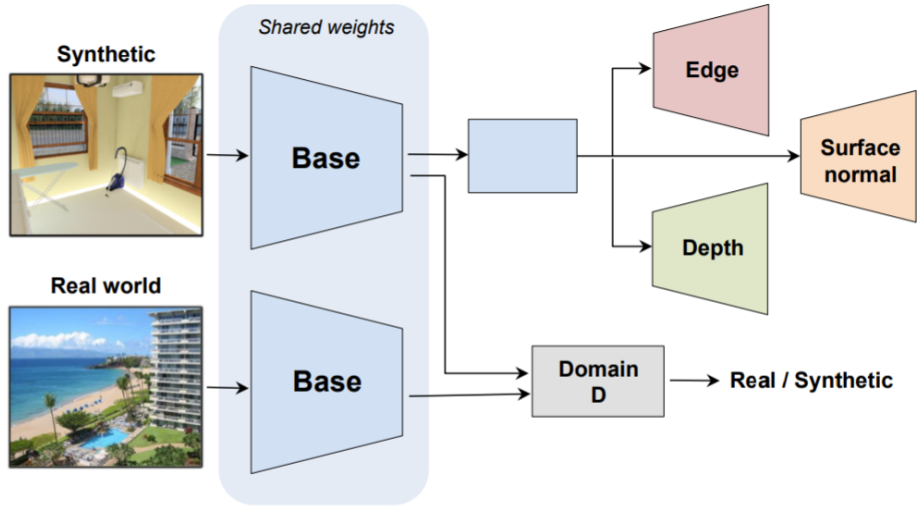


Figure 3.1 – **Cross-Domain Multi-Task Feature Learning Architecture.**. While the upper net predicts the depth, surface normal and edges of synthetic images (source domain) in a supervised fashion, a domain discriminator D tries to differentiate between real (target) and synthetic (source) features. Illustration from (Ren and Y. J. Lee 2018).

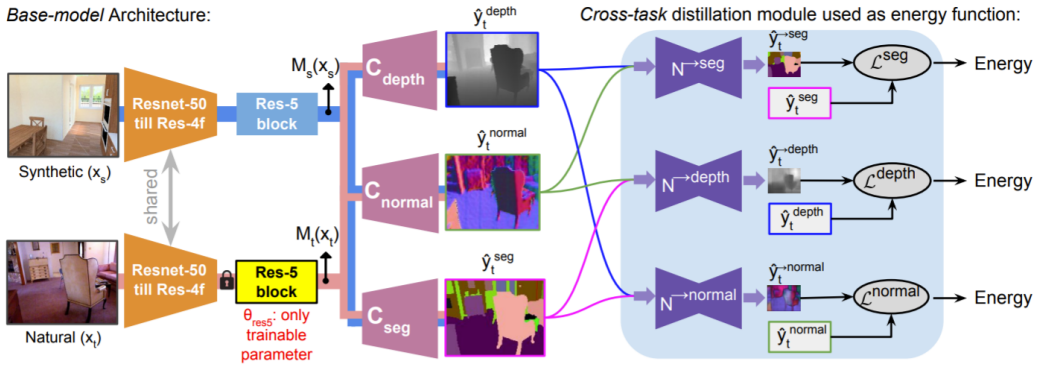


Figure 3.2 – **Cross-Task Distillation Architecture.**. The model is pre-trained on source along with a cross-task distillation module (blue block). It is adapted in a second phase on the target domain by finetuning only the last residual block (in yellow); this is achieved by minimizing the cross-task distillation module discreapancy losses to transfer knowledge from each task to the other tasks. Illustration from (Kundu et al. 2019).

3.1.2 Privileged Information in Unsupervised Domain Adaptation

Nonetheless, the additional tasks on the source domain could be instrumental and a way to regularize and learn better features, while the objective of the UDA model is still to perform a single task on the target domain. In the context of PI, the scenario where the model is trained in an MTL on the source domain while only the main task is evaluated on the target domain is more generally considered.

SPIGAN. K.-H. Lee et al. (2019) propose Simulator Privileged Information and Generative Adversarial Network (SPIGAN), illustrated in Figure 3.3. Their architecture is inspired by image-to-image translation-based UDA approaches such as (Bousmalis et al. 2017) (see Section 2.2.2.1). A Generative Adversarial Network (GAN) G learns to transform source images into the style of the target domain. A discriminator D is trained to distinguish between fake generated images and real target domain images. Both the semantic segmentation task T and a depth prediction model P, based on the depth privileged information $\boldsymbol{z}_s$, are trained on source data using the original and generated images while directly transferring the annotations from the former to the latter.

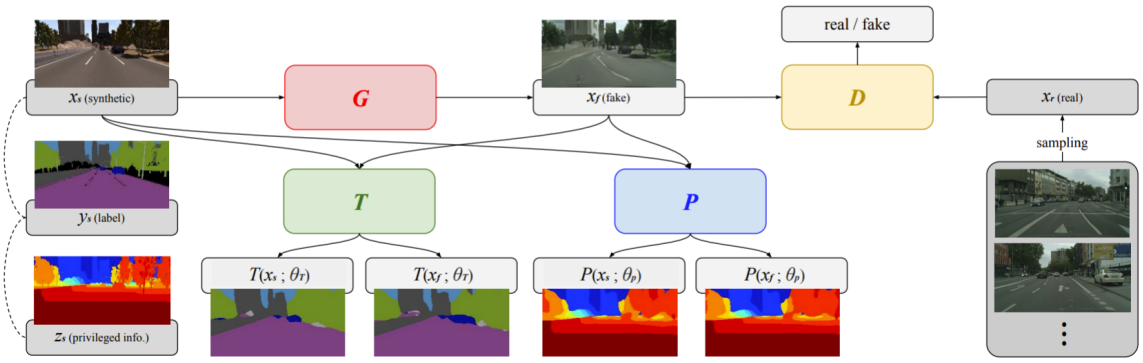


Figure 3.3 – **Simulator Privileged Information and Generative Adversarial Network Architecture**. SPIGAN learns a GAN G to translate source synthetic image in the target real style by adversarial training with a discriminator D. The model uses both original and translated source images to train a semantic segmentation network T and a depth prediction network P by exploiting the ground-truth predictions both for semantic segmentation $\boldsymbol{y}_s$ and for depth $\boldsymbol{z}_s$ as PI. Though it is not used at test time, training the depth prediction network P serves as PI regularization to the generator G. Illustration from (K.-H. Lee et al. 2019).

DADA. Vu et al. (2019b) take some inspiration from the primary MTL context of (Mordan et al. 2018) and Adversarial Entropy Minimization (AdvEnt) (Vu et al. 2019a) (see Section 2.2.3.2) to build Depth-Aware Domain Adaptation (DADA). First, the semantic segmentation model is augmented by an auxiliary depth prediction module inspired by (Mordan et al. 2018). From the extracted features of the model, an encoder computes features for depth prediction, simply computed with an average pooling – the depth is entirely encoded in these features since a simple pooling is required to extract the predictions. These auxiliary features are then merged into the segmentation model with a decoder plus a pointwise feature fusion. Moreover, the adversarial discrimination of DADA is applied on the joint product of the depth output and the weighted self-information of the output. The DADA architecture is illustrated in Figure 3.4.

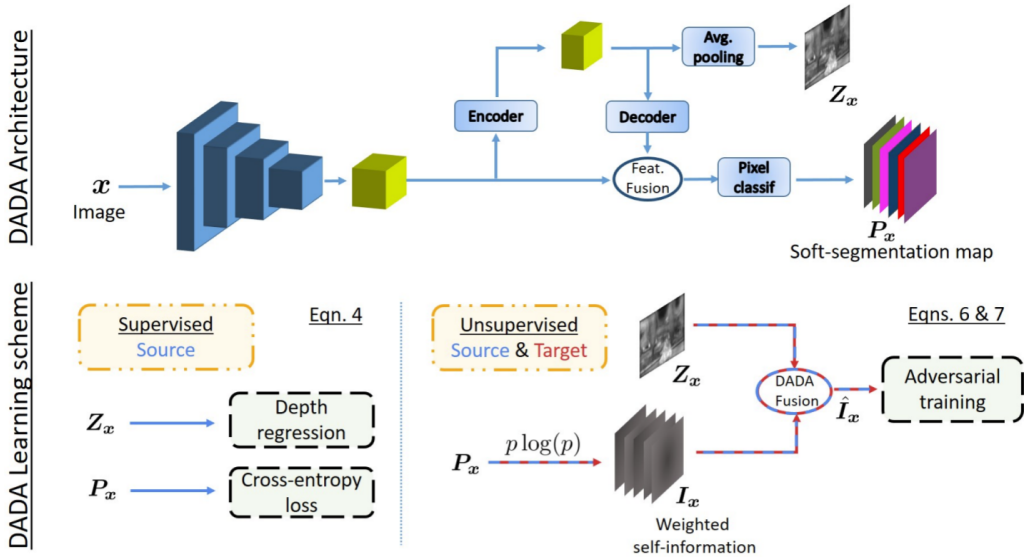


Figure 3.4 – **Depth-Aware Domain Adaptation Architecture**. DADA uses an auxiliary encoder from the features of the segmentation model to perform depth prediction and injects these new features into the segmentation model with a decoder and feature fusion. Furthermore, adversarial discrimination is performed on the fusion of predicted depth and weighted self-information. Illustration from (Vu et al. 2019b).

3.1.3 Dealing with Multiple Modalities

Both (K.-H. Lee et al. 2019) and (Vu et al. 2019b) exploit the depth PI by learning an additional depth prediction module. SPIGAN (K.-H. Lee et al. 2019) completely separates the two task modules – only the image generator is "shared". Following a similar strategy in an output-space adversarial approach, one could consider

training a different discriminator for each modality. On the other hand, DADA (Vu et al. 2019b) explicitly leverages both modalities in its single model with both a feature fusion and a DADA fusion (joint product) between depth map and weighted self-information to perform adversarial training on this merged representation. Merging the multiple modalities into a single representation in the context of UDA with PI allows leveraging the PI while performing the alignment between source and target on this single representation. This section studies *bilinear fusions* which are a class of functions that can fuse multiple modalities into a single representation.

Visual Question Answering and Block. Bilinear fusions have been widely studied for the Visual Question Answering task, where such models have been used with text and image modalities (Fukui et al. 2016; J.-H. Kim et al. 2016; Ben-Younes et al. 2017; Ben-Younes et al. 2019).

Ben-Younes et al. (2017) propose a general tensor-based fusion, taking as inputs the two modality representations – text and image – and, through the tensor bilinear product, outputting a fused representation. Practically, their method MUTAN reduces the number of learnable parameters required for the tensor fusion by performing a Tucker decomposition: the 3-way tensor $\mathcal{T} \in \mathbb{R}^{d_q \times d_v \times d_o}$ is expressed as the product between factor matrices $\boldsymbol{W}_q, \boldsymbol{W}_v, \boldsymbol{W}_o$, and a core tensor $\mathcal{T}_c$:

$$\mathcal{T} = ((\mathcal{T}_c \times_1 \boldsymbol{W}_q) \times_2 \boldsymbol{W}_v) \times_3 \boldsymbol{W}_o, \tag{3.1}$$

where $\boldsymbol{W}_q \in \mathbb{R}^{d_q \times t_q}$, $\boldsymbol{W}_v \in \mathbb{R}^{d_v \times t_v}$, $\boldsymbol{W}_o \in \mathbb{R}^{d_o \times t_o}$, $\mathcal{T}_c \in \mathbb{R}^{t_q \times t_v \times t_o}$, and $\times_i$ is the $i-mode$ product between a tensor and a matrix. The MUTAN fusion is illustrated in Figure 3.5.

Ben-Younes et al. (2019) extend this approach with BLOCK: the core tensor is now expressed as a block-diagonal tensor. MUTAN can be seen as a particular case of BLOCK where there is only a single block in the core tensor.

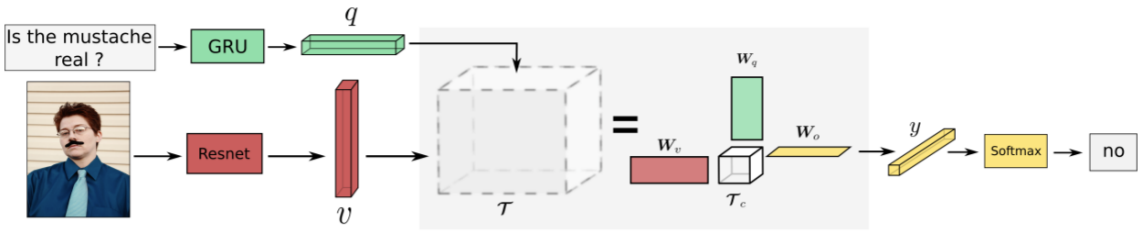
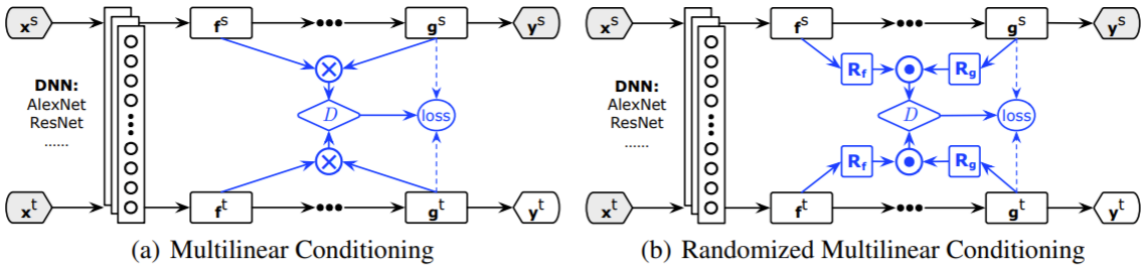


Figure 3.5 – **MUTAN Fusion Architecture**. The multiple modalities are merged into a single representation by the bilinear product associated to the tensor $\mathcal{T}$. The Tucker decomposition $\{\mathcal{T}_c, \boldsymbol{W}_q, \boldsymbol{W}_v, \boldsymbol{W}_o\}$ allows a parameter-efficient fusion. Illustration from (Ben-Younes et al. 2017).

Bilinear Fusion in Domain Adaptation. Bilinear fusions have also been explored in UDA contexts without PI. Long et al. (2018) propose a conditional adversarial alignment approach that relies on the multimodal information of the feature representation and the classifier prediction obtained by multilinear fusion, either the tensor product of those two modalities or a randomized version for higher-dimensional scenarios. Figure 3.6 illustrates their method.



(a) Multilinear Conditioning (b) Randomized Multilinear Conditioning

Figure 3.6 – **Conditional Domain Adversarial Network Architecture**. (a) In lower-dimensional scenario, the discriminator is applied on the multilinear map between feature representation and classifier prediction. (b) In higher-dimensional scenario, a randomized multilinear map is considered where a few components of the feature representation and the classifier prediction are selected at random. Illustration from (Long et al. 2018).

On a very different setting, Qi et al. (2018) use bilinear fusions to align audio-visual multimodal distributions, without PI.

Finally, in UDA with PI and as mentioned earlier, DADA (see Figure 3.1.2) also uses bilinear fusion as the joint product of depth and weighted self-information.

3.2 BerMuDA : Bilinear Multimodal Discriminator for Adversarial Domain Adaptation with Privileged Information

SPIGAN (K.-H. Lee et al. 2019) and DADA (Vu et al. 2019b) are two approaches tackling UDA from synthetic images and using depth as privileged information. The first one uses depth as an additional way to regularize the training of the generator that translates images from the source domain to the target one. PI is therefore not directly used to enhance the segmentation network, which should yield less transfer between tasks. On the other hand, DADA, among other ideas, employs a simple element-wise product to fuse semantic segmentation and depth modalities, thus focusing on close objects.

Our approach BerMuDA, presented in this section, opts for bilinear transformations based on BLOCK (Ben-Younes et al. 2019) which allow performing more complex fusions between the two modalities, leading to more discriminative fused features.

3.2.1 Model

BerMuDA addresses UDA with depth as PI on the source domain. The model is built on standard output-space adversarial training for UDA (see Section 2.2.3.2), and adapts both the segmentation and discriminator networks to accommodate to the additional modality, as illustrated in Figure 3.7. In particular, alignment between both modalities is performed with a bilinear fusion integrated within the discriminator.

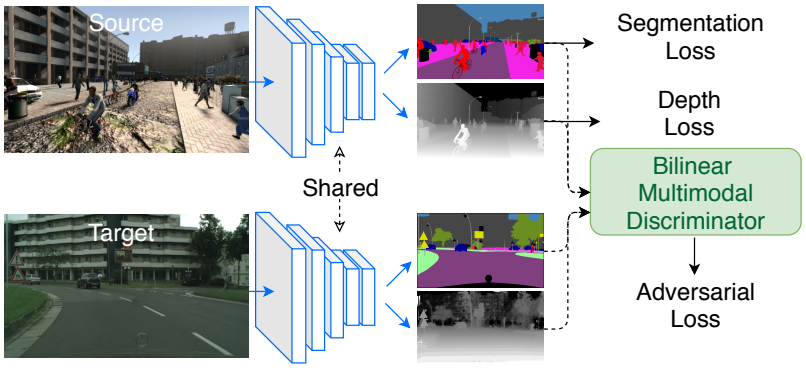


Figure 3.7 – **BerMuDA's architecture.** A bilinear multimodal discriminator aligns segmentation and depth predictions into a common representation of both modalities used for adversarial training.

3.2.1.1 Overview of the approach

Let us consider two image domains: the synthetic source $\mathfrak{X}_s$, with dataset $\mathcal{X}_s$, and the real target $\mathfrak{X}_t$, with dataset $\mathcal{X}_t$. A source example $\left(\boldsymbol{x}_s, \boldsymbol{y}_s^{\text{seg}}, \boldsymbol{y}_s^{\text{dep}}\right)$ is composed of a synthetic image $\boldsymbol{x}_s \in \mathcal{X}_s$ with the corresponding ground truth segmentation map $\boldsymbol{y}_s^{\text{seg}}$ and depth map $\boldsymbol{y}_s^{\text{dep}}$, represented as spatial arrays of one-hot vectors of C classes and of scalar distance values respectively. On the other hand, a target domain example only contains an unlabeled real image $\boldsymbol{x}_t \in \mathcal{X}_t$. In this context of learning with PI, the model is a main network F taking either a source or target image $\boldsymbol{x}$ as input and yielding both a segmentation $\boldsymbol{P}_{\boldsymbol{x}}^{\text{seg}}$ and a depth map $\boldsymbol{P}_{\boldsymbol{x}}^{\text{dep}}$ as output, denoted as $\left(\boldsymbol{P}_{\boldsymbol{x}}^{\text{seg}}, \boldsymbol{P}_{\boldsymbol{x}}^{\text{dep}}\right) = F(\boldsymbol{x})$ and illustrated in Figure 3.7. The model consists of a backbone network with two task-specific prediction branches

for segmentation and depth prediction that are learned jointly, as is common practice in MTL problems (Girshick 2015; Kokkinos 2017).

Supervised training on source domain. Since annotations are available on the source domain, the model is learned in the standard supervised MTL way for source examples, with the supervised loss $\mathcal{L}_{F,\text{sup}}$ being a linear combination of all task losses:

$$\mathcal{L}_{F,\text{sup}} = \mathcal{L}_{F,\text{seg}} + \lambda_{F,\text{depth}} \mathcal{L}_{\text{depth}}, \tag{3.2}$$

where λ_{depth} is a hyper-parameter balancing both task losses. The segmentation loss $\mathcal{L}_{F,\text{seg}}$ is the usual pixel-wise cross-entropy loss (Equation 2.5). The depth prediction loss $\mathcal{L}_{F,\text{depth}}$ is a reverse Huber regression loss (Laina et al. 2016) applied on depth predictions $\boldsymbol{P}^{\text{dep}}$ in log space, to focus more on closer objects, as in (Mordan et al. 2018):

$$\mathcal{L}_{F,\text{depth}}\left(\boldsymbol{P}^{\text{dep}}_{\boldsymbol{x}_s}, \boldsymbol{y}^{\text{dep}}_s\right) = \sum_{h=1}^{H}\sum_{w=1}^{W} \text{berHu}\left(\boldsymbol{P}^{\text{dep}}_{\boldsymbol{x}_s}{}_{[h,w,\cdot]} - \boldsymbol{y}^{\text{dep}}_s{}_{[h,w,\cdot]}\right), \tag{3.3}$$

where $H \times W$ are the dimensions of the input image $\boldsymbol{x}_s$ (and thus are also the dimensions of its ground-truth depth map $\boldsymbol{y}^{\text{dep}}_s$) and with the reverse Huber function defined by

$$\text{berHu}\left(e\right) = \begin{cases} |e| & \text{if } |e| \leq \tau, \\ \frac{e^2+\tau^2}{2\tau} & \text{if } |e| > \tau, \end{cases} \tag{3.4}$$

τ being a threshold set to $1/5$ of the maximum error in the mini-batch.

Adversarial training on target domain. Target domain examples are used for learning through an adversarial training procedure as no annotation is available on this domain. For this, a discriminator D is learned concurrently to the main network F, and they compete against each other for optimizing exclusive objectives. Following output-space adversarial alignment strategies (Section 2.2.3.2), for an input image $\boldsymbol{x}$ in either domain, the discriminator D takes the output $F(\boldsymbol{x})$ of network F as input, and makes a prediction $D(F(\boldsymbol{x}))$ for the domain of $\boldsymbol{x}$. Let us introduce a new kind of discriminator network D, called Bilinear Multimodal Discriminator, to handle the multiple modalities output by the network F, as showcased in Figure 3.7. With the architecture of this discriminator D further described in Section 3.2.1.2, the training procedure is similar to standard output-space adversarial alignment with, on one hand, the minimization of the binary cross-entropy loss $\mathcal{L}_{\text{D}}$ to learn D (Equation 2.15) and, on the other hand, the adversarial binary cross-entropy loss $\mathcal{L}_{F,\text{adv}}$ on the target domain examples to train F in order to fool the discriminator D (Equation 2.16).

Full training on both domains and deployment. The full loss function $\mathcal{L}_F$ minimized by the network F is then the weighted sum of the supervised loss (Equation 3.2) and the adversarial loss (Equation 2.16):

$$\mathcal{L}_F = \mathcal{L}_{F,\text{seg}} + \lambda_{\text{depth}}\mathcal{L}_{F,\text{depth}} + \lambda_{\text{adv}}\mathcal{L}_{F,\text{adv}}, \tag{3.5}$$

with λ_{adv} a hyper-parameter controlling the weight of the adversarial loss. The training therefore alternates between optimizing the network F (Equation 3.5) and the discriminator D (Equation 2.15).

When deployed, our model does not induce any overhead compared to a standard segmentation network, as the depth branch is not used. The associated auxiliary task is leveraged during training only, under the primary MTL framework (Mordan et al. 2018).

3.2.1.2 Bilinear Multimodal Discriminator

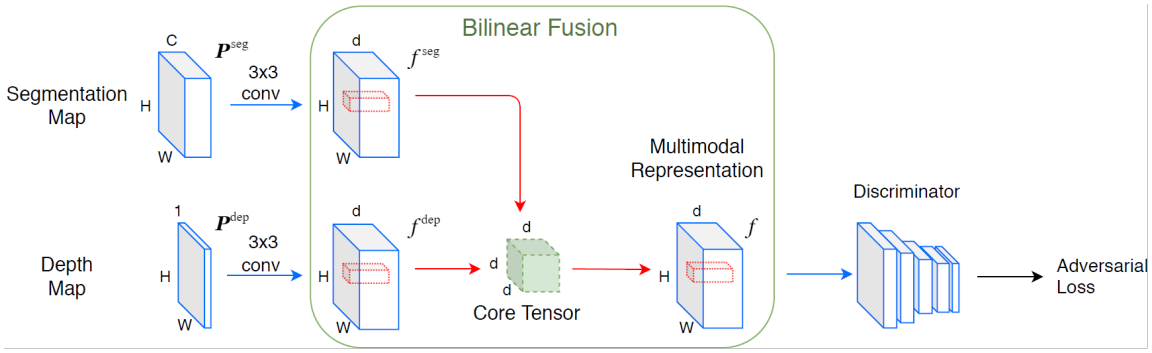


Figure 3.8 – **Bilinear Multimodal Discriminator.** The discriminator is composed of two parts: a bilinear fusion between segmentation and depth representations, yielding a multimodal representation of both inputs, and a fully convolutional classifier to predict the domain of the input image.

The discriminator D, detailed in Figure 3.8, is composed of a bilinear fusion and a fully convolutional classifier. The last part is commonly used alone in adversarial training, and we here add a prior merging layer to handle multimodal inputs.

This additional step takes both segmentation $\boldsymbol{P}^{\text{seg}}$ and depth $\boldsymbol{P}^{\text{dep}}$ predictions from F as inputs, and projects them onto $\boldsymbol{f}^{\text{seg}}$ and $\boldsymbol{f}^{\text{dep}}$, in a feature space of higher dimension d through two separate 3×3 spatial convolutions. Using convolutions instead of pointwise mappings (Ben-Younes et al. 2017; Ben-Younes et al. 2019) is a way to generalize to spatial inputs, and therefore to locally aggregate spatial context, which is especially useful to handle depth prediction $\boldsymbol{P}^{\text{dep}}$ composed of a single channel. The block-diagonal fusion of (Ben-Younes et al. 2019), with K full-

rank blocks, lets us perform the actual alignment, represented by the green core tensor in Figure 3.8 (note that the K-block decomposition is not shown for clarity). This decomposition allows modelling fine interactions while still controlling the number of parameters in this core tensor. At each spatial position (h, w), ${\boldsymbol{f}^{\text{seg}}}_{[h,w,\cdot]}$ and ${\boldsymbol{f}^{\text{dep}}}_{[h,w,\cdot]}$ are fused into $\boldsymbol{f}_{[h,w,\cdot]}$ through the action of the core tensor, shown with red arrows in Figure 3.8. For this, they are uniformly divided into K chunks ${\tilde{\boldsymbol{f}}_k^{\text{seg}}}_{[h,w,\cdot]}$ and ${\tilde{\boldsymbol{f}}_k^{\text{dep}}}_{[h,w,\cdot]}$. Each pair of features is then bilinearly fused to yield a multimodal feature $\tilde{\boldsymbol{f}}_{k[h,w,\cdot]}$:

$$\tilde{\boldsymbol{f}}_{k[h,w,\cdot]} = \left({\tilde{\boldsymbol{f}}_k^{\text{seg}}}_{[h,w,\cdot]}\right)^\top \boldsymbol{A}_k \left({\tilde{\boldsymbol{f}}_k^{\text{dep}}}_{[h,w,\cdot]}\right) + \boldsymbol{b}_k, \tag{3.6}$$

with $\boldsymbol{A}_k$ and $\boldsymbol{b}_k$ the weight matrix and bias learned for chunk k in the core tensor. The final multimodal representation $\boldsymbol{f}_{[h,w,\cdot]}$ at spatial position (h, w) is the concatenation of all K $\tilde{\boldsymbol{f}}_{k[h,w,\cdot]}$. The complete multimodal map $\boldsymbol{f}$ is then fed into the fully convolutional classifier to output the prediction δ of the domain of input image.

3.2.2 Experimental Results

3.2.2.1 Datasets

As source domain, we use the SYNTHIA dataset (Ros et al. 2016). As detailed in Section 2.3.2, this dataset is composed of synthetic images annotated with pixel-wise semantic labels and has the particularity of also including depth map annotations which may be used as PI in our UDA context. For the target domain, we use the Cityscapes dataset (Cordts et al. 2016a), without any annotation. The models are trained on the union of SYNTHIA and Cityscapes training sets with the 16 common classes between them, and are evaluated on the 16-class and 13-class (excluding *wall*, *fence* and *pole* classes) subsets with the mean Intersection over Union (mIoU) metric, reported as the 'mIoU-16' and 'mIoU-13' metrics respectively, of Cityscapes validation set. Section 2.3.2 and more specifically Table 2.1 and Table 2.3 give more details on the datasets and their classes.

3.2.2.2 Implementation details

As it is common practice, we adopt DeepLab-V2 (L.-C. Chen et al. 2018a) as the base semantic segmentation architecture, with a ResNet-101 (K. He et al. 2016) backbone model pre-trained on ImageNet (Deng et al. 2009). The predictors for segmentation and depth estimation are two separate Atrous Spatial Pyramidal Pooling (ASPP) modules applied in parallel on the last layer from the backbone. The discriminator is composed of a bilinear fusion block with $K = 50$ full-rank

chunks in dimension $d = 200$, followed by five convolutional layers with kernel 4×4, stride 2, $\{64, 128, 256, 512, 1\}$ channels respectively, and leaky Rectified Linear Unit (ReLU) as activation function.

The main network is learned with Stochastic Gradient Descent (SGD), with an initial learning rate of 2.5×10^{-4} polynomially decayed with a factor of 0.9, a momentum of 0.9 and a weight decay of 5×10^{-4}. The loss weights are set to $\lambda_{\text{depth}} = 0.001$ and $\lambda_{\text{seg}} = 0.001$. The threshold τ of the reverse Huber function (Equation 3.4) is set to $1/5$ of the maximum error in the mini-batch. The discriminator is trained with Adam, with 10^{-4} as the initial learning rate for the same polynomial decay, and $(0.9, 0.99)$ for momentum. Each mini-batch contains one source image of size 1280×760 pixels and one target image of size 1024×512 pixels.

3.2.2.3 Results and comparison between approaches

In this work, we study several ways to leverage privileged depth information for UDA and to integrate it within a simple baseline, in order to focus on this aspect with fair comparisons. These experiments all train a single feature extractor model with two separate task heads, as illustrated in the BerMuDA's architecture figure (Figure 3.7), but merge the modalities with different methods. The improvement brought by BerMuDA compared to other fusion strategies is detailed in Table 3.1.

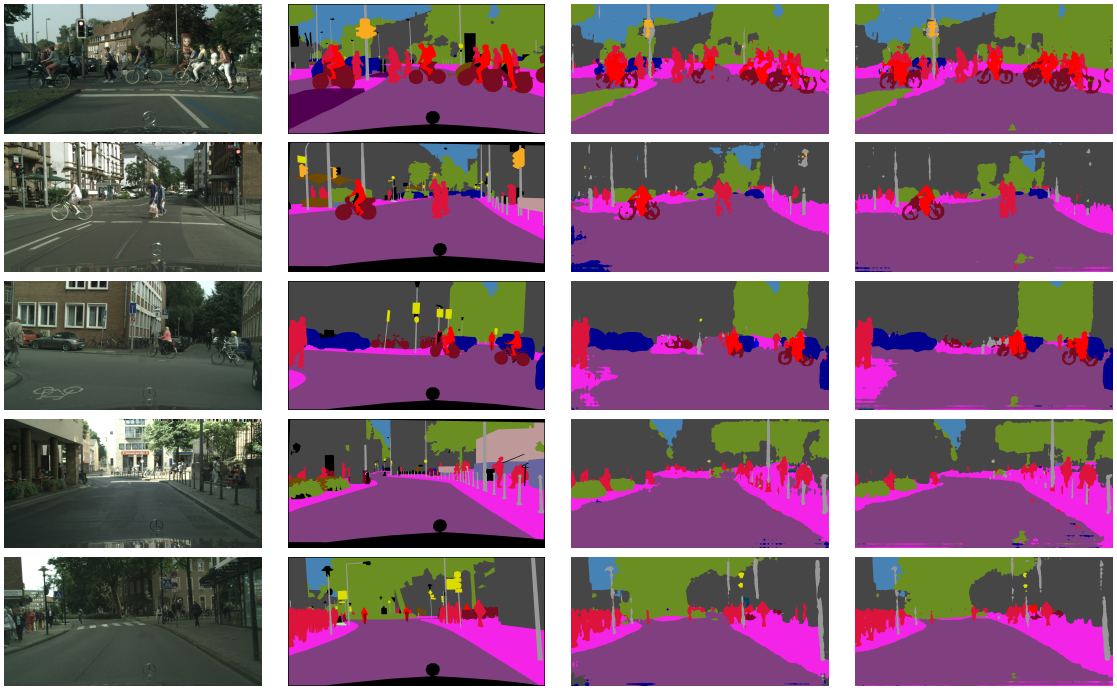
Table 3.1 – Comparison of modality fusion approaches in mIoU (%).

Model				Results	
Name	Approach used in	Privileged information	Modality fusion	mIoU-16	mIoU-13
Segmentation discriminator	AdaptSegNet (Tsai et al. 2018)	-	-	39.0	45.6
Independent discriminators	-	✓	-	39.0	45.9
Joint concatenation	-	✓	concatenation	39.3	46.0
Joint product	DADA (Vu et al. 2019b)	✓	product	39.5	46.3
Joint bilinear	BerMuDA	✓	bilinear	40.2	47.0

The first row shows the baseline method, where no depth information is used, and the discriminator is applied on segmentation output only. This correspond to the approach used by AdaptSegNet (Tsai et al. 2018). All subsequent rows leverage depth in different ways. On the second row is a model simply integrating depth with two independent discriminators applied on segmentation and depth outputs, respectively. Its results are comparable with the baseline's ones, with a slight improvement of 0.3 points only in mIoU-13, and show that adding depth supervision does not improve transfer between domains by itself if used in a naive way. The last three rows present several variants of a joint discriminator, taking both modalities as input, with differences lying in the way to fuse them. The first version uses a simple concatenation and the second an element-wise

product, similar to DADA (Vu et al. 2019b). The last one is our proposed model, with a bilinear discriminator. It yields the best performance, with improvements of 1.2 and 1.4 points in both metrics with respect to not using depth information. The other two variants of a joint discriminator have more limited improvements, indicating that the alignment step is useful to leverage depth effectively.

Figure 3.9 displays qualitative visualization of segmentation results. These examples show that including the depth as PI with BerMuDA leads to finer segmentation results and, in particular, the model better detects Vulnerable Road Users associated to the *pedestrian* and *rider* classes.



(a) Image (b) Ground truth (c) AdaptSegNet (d) BerMuDA

Figure 3.9 – **Qualitative segmentation results.** Input images are presented in (a) with corresponding ground truths in (b). We show segmentation predictions by the baseline AdaptSegNet (Tsai et al. 2018) in (c) and by our proposed model BerMuDA in (d). It is noticeable that BerMuDA obtains finer segmentation masks than AdaptSegNet for some practically important classes, such as the *pedestrian* and *rider* ones in these examples.

Finally, we compare the performance of BerMuDA with the other works of the UDA with PI in the literature in Table 3.2.

BerMuDA achieves 40.2% and 47.0% in mIoU-16 and mIoU-13 respectively. It is noticeable that it yields better performance than SPIGAN (K.-H. Lee et al. 2019)

Table 3.2 – Comparison of mIoU results (in %) for SYNTHIA. → Cityscapes experiments.

Models	road	sidewalk	building	wall	fence	pole	light	sign	veg	sky	person	rider	car	bus	mbike	bike	mIoU-16	mIoU-13
SPIGAN (K.-H. Lee et al. 2019)	71.1	29.8	71.4	3.7	**0.3**	**33.2**	6.4	**15.6**	81.2	78.9	52.7	13.1	75.9	25.5	10.0	20.5	36.8	42.5
DADA (Vu et al. 2019b)	**89.2**	**44.8**	**81.4**	6.8	**0.3**	26.2	**8.6**	11.1	**81.8**	**84.0**	54.7	19.3	**79.7**	**40.7**	14.0	**38.8**	**42.6**	**49.8**
AdaptSegNet (Tsai et al. 2018)	87.0	38.5	77.7	6.6	0.1	24.7	5.6	7.9	78.2	83.4	52.4	18.5	73.5	30.5	14.7	24.5	39.0	45.6
BerMuDA [ours]	87.6	41.9	79.2	**10.8**	**0.3**	21.2	3.7	5.7	78.5	83.1	**55.7**	**21.1**	76.5	33.2	**19.9**	24.4	40.2	47.0

(36.8% and 42.5%), which also leverages depth as privileged information to improve adaptation between domains.

Furthermore, it is interesting to compare BerMuDA to DADA (Vu et al. 2019b), as they are both based around similar ideas, as discussed before. DADA achieves 42.6% and 49.8%, but we can notice that BerMuDA yields better scores on most of the Vulnerable Road Users classes, which are arguably some of the most important classes for autonomous driving safety in practice. Specifically, BerMuDA reaches 55.7% for the *person* class, 21.1% for the *rider* class and 19.9% for the *motorbike* one. This fact that privileged depth information is especially useful for Vulnerable Road Users has already been hinted at in (Vu et al. 2019b), and we here confirm it with more emphasis.

In addition to a multimodal fusion, DADA also includes two recent advances, namely a feature fusion step (Mordan et al. 2018) in the main network, which modifies its architecture to tailor it to primary MTL and to focus more on segmentation than on depth estimation, and adversarial adaptation on weighted self-information of AdvEnt (Vu et al. 2019a) instead of the soft-segmentation predictions of AdaptSegNet (Tsai et al. 2018) we use in this work. Both of these improvements have been shown to yield better results and are orthogonal to the choice of fusion. We have shown in Table 3.1 that our bilinear fusion is superior to the element-wise product used in DADA. Thus, we believe that, as future work, integrating it within DADA, *i.e.*, combining it with the two improvements, should then result in greater performance.

3.3 Conclusion

This chapter has investigated UDA in the presence of additional information in the source domain only, thus called PI. Having access to more annotated data, even for a different task, proves to help significantly improve the performance of the UDA model on its primary task.

I introduced BerMuDA, a contribution of this thesis to UDA for semantic segmentation with PI, such as depth information. Under an adversarial training framework, its discriminator aligns all modalities thanks to a bilinear fusion operation. This alignment step enables the learning of the privileged modality to better transfer to the adaptation between domains.

It is notable that latest approaches, *e.g.*, Domain Invariant Structure Extraction (DISE) (Chang et al. 2019) or Bidirectional Learning (BDL) (Y. Li et al. 2019a), explore pseudo-labeling and CycleGAN-based image translation strategies, which yield great results but are also computationally heavy to train. Since they do not affect the architecture of the discriminator, they should be complementary to the use of depth as PI through a multimodal discriminator. Overall, the main idea behind a bilinear multimodal discriminator is to optimize over a large family of bilinear functions applied on segmentation and depth predictions, encompassing a wide variety of relevant representations. Although this operation is well suited to merge depth maps, as bilinear interactions are more expressive, they should also better generalize to other kinds of structured PI and not just depth maps.

While this chapter developed how to leverage additional data in the source domain, which is supposedly easily and automatically accessible when considering synthetic source data, it would be extremely relevant to wonder whether a similar strategy could be applied directly to the target data and whether it would be possible to automatically collect annotation from the unlabeled target data and use it to improve a UDA learning. Chapter 4 develops this concept and introduces ways to extract and use confident pseudo-labels in the context of UDA.

CHAPTER 4

ESTIMATING AND EXPLOITING CONFIDENT PSEUDO-LABELS FOR SELF-TRAINING

Chapter abstract

Self-Training (ST) is a semi-supervised learning strategy that aims at exploiting the predictions of a pre-trained model on unlabeled data to produce pseudo-annotations that may be used in further training as true annotations. This chapter studies how to employ this strategy in the context of Unsupervised Domain Adaptation (UDA) for semantic segmentation, how to estimate confident pseudo-labels as well as how to exploit them to improve existing UDA methods. It introduces two of our contributions to ST for UDA; the first one, Entropy-based Self-supervised Learning (ESL), exploits the entropy of the predictions as a measure of confidence, and the second one, Confidence Learning for Domain Adaptation (ConDA), trains an auxiliary model to predict the confidence of the model in its predictions.

The work in this chapter has led to the publication of a conference workshop paper and a journal paper:

- Antoine Saporta, Tuan-Hung Vu, Matthieu Cord, and Patrick Pérez (2020). "ESL: Entropy-guided self-supervised learning for domain adaptation in semantic segmentation". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshop on Scalability in Autonomous Driving*;
- Charles Corbière, Nicolas Thome, Antoine Saporta, Tuan-Hung Vu, Matthieu Cord, and Patrick Perez (2021). "Confidence Estimation via Auxiliary Models". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*.

Contents

The objective of Unsupervised Domain Adaptation (UDA), developed in Chapter 2, is to train a model to perform on an unlabeled target dataset. While the literature review of Chapter 2 shows that UDA strategies perform much better by actively using this unlabeled target data rather than simply using a model previously trained on a labeled source dataset, UDA model performance is still far from the performance of models trained on the same target data in a fully-supervised fashion. Finding better ways to exploit this unlabeled data from the self-supervised (X. Liu et al. 2021) or semi-supervised learning (X. Yang et al. 2021) community could help acquire supervisory signals on the target domain that should improve the performance of any UDA model.

Self-Training (ST) with pseudo-labeling (D.-H. Lee 2013) is a simple strategy that relies on picking up the current predictions on the unlabeled data and using them as if they were true labels for further training. D.-H. Lee (2013) show that the effect of pseudo-labeling is equivalent to entropy regularization (Grandvalet and Bengio 2005). In a UDA setting, the idea is to collect pseudo-labels on the unlabeled target-domain samples to have an additional supervision loss in the target domain while making sure to select only reliable pseudo-labels, such that the performance of the adapted semantic segmentation network effectively improves.

This chapter focuses on ST and pseudo-labeling in the context of UDA for semantic segmentation; in particular, how to extract pseudo-labels, and how to use them in a UDA learning procedure. It first describes a unified ST protocol for extracting and exploiting pseudo-labels in UDA frameworks and proposes an overview of ST strategies in the UDA literature. After this general review, I discuss the confidence metric these methods consider for the model predictions, employed to select confident pseudo-labels.

Afterward, the chapter presents two of my contributions to ST for UDA in semantic segmentation, both focused on proposing confidence criteria in the model predictions which are used to estimate pseudo-labels. The first contribution, Entropy-based Self-supervised Learning (ESL), developed in Section 4.2, proposes a simple yet effective entropy criterion to extract pseudo-labels while the second contribution, Confidence Learning for Domain Adaptation (ConDA), developed in Section 4.3, introduces an auxiliary neural network to properly estimate the confidence of the model in its predictions.

4.1 Self-Training in Unsupervised Domain Adaptation

This section describes a unified ST framework, applicable to most UDA strategy. Then, it completes the literature review of Chapter 2 by discussing ST approaches for UDA in semantic segmentation.

4.1.1 Self-Training Framework

This section establishes a unified framework for exploiting pseudo-labels extracted from a given model for ST, independent of the method of extraction. Two confidence criteria for pseudo-labels extraction will be presented afterwards in the following Sections (Section 4.2 and Section 4.3)

In addition to the standard UDA strategy, an extra self-supervision with pseudo-labels on the target dataset $\mathcal{X}_t$, on which no ground-truth annotation is available, can be included to improve the performance of the segmentation network F. By attaching pseudo-labels $\hat{\boldsymbol{y}}_t$ to the target images $\boldsymbol{x}_t$, the objective function of F with ST can be written:

$$\mathcal{L}_F^{\text{ST}} = \mathcal{L}_F + \frac{\lambda_{\text{ST}}}{|\mathcal{X}_t|} \sum_{\boldsymbol{x}_t \in \mathcal{X}_t} \mathcal{L}_{\text{seg}}(\boldsymbol{P}_{\boldsymbol{x}_t}, \hat{\boldsymbol{y}}_t), \tag{4.1}$$

with a weight λ_{ST} to balance the ST term, $\boldsymbol{P}_{\boldsymbol{x}_t}$ being the softmax output of the segmentation model $\boldsymbol{P}_{\boldsymbol{x}_t} = F(\boldsymbol{x}_t)$, $\mathcal{L}_{\text{seg}}$ being the pixel-wise Cross-Entropy (CE) defined in Equation 2.5 and $\mathcal{L}_F$ being the objective function of F without ST (for output-based adversarial alignement strategies, written as in Equation 2.16).

ST in UDA leverages the domain alignment that has already been achieved by the UDA strategy, assuming that the predictions of the current segmentation

network F on the target domain are relatively accurate. Following this principle, a ST approach to UDA can be described as follows (see Figure 4.1):

(1) Train a segmentation network on the target domain using standard UDA;

(2) Collect pseudo-labels on the target training dataset from the predictions of this network;

(3) Train a new semantic segmentation network from scratch using standard UDA combined with supervised training on target domain data with pseudo-labels;

(4) Possibly, repeat from step (2) by collecting finer pseudo-labels after each iteration of the algorithm.

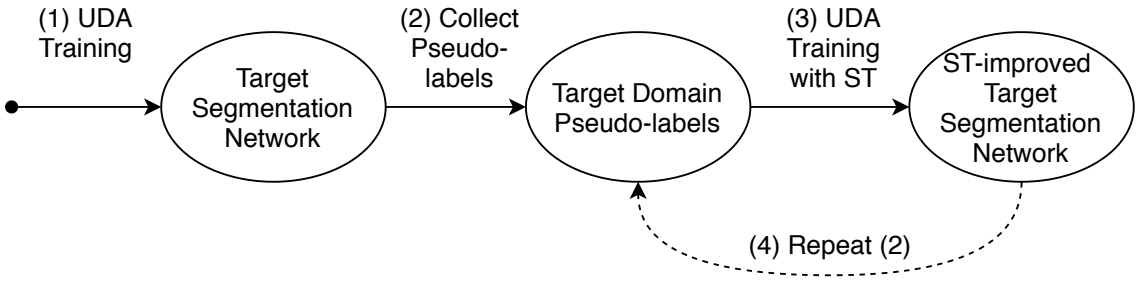


Figure 4.1 – **Self-training for Unsupervised Domain Adaptation**. A segmentation model is first learned with Unsupervised Domain Adaptation and used to collect pseudo-labels on target domain images. These automatically annotated data are used to subsequently retrain the model, an operation that can be iterated.

While the general idea of ST is simple and intuitive, collecting good pseudo-labels in step (2), which improves segmentation performance, is quite tricky, since erroneous predictions can significantly deteriorate performances. Indeed, in case many erroneous predictions are present among the pseudo-labels, ST can be counter-productive and significantly deteriorate the performance of the final model in the target domain. Thus, a measure of confidence should be used to gather only the most reliable predictions and use them as pseudo-labels.

4.1.2 Self-Training in the Literature

Maximum Class Probability. A standard strategy to select such pseudo-labels is to use the softmax score, or Maximum Class Probability (MCP), as a measure of confidence in the prediction. Y. Li et al. (2019a) perform multiple UDA training iterations; Figure 4.2 illustrates their architecture. At each iteration, their method trains a new generative model to do image-to-image translation from

source images to target style (see Section 2.2.2.1) based on the previous model, then output-space adversarial alignment (see Section 2.2.3.2), and finally extracts pseudo-labels using a MCP criterion based on a single threshold before re-training the model to complete the training iteration.

Enforcing this maximum softmax score to be greater than a threshold is a common practice to filter poor predictions from the constructed pseudo-labels. Formally, the pseudo-labels extracted using the trained segmentation network F can be written:

$$\hat{\boldsymbol{y}}_{t[h,w,c]} = \begin{cases} 1, & \text{if } \arg\max_{\widetilde{c}} \boldsymbol{P}_{\boldsymbol{x}_t[h,w,\widetilde{c}]} = c \\ & \text{and } \boldsymbol{P}_{\boldsymbol{x}_t[h,w,c]} > \mu^{(c)} \\ 0, & \text{otherwise,} \end{cases} \tag{4.2}$$

where $\mu^{(c)}$ is a threshold over the softmax prediction score for class c. Pixels with maximum class score below the relevant threshold are assigned a null pseudo-label vector $\hat{\boldsymbol{y}}_{t[h,w,\cdot]} = \mathbf{0}$ (not one-hot then). This assignment effectively excludes such pixels from the segmentation loss $\mathcal{L}_{\text{seg}}(\boldsymbol{x}_t, \hat{\boldsymbol{y}}_t)$ according to its definition in Equation 2.5.

A simple way to define this threshold, as in (Y. Li et al. 2019b), would be:

$$\mu^{(c)} = \min\left(\mu^*, \text{median}\left\{\boldsymbol{P}_{\boldsymbol{x}_t[h,w,c]} \,|\, \boldsymbol{x}_t \in \mathcal{X}_t, h \in [H], w \in [W] \wedge \arg\max_{\widetilde{c}} \boldsymbol{P}_{\boldsymbol{x}_t[h,w,\widetilde{c}]} = c\right\}\right), \tag{4.3}$$

where μ^* is a hyper-parameter. This threshold ensures that at least 50% of the predictions for each class are kept based on their softmax prediction score and that all the predicted labels with a maximum softmax prediction score greater than μ^* are kept on the easier-to-predict classes (on which the prediction scores are rather high over the training set).

Overall, the strategy of ST as described in Section 4.1.1 using this softmax-based pseudo-label selection criterion is refered as Softmax-based Self-supervised Learning (SSL) in what follows.

Class-Balanced Self-Training. Zou et al. (2018) propose an iterative ST procedure where the pseudo-labels are generated based on a loss minimization. While very close to an MCP criterion, the softmax output is normalized by a separate parameter for each class, determining the proportion of selected pseudo-labels for the considered class. The difference between these parameters introduce a different level of bias for each class and tackles inter-class balance.

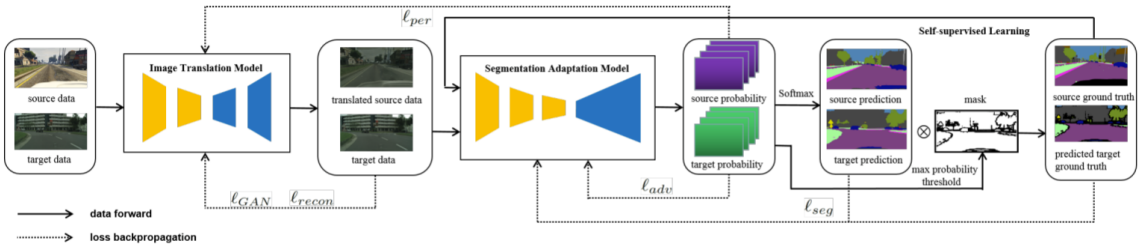


Figure 4.2 – **Bidirectional Learning Network Architecture**. An image translation model and the segmentation adaptation model collaborate and improve each other at each training iteration. Pseudo-labels are extracted based on the MCP and used in self-training to further enhance the segmentation performance. Illustrations from (Y. Li et al. 2019a)

Instance Adaptive Self-Training. Mei et al. (2020) propose yet another twist on the MCP approach called Instance Adaptive Self-Training. Their pseudo-label selection does not rely simply on a single threshold per class, but dynamically updates these thresholds based on the current image instances depending on the confidence probability of the class in the image.

Confidence-Regularized Self-Training. Zou et al. (2019) identify that selected pseudo-labels with MCP-based criteria may lead to overconfident mistakes and upgrades (Zou et al. 2018) by introducing multiple types of confidence regularization to limit the propagation of errors caused by noisy pseudo-labels. First, they suggest a label regularization based on negative entropy that only depends on the pseudo-labels. Second, based on the network softmax output probabilities, they propose multiple model regularization based on L_2, negative entropy, and Kullback-Leibler (KL) distillation with the uniform distribution.

Discussion. In the pseudo-label extraction strategies previously described, the MCP score is used as a confidence score for the prediction. I argue that the MCP score is not the best measure of confidence of the network. Indeed, while the MCP score may be greater than the given threshold, this measure does not take into account the softmax prediction score over other classes, overlooking potentially high softmax prediction scores on those other classes that would question the confidence of the model. This phenomenon has been discussed in (Corbière et al. 2019) and will be developed into more details in Section 4.3.1. I introduce in the rest of this chapter two different ways to address this issue by proposing better measures of confidence of Deep Neural Network (DNN) predictions and by using them to extract better semantic segmentation pseudo-labels for UDA.

4.2 ESL : Entropy-guided Pseudo-Label Generation

As discussed in the previous section, the MCP, which is commonly used as a measure of confidence for ST in UDA, may lead to catastrophic errors in the pseudo-label generation due to the behavior of DNN. As a consequence, we propose an entropy-guided pseudo-label extraction strategy that uses the entropy of the softmax prediction as a measure of confidence. Unlike the MCP, the entropy takes into account the complete distribution of the softmax prediction score for each pixel, making this measure more reliable in assessing the confidence of the network. Such a behavior of MCP and entropy is illustrated in Figure 4.3.

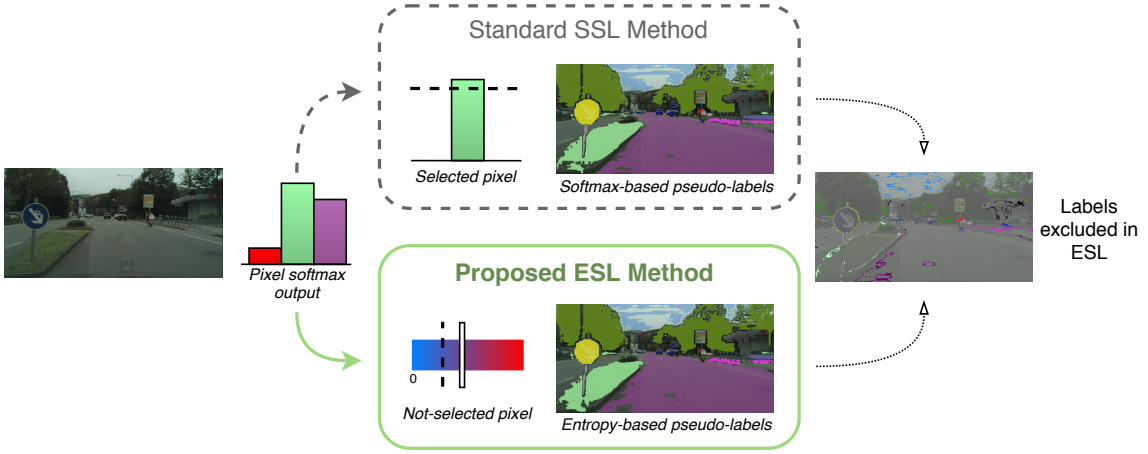


Figure 4.3 – **Comparison of Softmax-based Self-supervised Learning (SSL) and Entropy-based Self-supervised Learning (ESL)**. While SSL only considers the maximum softmax prediction score, ESL uses the entropy of the output distribution to better assess the confidence of the prediction. Less confident predictions in terms of entropy are excluded from the pseudo-labels by ESL (see right-most figure, best viewed in color and zoomed-in), effectively improving the quality of the label map and boosting the performance of self-trained segmentation networks.

4.2.1 Method

We propose to use the entropy of the predictions of the model to measure its confidence in the predictions. The extracted pseudo-labels using this measure can be written as follow:

$$\hat{\boldsymbol{y}}_t[h,w,c] = \begin{cases} 1, & \text{if } \arg\max_{\widetilde{c}} \boldsymbol{P}_{\boldsymbol{x}_t}[h,w,\widetilde{c}] = c \\ & \text{and } \boldsymbol{E}_{\boldsymbol{x}_t}[h,w] < \nu^{(c)} \\ 0, & \text{otherwise} \end{cases} \tag{4.4}$$

where the entropy $\boldsymbol{E}_{\boldsymbol{x}_t}[h,w]$ is defined as:

$$\boldsymbol{E}_{\boldsymbol{x}_t}[h,w] = -\frac{1}{\log(C)} \sum_{c=1}^{C} \boldsymbol{P}_{\boldsymbol{x}_t}[h,w,c] \log \boldsymbol{P}_{\boldsymbol{x}_t}[h,w,c] \tag{4.5}$$

and $\nu^{(c)}$ is a threshold over the entropy score for pixels of class c. Similarly to the threshold of the previous method, we can define $\nu^{(c)}$ as:

$$\nu^{(c)} = \max\left(\nu^*, \text{median}\left\{\boldsymbol{E}_{\boldsymbol{x}_t}[h,w] \,|\, \boldsymbol{x}_t \in \mathcal{X}_t, h \in [H], w \in [W] \wedge \arg\max_{\widetilde{c}} \boldsymbol{P}_{\boldsymbol{x}_t}[h,w,\widetilde{c}] = c\right\}\right), \tag{4.6}$$

where ν^* is a hyper-parameter. This threshold ensures at least the 50% most confident predictions in terms of entropy are kept for each class for our pseudo-labels. Moreover, all the predictions with an entropy score lower than the hyper-parameter ν^* are kept for the easier to predict classes on which the segmentation network is more confident. Overall, the strategy of ST as described in Section 4.1.1 using this entropy-based pseudo-label selection criterion is refered as ESL.

4.2.2 Experimental Results

We present experimental results on models trained with self-supervised learning techniques on various semantic segmentation domain adaptation datasets. We compare them to different baselines, showing that self-supervised learning helps boost the performance and that models trained with entropy-guided self-supervised learning consistently outperform baselines and models with standard self-supervised learning. The code developed for this work is available on GitHub [1].

1. https://github.com/valeoai/ESL

4.2.2.1 Architectures and baseline frameworks

We experiment with three state-of-the-art domain adaptation baselines. The three of them are based on DeepLab-V2 (L.-C. Chen et al. 2018a) for the semantic segmentation module of the architecture, but the domain adaptation frameworks of these baselines are different:

- AdaptSegNet (Tsai et al. 2018) considers semantic segmentation as structured outputs that contain spatial similarities between source and target domains. For this reason, they adopt an adversarial learning framework on the output space. Moreover, they construct a multi-level adversarial network to perform adaptation at different feature levels.

- Adversarial Entropy Minimization (AdvEnt) (Vu et al. 2019a), alternatively, adopts an adversarial learning framework on the entropy of the pixel-wise predictions instead of the raw softmax output predictions as in AdaptSegNet (Tsai et al. 2018).

- Bidirectional Learning (BDL) (Y. Li et al. 2019b), as AdaptSegNet (Tsai et al. 2018), conducts adaptation on the output space of the semantic segmentation module. The method adds two main components to the framework: first, an image translation module based on CycleGAN (Zhu et al. 2017) to transfer the style of the target domain to source domain images ; second, a bidirectional learning training procedure in which this image translation module and the semantic segmentation module are trained alternately and contribute to each other's performance. Furthermore, this strategy already incorporates ST using standard pseudo-label extraction. In our experiments, we will focus on two given steps of the sequential model training on GTA5 $\rightarrow$ Cityscapes, called 'Step 1' and 'Step 2', which can be found on the authors' GitHub [2]. We apply ST strategies on those two pre-trained models, possibly adding Image Translation (IT).

4.2.2.2 Implementation details

Implementations are done with the PyTorch Deep Learning (DL) framework (Paszke et al. 2017). Training and validation of the models are done on a single NVIDIA 1080TI GPU with 11GB memory. The semantic segmentation models are initialized with the ResNet-101 (K. He et al. 2016) pre-trained on ImageNet (Deng et al. 2009). The semantic segmentation models are trained by a Stochastic Gradient Descent optimizer (Bottou 2010) with learning rate 2.5×10^{-4}, momentum 0.9 and weight decay of 10^{-4}. The discriminators are trained by an Adam optimizer

2. https://github.com/liyunsheng13/BDL

(Diederik P. Kingma 2015) with learning rate 10^{-4}. We fix λ_{adv} as 10^{-3} and λ_{SL} as 1. Following the recommendation from (Y. Li et al. 2019b), we use a value of 0.9 for μ^* in the SSL experiments. Moreover, we adopt a value of 0.1 for ν^* in the ESL experiments and justify this choice in Section 4.2.2.4.

4.2.2.3 Quantitative Results

We consider two synthetic source datasets – SYNTHIA and GTA5 – and two real target datasets – Cityscapes and Mapillary Vistas. The datasets are described into more details in Chapter 2 (Section 2.3.2). SYNTHIA is the only synthetic source dataset considered in Chapter 3 due to the need for depth annotation for training with Privileged Information (PI). Since the setting considered in this chapter is the one of traditional UDA, we also consider as source the GTA5 dataset, in particular in the very popular GTA5 $\rightarrow$ Cityscapes setting.

GTA5 $\rightarrow$ Cityscapes. Table 4.1 reports semantic segmentation performance in terms of mIoU (%) on the Cityscapes validation set using GTA5 as source domain. As explained in Section 4.2.2.1, 'BDL (step 1)' and 'BDL (step 2)' represent the two pre-trained models which can be found on the authors' GitHub. We can notice that ESL consistently outperforms SSL on every setup, giving the best performance for each baseline framework. The performance absolute change in terms of mIoU using ESL compared to the baseline state-of-the-art models range from $+1.0\%$ to $+2.2\%$, which is a significant improvement. Along with the quantitative results, Figure 4.4 displays some samples of pseudo-labels extracted with both SSL and ESL. First, the pseudo-label maps look globally similar inside of the semantic regions between SSL and ESL.

Indeed, in these regions, the softmax prediction is often very peaky with a high maximum score and low entropy. Nevertheless, marked differences can be observed along the boundaries of these regions. These transition areas are those where the prediction models are the most uncertain, despite maximum scores remaining often high. Indeed, the boundaries should be the areas where the model is the less confident about its predictions. This clearly shows that most missing pixels in both pseudo-label maps lie on such locations. Around the boundaries, however, the prediction entropy tends to get higher even if the softmax prediction score stays high. This results in pixels being rightly excluded from the ESL pseudo-label map while present in the SSL one (column (d) of Figure 4.4). Reversely, there are much fewer pixels that are added to ESL compared to SSL.

SYNTHIA $\rightarrow$ Cityscapes. Table 4.2 reports semantic segmentation performance in terms of mIoU (%) on the Cityscapes validation set using SYNTHIA

Table 4.1 – Comparison of mean Intersection over Union (mIoU) results (in %) for GTA5 → Cityscapes experiments.

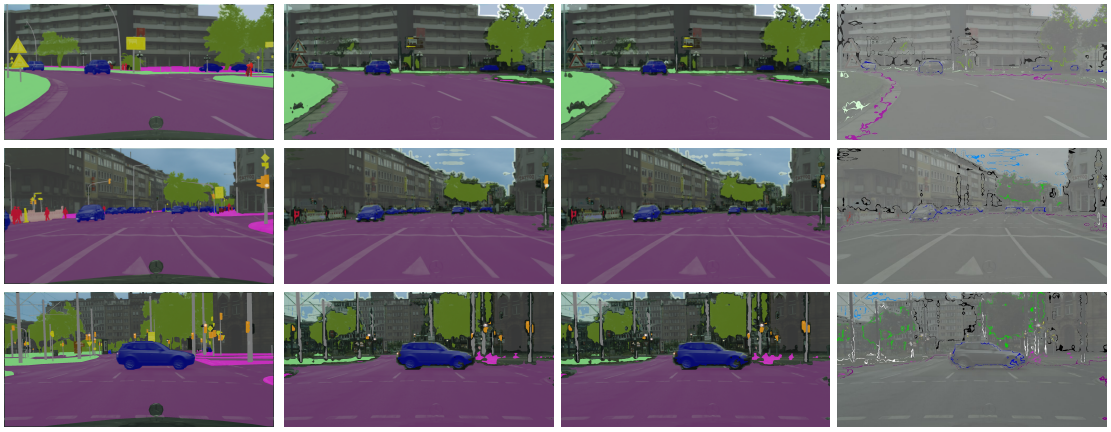
GTA5 → Cityscapes		
Method	Self-Training	mIoU-19
AdaptSegNet (Tsai et al. 2018)	-	37.6
	SSL	38.9
	ESL	39.7
AdvEnt (Vu et al. 2019a)	-	43.7
	SSL	45.6
	ESL	45.9
BDL (step 1) (Y. Li et al. 2019a)	-	44.3
	SSL	45.0
	ESL	45.6
	SSL + IT	46.8
	ESL + IT	47.2
BDL (step 2) (Y. Li et al. 2019a)	-	47.3
	SSL	47.0
	ESL	47.6
	SSL + IT	48.2
	ESL + IT	48.6

as source domain. Again, ESL consistently outperforms SSL on every setup, even when SSL fails to improve the performance over the baseline.

Table 4.2 – Comparison of mIoU results (in %) for SYNTHIA → Cityscapes experiments. mIoU* is the 13-class setup (excluding the classes 'wall', 'fence' and 'pole') as used in earlier works.

SYNTHIA → Cityscapes			
Method	Self-Training	mIoU-16	mIoU-13
AdaptSegNet (Tsai et al. 2018)	-	40.0	46.5
	SSL	39.7	46.0
	ESL	40.4	46.9
AdvEnt (Vu et al. 2019a)	-	41.2	48.3
	SSL	43.1	50.2
	ESL	43.5	50.7

Mapillary Vistas. Table 4.3 reports semantic segmentation performance in terms of mIoU (%) on the Mapillary Vistas validation set using SYNTHIA as



(a) Ground-truth labels (b) SSL pseudo-labels (c) ESL pseudo-labels (d) Excluded by ESL

Figure 4.4 – **Qualitative results of Entropy-based pseudo-labels.** The four columns visualize (a) ground-truth label maps, (b) standard SSL pseudo-labels with maximum softmax predictions, (c) our entropy-based ESL pseudo-labels and (d) pseudo-labels in (b) but excluded by our entropy criterion. Most excluded pixels lie in region boundaries where segmentation models are the most uncertain. Experiments show that excluding them boosts the performance of self-trained networks. There are much fewer pseudo-labels added in (c) compared to (b), thus we don't display them.

source domains and ADVENT as baseline model. Proposed ESL outperforms SSL and improves over the baseline performance.

Table 4.3 – Comparison of mIoU results (in %) for Mapillary Vistas experiments.

Method	Self-Training	flat	construction	object	nature	sky	human	vehicle	mIoU-7
AdvEnt (Vu et al. 2019a)	-	87.5	**63.2**	29.2	72.2	**88.8**	43.1	70.4	64.9
	SSL	88.3	55.4	29.1	73.3	81.8	**52.4**	75.7	65.1
	ESL	**88.4**	55.7	**32.0**	**75.4**	84.3	43.5	**76.2**	**65.4**

4.2.2.4 Ablation Studies

Choice of threshold ν^*. Let us describe how to choose the threshold ν^* such that a good balance is achieved between having as many high confidence predicted labels as possible and avoiding as much as possible noise from incorrect predictions. A quick computation suggests 0.1 as a good threshold value. Indeed, considering the maximum softmax prediction score for a given pixel is 0.95 on a

19-classes setup, the entropy of the distribution would range from 0.07 in the best case to 0.12 in the worst case. We confirm this choice experimentally. Table 4.4 segmentation results on the domain adaptation problem GTA5 $\rightarrow$ Cityscapes with the AdvEnt baseline using different thresholds in the ESL label extraction. Alternately, we show the limit case where the threshold is always selected as the median of the entropy for each class. The result of this experiment is shown on the last row of Table 4.4. When the threshold is higher than 0.1, the incorrect predictions degrade the quality of the pseudo-label maps and induce more noise during training. When the threshold is lower than 0.1, we don't keep as many confident predictions, slightly reducing the effectiveness of the pseudo-labeling. This experiment confirms that 0.1 seems to be a good threshold for ESL.

Table 4.4 – Influence of threshold $\nu*$ in ESL. Baseline model is AdvEnt (Vu et al. 2019a)

GTA5 $\rightarrow$ Cityscapes							
Threshold	Baseline	0.05	0.1	0.15	0.2	0.3	Median
mIoU-19	43.7	45.6	**45.9**	45.5	45.3	44.9	45.1

Incorrect predictions in the pseudo-labels. In order to further motivate the choice of entropy as a confidence measure in the proposed ESL method over the softmax prediction score of SSL, Table 4.5 reports the ratio of incorrect predictions (in $\%$) in the selected pseudo-labels for every class for both SSL and ESL on the GTA5 $\rightarrow$ Cityscapes experiment with the AdvEnt baseline (the lower the score, the better). Additionally, the last row of the Table displays the relative change in the ratio of incorrect predictions for every class. The results show that ESL performs significantly better than SSL on the easier-to-predict classes (less than 10% incorrect predictions in the selected pseudo-labels). Indeed, ESL reduces the number of incorrect predictions for those classes by a significant margin, ranging from 4.2% up to 18.6%. This means that ESL induces significantly less noise during training for those classes. These changes can be explained by the "overconfidence" the model may have in terms of softmax prediction score for easier-to-predict classes, which is not as significant for the entropy as measure of confidence. Overall, ESL decreases the ratio of incorrect predictions for 14 classes out of 19 and globally reduces the ratio of incorrect predictions by 3.7% over SSL.

Table 4.5 – Comparison of incorrect predictions (in %) selected in the pseudo-labels extracted. Baseline architecture is AdvEnt (Vu et al. 2019a)

GTA5 → Cityscapes																				
Method	road	sidewalk	building	wall	fence	pole	light	sign	veg	terrain	sky	person	rider	car	truck	bus	train	mbike	bike	global
SSL	19.3	**33.5**	9.8	54.0	51.3	**33.8**	21.3	10.4	4.0	**37.7**	6.9	4.4	33.7	2.9	53.2	50.0	66.9	**27.5**	**15.8**	14.5
ESL	**18.5**	36.1	**8.7**	**53.4**	**51.2**	35.1	**20.5**	**10.2**	**3.4**	38.7	**6.5**	**4.2**	**33.1**	**2.4**	**53.1**	**49.8**	**59.8**	28.8	17.0	**13.9**
Rel. Change (%)	**-3.9**	+8.0	**-11.7**	**-1.1**	**-0.3**	+3.9	**-3.6**	**-2.2**	**-16.1**	+2.6	**-5.3**	**-4.2**	**-1.8**	**-18.6**	**-0.2**	**-0.5**	**-10.7**	+5.0	+7.2	**-3.7**

4.3 ConDA : Pseudo-Label Generation with Learned Confidence

The previous section shows that entropy serves as a better measure of confidence than MCP for DNN predictions, which is particularly impactful in the performance of models trained with ST in a UDA context. Nonetheless, even though it may cover some cases where MCP is overconfident, it is not an infallible measure of confidence. Ideally, the True Class Probability (TCP) would be the perfect measure of confidence since it directly gives the probability of the predicted class being the ground-truth class. However, the TCP is not practically accessible when the ground-truth is unknown. ConDA, one of our ST for UDA contribution, proposes to learn an auxiliary model that estimates the TCP of the semantic segmentation network and serves as a measure of confidence for pseudo-label extraction on the unlabeled target data.

4.3.1 Confidence Network

I shortly present some work done by co-authors of our journal paper (Corbière et al. 2021) previously published in the conference article (Corbière et al. 2019). This work defines a neural network that gives the confidence of a model in its predictions. Our contribution on pseudo-labeling presented in Section 4.3 extends this method to the task of semantic segmentation and Domain Adaptation (DA), which are not studied in the original work of the co-authors (Corbière et al. 2019).

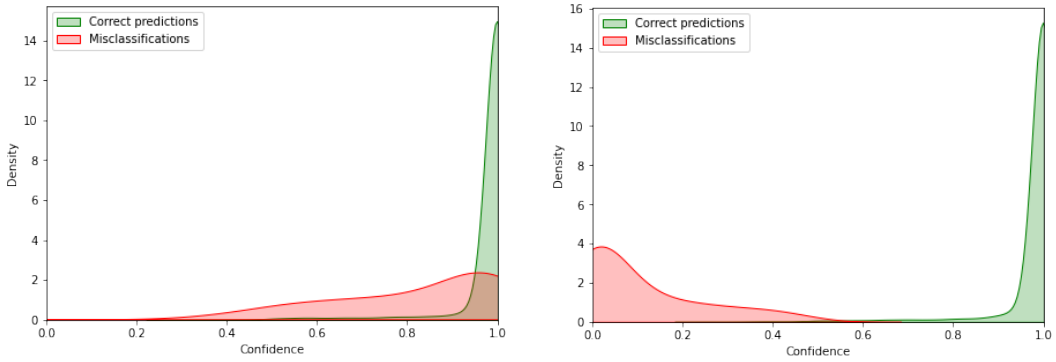
4.3.1.1 True Class Probability as confidence-rate function

In Section 4.3.1, to simplify, let's consider a classification task with classes $\mathcal{Y}$. For a given input $\boldsymbol{x}$, a standard confidence-rate function for a classifier F is the probability associated to the predicted max-score class, that is the MCP:

$$\text{MCP}_F(\boldsymbol{x}) = \max_{k \in \mathcal{Y}} P(Y = k|\boldsymbol{x}) = \max_{k \in \mathcal{Y}} F(\boldsymbol{x})[k]. \tag{4.7}$$

However, by taking the largest softmax probability as confidence estimate, MCP leads to high confidence values both for correct and erroneous predictions alike, making it hard to distinguish them, as shown in Figure 4.5a. On the other hand, when the model misclassifies an example, the probability associated to the true class y is lower than the maximum one and likely to be low. Based on this simple observation, we propose to consider instead this TCP as a suitable confidence-rate function. For any admissible input $\boldsymbol{x} \in \mathcal{X}$, we assume the *true* class $y(\boldsymbol{x})$ is known, which we denote y for simplicity. The TCP confident rate is defined as

$$\text{TCP}_F(\boldsymbol{x}, y) = P(Y = y|\, \boldsymbol{x}) = F(\boldsymbol{x})[y]. \tag{4.8}$$



(a) Maximum Class Probability (b) True Class Probability

Figure 4.5 – **Distributions of different confidence measures over correct and erroneous predictions of a given model.** When ranking according to MCP (a) the test predictions of a convolutional model trained on CIFAR-10, we observe that correct ones (in green) and misclassifications (in red) overlap considerably, making it difficult to distinguish them. On the other hand, ranking samples according to TCP (b) alleviates this issue and allows a much better separation.

Simple guarantees. With TCP, the following properties hold. Given a properly labelled example $(\boldsymbol{x}, y)$, then:

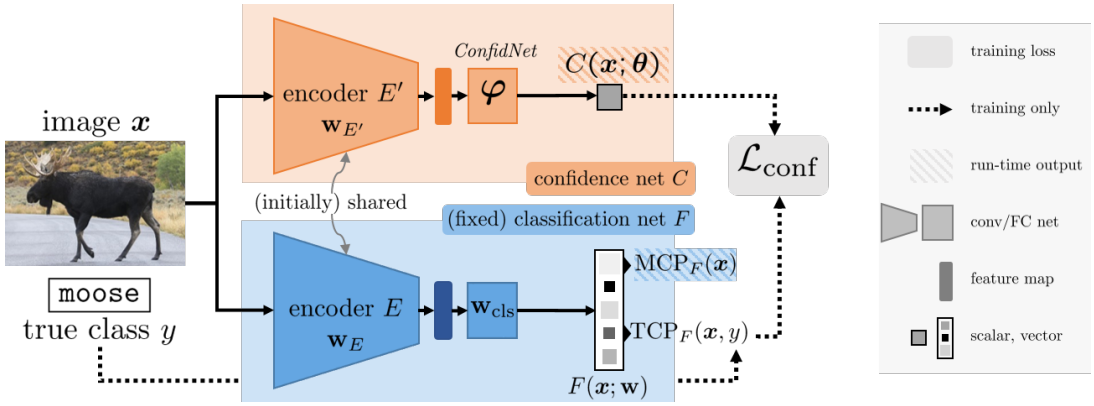


Figure 4.6 – **ConfidNet learning confidence approach.** The fixed classification network F, with parameters $\mathbf{w} = (\mathbf{w}_E, \mathbf{w}_{\text{cls}})$, is composed of a succession of convolutional and fully-connected layers (encoder E) followed by last classification layers with softmax activation. The auxiliary confidence network C, with parameters $\boldsymbol{\theta}$, builds upon the feature maps extracted by the encoder E, or its fine-tuned version E' with parameters $\mathbf{w}_{E'}$: they are passed to ConfidNet, a trainable multi-layer module with parameters $\boldsymbol{\phi}$. The auxiliary model outputs a confidence score $C(\boldsymbol{x};\boldsymbol{\theta}) \in [0,1]$, with $\boldsymbol{\theta} = \boldsymbol{\phi}$ in absence of encoder fine-tuning and $\boldsymbol{\theta} = (\mathbf{w}_{E'}, \boldsymbol{\phi})$ in case of fine-tuning.

- $\text{TCP}_F(\boldsymbol{x},y) > 1/2 \Rightarrow f(\boldsymbol{x}) = y$, i.e. the example is correctly classified by the model;

- $\text{TCP}_F(\boldsymbol{x},y) < 1/K \Rightarrow f(\boldsymbol{x}) \neq y$, i.e. the example is wrongly classified by the model,

where class prediction $f(\boldsymbol{x})$ is defined by:

$$f(\boldsymbol{x}) = \arg\max_{k \in \mathcal{Y}} \; P(Y = k|\boldsymbol{x}) = \arg\max_{k \in \mathcal{Y}} F(\boldsymbol{x})[k]. \tag{4.9}$$

Within the range $[1/K, 1/2]$, there is no guarantee that correct and incorrect predictions will not overlap in terms of TCP. However, when using deep neural networks, we observe that the actual overlap area is extremely small in practice, as illustrated in Figure 4.5b on the CIFAR-10 dataset (Krizhevsky and Hinton 2009). One possible explanation comes from the fact that modern deep neural networks output overconfident predictions and, therefore, non-calibrated probabilities (Guo et al. 2017).

4.3.1.2 Predict the True Class Probability with a neural network

Using TCP as a confidence-rate function on a model's output would be of great help when it comes to reliably estimate its confidence. However, the true classes y are obviously not available when estimating confidence on test inputs.

The work proposes to *learn TCP confidence from data*. More formally, for the classification task at hand, we consider a parametric selective classifier (f, g), with f based on an already-trained neural network F. They aim at deriving its companion selection function g from a learned estimate of the TCP function of F. To this end, they introduce an *auxiliary model* C, with parameters $\boldsymbol{\theta}$, that is intended to predict TCP_F and to act as a confidence-rate function for the selection function g. An overview of their proposed approach is available in Figure 4.6. This model is trained such that, at runtime, for an input $\boldsymbol{x} \in \mathcal{X}$ with (unknown) true label y:

$$C(\boldsymbol{x}; \boldsymbol{\theta}) \approx \text{TCP}_F(\boldsymbol{x}, y). \tag{4.10}$$

In practice, this auxiliary model C will be a neural network trained under full supervision on a dataset $(\mathcal{X}, \mathcal{Y})$ to produce this confidence estimate. To design this network, knowledge can be transferred from the already-trained classification network. Throughout its training, F has indeed learned to extract increasingly-complex features that are fed to its final classification layers. Calling E the encoder part of F, a simple way to transfer knowledge consists in defining and training a multi-layer head with parameters $\boldsymbol{\phi}$ that regresses TCP_F from features encoded by E. This module is called Confidence Network (ConfidNet). As a result of this design, the complete confidence network C is composed of a frozen encoder followed by trained ConfidNet layers. The whole architecture might be later fine-tuned, including the encoder. In that case, $\boldsymbol{\theta}$ will encompass the parameters of both the encoder and the ConfidNet's layers.

4.3.2 Model

4.3.2.1 Selecting pseudo-labels with a confidence model

Following the ST framework previously described in Section 4.1.1, a confidence network C is learned at step (2) to predict the confidence of the UDA-trained semantic segmentation network F and used to select only trustworthy pseudo-labels on target-domain images. To this end, the framework of (Corbière et al. 2021) presented in Section 4.3.1 in an image classification setup needs here to be adapted to the structured output of semantic segmentation.

Semantic segmentation can be seen as a pixel-wise classification problem. Given a target-domain image $\boldsymbol{x}_t$, we want to predict both its soft semantic map $F(\boldsymbol{x}_t)$ and, using an auxiliary model with trainable parameters $\boldsymbol{\theta}$, its confidence map $C(\boldsymbol{x};\boldsymbol{\theta}) = \boldsymbol{C}^{\boldsymbol{\theta}}_{\boldsymbol{x}_t} \in [0,1]^{H\times W}$. Given a pixel (h,w), if its confidence $\boldsymbol{C}^{\boldsymbol{\theta}}_{\boldsymbol{x}_t[h,w]}$ is above a chosen threshold δ, we label it with its predicted class $f(\boldsymbol{x}_t)_{[h,w]} = \operatorname{argmax}_c \boldsymbol{P}_{\boldsymbol{x}_t[h,w,c]}$, otherwise it is masked out. Computed over all images in the target dataset $\mathcal{X}_t$, these incomplete segmentation maps constitute target pseudo-labels that are used to train a new semantic-segmentation network. Optionally, we may repeat from step (2) and learn alternately a confidence model to collect pseudo-labels and a segmentation network using this ST.

4.3.2.2 Confidence training with adversarial loss

To train the segmentation confidence network C, we propose to jointly optimize two objectives. Following the approach presented in Section 4.3.1, the first one supervises the confidence prediction on annotated source-domain examples using the known TCPs for the predictions from F. Specific to semantic segmentation with UDA, the second one is an adversarial loss that aims at reducing the domain gap between source and target. A complete overview of the approach is provided in Figure 4.7.

Confidence loss. On annotated source-domain images, it requires C to predict at each pixel the score assigned by F to the (known) true class:

$$\mathcal{L}_{\text{conf}} = \frac{1}{|\mathcal{X}_s|} \sum_{\boldsymbol{x}_s \in \mathcal{X}_s} \left\| \boldsymbol{C}^{\boldsymbol{\theta}}_{\boldsymbol{x}_s} - \text{TCP}_F(\boldsymbol{x}_s, \boldsymbol{y}_s) \right\|^2_{\text{F}}, \tag{4.11}$$

where $\|\cdot\|_{\text{F}}$ denotes the Frobenius norm and, for an image $\boldsymbol{x}$ with true segmentation map $\boldsymbol{y}$ and predicted soft one $F(\boldsymbol{x})$, we note

$$\text{TCP}_F(\boldsymbol{x}, \boldsymbol{y})_{[h,w]} = F(\boldsymbol{x})[h, w, \arg\max \boldsymbol{y}_{[h,w,\cdot]}] \tag{4.12}$$

at spatial position (h,w). On a new input image, C should predict at each pixel the score that F will assign to the unknown true class, which will serve as a confidence measure.

However, compared to the application in Section 4.3.1, we have here the additional problem of the gap between source and target domains, an issue that might affect the training of the confidence model as in the training of the segmentation model.

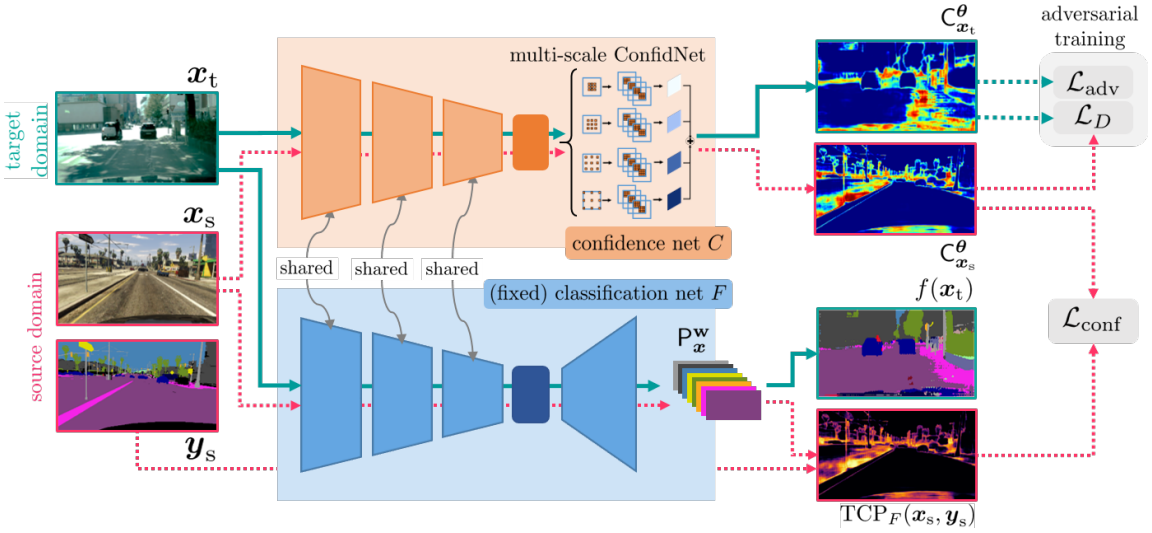


Figure 4.7 – **Overview of proposed Confidence Learning for Domain Adaptation (ConDA) in semantic segmentation**. Given images in source and target domains, we pass them to the encoder part of the segmentation network F to obtain their feature maps. This network F is fixed during this phase and its weights are not updated. The confidence maps are obtained by feeding these feature maps to the trainable head of the confidence network C, which includes a multi-scale ConfidNet module. For source-domain images, a regression loss $\mathcal{L}_{\text{conf}}$ (Equation 4.11) is computed to minimize the distance between $\boldsymbol{C}^{\boldsymbol{\theta}}_{\boldsymbol{x}_s}$ and the fixed true-class-probability map $\text{TCP}_F(\boldsymbol{x}_s, \boldsymbol{y}_s)$. An adversarial training scheme – based on discriminator's loss $\mathcal{L}_D(\boldsymbol{\psi})$ (Equation 4.13) and adversarial part $\mathcal{L}_{\text{adv}}(\boldsymbol{\theta})$ of confidence net's loss (Equation 4.15) –, is also added to enforce the consistency between the $\boldsymbol{C}^{\boldsymbol{\theta}}_{\boldsymbol{x}_s}$'s and $\boldsymbol{C}^{\boldsymbol{\theta}}_{\boldsymbol{x}_T}$'s. Dashed arrows stand for paths that are used only at train time.

Adversarial loss. The second objective concerns the domain gap. While model C learns to estimate TCP on source-domain images, its confidence estimation on target-domain images may suffer dramatically from this domain shift. As classically done in UDA, we propose adversarial learning of our auxiliary model to address this problem. More precisely, we want the confidence maps produced by C in the source domain to resemble those obtained in the target domain.

A discriminator D_C, with parameters $\boldsymbol{\psi}$, is trained concurrently with C with the aim to recognize the domain (1 for source, 0 for target) of an image given its confidence map. The following loss is minimized w.r.t. $\boldsymbol{\psi}$:

$$\mathcal{L}_{D_C} = \frac{1}{|\mathcal{X}_s|} \sum_{\boldsymbol{x}_s \in \mathcal{X}_s} \mathcal{L}_{C,\text{adv}}(\boldsymbol{x}_s, 1) + \frac{1}{|\mathcal{X}_t|} \sum_{\boldsymbol{x}_t \in \mathcal{X}_t} \mathcal{L}_{C,\text{adv}}(\boldsymbol{x}_t, 0), \tag{4.13}$$

where $\mathcal{L}_{\text{adv}}$ denotes the Binary Cross-Entropy (BCE) loss of the discriminator based on confidence maps:

$$\mathcal{L}_{C,\text{adv}}(\boldsymbol{x}, \lambda) = -\lambda \log \left(D_C(\boldsymbol{C}^{\boldsymbol{\theta}}_{\boldsymbol{x}}; \boldsymbol{\psi})\right) - (1-\lambda) \log(1 - D_C(\boldsymbol{C}^{\boldsymbol{\theta}}_{\boldsymbol{x}}; \boldsymbol{\psi})), \tag{4.14}$$

for $\lambda \in \{0, 1\}$, which is a function of both $\boldsymbol{\psi}$ and $\boldsymbol{\theta}$. In alternation with the training of the discriminator using Equation 4.13, the adversarial training of the confidence net is conducted by minimizing, w.r.t. $\boldsymbol{\theta}$, the following loss:

$$\mathcal{L}_C = \mathcal{L}_{\text{conf}} + \frac{\lambda_{\text{adv}}}{|\mathcal{X}_t|} \sum_{\boldsymbol{x}_t \in \mathcal{X}_t} \mathcal{L}_{C,\text{adv}}(\boldsymbol{x}_t, 1), \tag{4.15}$$

where the second term, weighted by λ_{adv}, encourages C to produce maps in target domain that will confuse the discriminator.

This adversarial scheme for confidence learning also acts as a regularizer during training, improving the robustness of the unknown TCP target confidence. As the training of C may be unstable, adversarial training provides additional information signal, in particular, imposing that confidence estimation should be invariant to domain shifts. We empirically observed that this adversarial confidence learning provides better confidence estimates and improves the convergence and stability of the training scheme.

4.3.2.3 Multi-scale ConfidNet architecture

In semantic segmentation, models consist of fully convolutional networks where hidden representations are 2D feature maps. This is in contrast with the architecture of classification models considered in Section 4.3.1. As a result, ConfidNet module must have a different design here: instead of fully-connected layers, it is composed of 1×1 convolutional layers with an adequate number of channels.

In many segmentation datasets, the existence of objects at multiple scales may complicate confidence estimation. As in works dealing with varying object sizes (L.-C. Chen et al. 2018a), we further improve our confidence network C by adding a multi-scale architecture based on Atrous Spatial Pyramidal Pooling (ASPP). It consists of a computationally efficient scheme to re-sample a feature map at different scales, and then, to aggregate the confidence maps.

From a feature map, we apply parallel atrous convolutional layers with 3×3 kernel size and different sampling rates, each of them followed by a series of 4 standard convolutional layers with 3×3 kernel size. In contrast with convolutional layers with large kernels, atrous convolution layers enlarge the field of view of filters and help to incorporate a larger context without increasing the number of parameters and the computation time. The resulting features are then summed

before upsampling to the original image size of $H \times W$. A final sigmoid activation outputs a confidence map with values between 0 and 1.

The whole architecture of the confidence model C is represented in the orange block of Figure 4.7, along with its training given a fixed segmentation model F (blue block) with which it shares the encoder. Such as in Section 4.3.1, fine-tuning the encoder within C is also possible, although we did not explore the option in this semantic segmentation context due to the excessive memory overhead it implies.

4.3.3 Experimental Results

In this section, we analyze on several semantic segmentation benchmarks the performance of ConDA, our approach to domain adaptation with learned confidence-based ST. We report comparisons with state-of-the-art methods on each benchmark. We also analyze further the quality of ConDA's pseudo-labeling and demonstrate via an ablation study the importance of each of its components.

4.3.3.1 Experimental setup

As in the experiments with ESL (Section 4.2), we experiment in the common GTA5 $\rightarrow$ Cityscapes setup as well as two other benchmarks – SYNTHIA $\rightarrow$ Cityscapes and SYNTHIA $\rightarrow$ Mapillary Vistas. The datasets are described into more detail in Section 2.3.2.

We evaluate the proposed ST method on AdvEnt (Vu et al. 2019a). All the implementations are done with the PyTorch framework (Paszke et al. 2017). The semantic segmentation models are initialized with DeepLabv2 backbones pre-trained on ImageNet (Krizhevsky et al. 2012). Due to computational constraints, we only train the multi-scale ConfidNet without encoder fine-tuning.

GTA5 $\rightarrow$ Cityscapes. The results of semantic segmentation on the Cityscapes validation set using GTA5 as source domain are available in Table 4.6. All the methods rely on DeepLabv2 as their segmentation backbone. We first notice that ST-based methods from the literature are superior on this benchmark, with performance reaching up to 48.6% mIoU with ESL (Section 4.2). ConDA outperforms all those methods by reaching 49.9% mIoU.

SYNTHIA $\rightarrow$ Cityscapes. Table 4.7 reports in a consistent way adaptation results for the task SYNTHIA $\rightarrow$ Cityscapes. Following relevant literature on this

Table 4.6 – **Comparative performance of ConDA for GTA5 → Cityscapes experiments.** Results in per-class Intersection over Union (IoU) and class-averaged mIoU on GTA5 → Cityscapes. All methods are based on a DeepLabv2 backbone.

		GTA5 → Cityscapes																			
Method	Self-Train.	road	sidewalk	building	wall	fence	pole	light	sign	veg	terrain	sky	person	rider	car	truck	bus	train	mbike	bike	mIoU-19
AdaptSegNet (Tsai et al. 2018)		86.5	25.9	79.8	22.1	20.0	23.6	33.1	21.8	81.8	25.9	75.9	57.3	26.2	76.3	29.8	32.1	7.2	**29.5**	32.5	41.4
CyCADA (Hoffman et al. 2018)		86.7	35.6	80.1	19.8	17.5	**38.0**	**39.9**	**41.5**	82.7	27.9	73.6	**64.9**	19.0	65.0	12.0	28.6	4.5	31.1	42.0	42.7
DISE (Chang et al. 2019)		91.5	47.5	82.5	31.3	25.6	33.0	33.7	25.8	82.7	28.8	82.7	62.4	30.8	85.2	27.7	34.5	6.4	25.2	24.4	45.4
AdvEnt (Vu et al. 2019a)		89.4	33.1	81.0	26.6	26.8	27.2	33.5	24.7	83.9	36.7	78.8	58.7	30.5	84.8	38.5	44.5	1.7	31.6	32.4	45.5
CBST (Zou et al. 2018)	✓	91.8	53.5	80.5	32.7	21.0	34.0	28.9	20.4	83.9	34.2	80.9	53.1	24.0	82.7	30.3	35.9	16.0	25.9	**42.8**	45.9
MRKLD (Zou et al. 2019)	✓	91.0	55.4	80.0	33.7	21.4	37.3	32.9	24.5	85.0	34.1	80.8	57.7	24.6	84.1	27.8	30.1	**26.9**	26.0	42.3	47.1
BDL (Y. Li et al. 2019a)	✓	91.0	44.7	84.2	34.6	**27.5**	30.2	36.0	36.0	85.0	**43.6**	83.0	58.6	**31.6**	83.3	35.3	49.7	3.3	28.8	35.6	48.5
ESL (Section 4.2)	✓	90.2	43.9	84.7	35.9	28.5	31.2	37.9	34.0	84.5	42.2	83.9	59.0	32.2	81.8	36.7	49.4	1.8	30.6	34.1	48.6
ConDA	✓	**93.5**	**56.9**	**85.3**	**38.6**	26.1	34.3	36.9	29.9	**85.3**	40.6	**88.3**	58.1	30.3	**85.8**	**39.8**	**51.0**	0.0	28.9	37.8	**49.9**

Table 4.7 – **Comparative performance of ConDA for SYNTHIA → Cityscapes experiments**. Results in per-class IoU and aggregated mIoU on SYNTHIA → Cityscapes ('mIoU*' is the 13-class setup, excluding the classes 'wall', 'fence' and 'pole', as used in earlier works). All methods are based on a DeepLabv2 backbone.

		SYNTHIA → Cityscapes																	
Method	Self-Train.	road	sidewalk	building	wall	fence	pole	light	sign	veg	sky	person	rider	car	bus	mbike	bike	mIoU-16	mIoU-13
AdaptSegNet (Tsai et al. 2018)		84.3	42.7	77.5	-	-	-	4.7	7.0	77.9	82.5	54.3	21.0	72.3	32.2	18.9	32.3	-	46.7
DISE (Chang et al. 2019)		**91.7**	**53.5**	77.1	2.5	0.2	27.1	6.2	7.6	78.4	81.2	55.8	19.2	82.3	30.3	17.1	34.3	41.5	48.8
AdvEnt (Vu et al. 2019a)		85.6	42.2	79.7	8.7	0.4	25.9	5.4	8.1	80.4	84.1	57.9	23.8	73.3	36.4	14.2	33.0	41.2	48.0
CBST (Zou et al. 2018)	✓	68.0	29.9	76.3	10.8	1.4	33.9	**22.8**	29.5	77.6	78.3	60.6	28.3	81.6	23.5	18.8	39.8	42.6	48.9
MRKLD (Zou et al. 2019)	✓	67.7	32.2	73.9	10.7	**1.6**	**37.4**	22.2	**31.2**	80.8	80.5	**60.8**	**29.1**	**82.8**	25.0	19.4	45.3	43.8	50.1
BDL (Y. Li et al. 2019a)	✓	83.9	43.7	80.2	**12.9**	0.5	30.1	18.0	17.3	79.7	83.5	52.2	25.8	72.5	35.5	**25.8**	45.4	44.2	51.0
ConDA	✓	88.1	46.7	**81.1**	10.6	1.1	31.3	22.6	19.6	**81.3**	**84.3**	53.9	21.7	79.8	**42.9**	24.2	**46.8**	**46.0**	**53.3**

dataset, mIoU results for 16 categories and for 13 categories are available. Again, ConDA achieves state-of-the-art performance on this benchmark with 46.0% mIoU.

SYNTHIA → Mapillary. Along with results on Cityscapes, we further study domain adaptation on another target dataset, namely Mapillary Vistas. Table 4.8 presents semantic segmentation performance using SYNTHIA as source dataset. This benchmark has also been used in other recent works, such as in AdvEnt (Vu et al. 2019a) and Depth-Aware Domain Adaptation (DADA) (Vu et al. 2019b). ConDA outperforms baseline method with 66.4% mIoU compared to 65.2% mIoU in AdvEnt.

Combining with PI. We also tested the proposed confidence-based ST approach on DADA (Vu et al. 2019b), another domain adaptation baseline which uses the depth information available on source-domain synthetic scenes as PI during segmentation training (see Chapter 3 and more specifically Figure 3.1.2). Again, the

Table 4.8 – **Comparative performance of ConDA for SYNTHIA → Mapillary experiments.** Results in per-class IoU and class-averaged mIoU on SYNTHIA → Mapillary.

SYNTHIA → Mapillary									
Method	Self-Train.	flat	constr.	object	nature	sky	human	vehicle	mIoU
AdvEnt (Vu et al. 2019a)		86.9	58.8	**30.5**	**74.1**	85.1	48.3	72.5	65.2
ConDA	✓	**89.1**	**63.5**	28.3	72.7	**88.2**	**49.7**	**73.0**	**66.4**

Table 4.9 – **Combining ConDA with DADA for UDA in semantic segmentation with PI.** Performance in IoU and mIoU on SYNTHIA → Mapillary.'ConDA*' is trained with DADA as baseline model with Bilinear Multimodal Domain Adaptation (BerMuDA) using depth as PI.

SYNTHIA → Mapillary									
Method	Self-Train.	flat	constr.	object	nature	sky	human	vehicle	mIoU
DADA (Vu et al. 2019b)		86.7	62.1	34.9	75.9	88.6	51.1	73.8	67.6
ConDA*	✓	**87.8**	**67.5**	**40.5**	**76.8**	**92.3**	**60.7**	**78.5**	**72.0**

proposed method (ConDA*) further increases performance from 67.6% mIoU to 72.0% mIoU.

Ablation Study. To study the effect of the adversarial training and of the multi-scale confidence architecture on the confidence model, we perform an ablation study on the GTA5 → Cityscapes benchmark. The results on domain adaptation after re-training the segmentation network using collected pseudo-labels are reported in Table 4.10. In this table, "ConfidNet" refers to the simple network architecture defined in Section 4.3.1 (adapted to segmentation by replacing the fully connected layers by 1×1 convolutions of suitable width); "Adv. ConfidNet" denotes the same architecture but with the adversarial loss from Section 4.3.2.2 added to its learning scheme; "Multi-scale ConfidNet" stands for the architecture introduced in Section 4.3.2.3; Finally, the full method, "ConDA" amounts to having both this architecture and the adversarial loss. We notice that adding the adversarial learning achieves significantly better performance, for both ConfidNet and multi-scale ConfidNet, with respectively $+1.4$ and $+0.8$ point increase. Multi-

scale ConfidNet (resp. adv. multi-Scale ConfidNet) also improves performance up to $+0.9$ point (resp. $+0.3$) from their ConfidNet architecture counterpart. These results stress the importance of both components of the proposed confidence model.

Table 4.10 – **Ablation study of ConDA on semantic segmentation with pseudo-labelling-based adaptation.** Full-fledged ConDA approach is compared on GTA5 $\rightarrow$ Cityscapes to stripped-down variants (with/without multi-scale architecture in ConfidNet, with/without adversarial learning).

Model	Multi-Scale.	Adv	mIoU-19
ConfidNet			47.6
Multi-Scale ConfidNet	✓		48.5
Adv. ConfidNet		✓	49.0
ConDA (Adv. Multi-scale ConfidNet)	✓	✓	**49.9**

Quality of pseudo-labels. Here we analyze the effectiveness of ConDA compared to MCP as confidence measures to select relevant pseudo-labels in the target domain. For a given fraction of retained pseudo-labels (coverage) on target-domain training images, we compare in Figure 4.8 the proportion of those labels that are correct (accuracy). ConDA outperforms MCP for all coverage levels, meaning it selects significantly fewer erroneous predictions for the next round of segmentation-model training. Along with the segmentation adaptation improvements presented earlier, these coverage results demonstrate that reducing the amount of noise in the pseudo-labels is key to learning a better segmentation adaptation model.

Figure 4.9 presents qualitative results of those pseudo-labels methods. We find again that MCP and ConDA seem to select around the same amount of correct predictions in their pseudo-labels, but with ConDA picking out a lot fewer erroneous ones.

4.4 Conclusion

ST is an interesting technique from the semi-supervised learning community that aims at generating pseudo-annotations of unlabeled data using the predictions of a pre-trained model. In the context of UDA, this is a popular strategy to generate supervision on the unlabeled target domain by leveraging the predictions of a model previously trained with a standard UDA strategy. This additional supervision helps improve substantially the performance of UDA approaches.

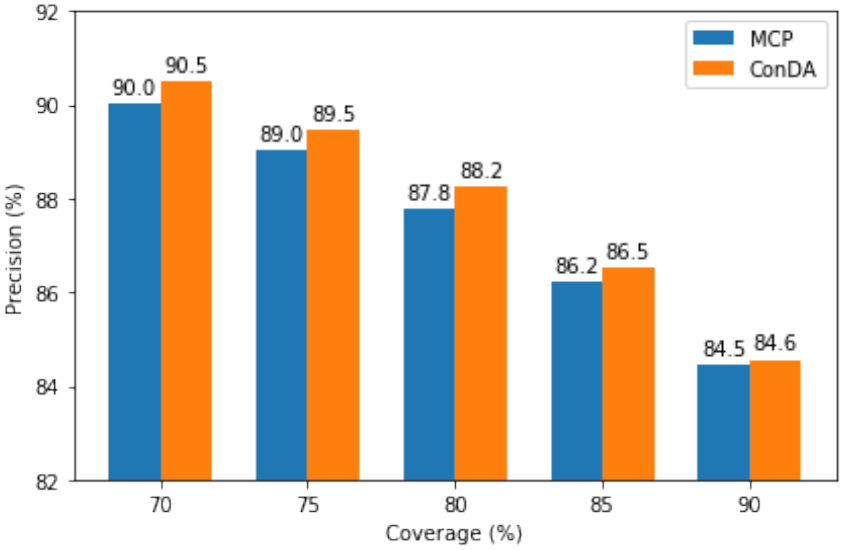


Figure 4.8 – **Comparative quality of selected pseudo-labels**. Proportion of correct pseudo-labels (precision) for different coverages on GTA5 → Cityscapes, for MCP and ConDA.

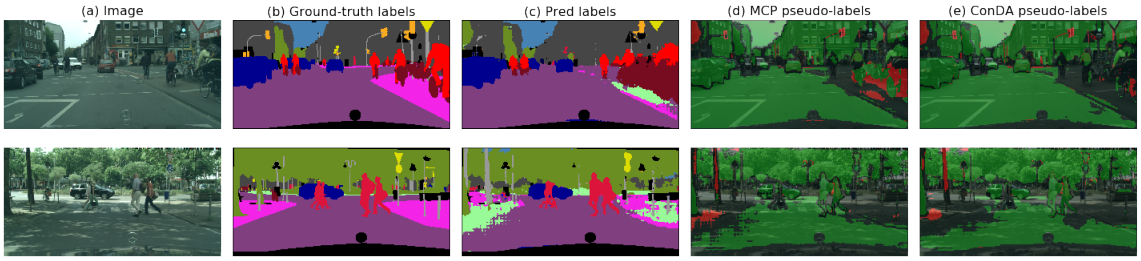


Figure 4.9 – **Qualitative results of pseudo-label selection of ConDA for semantic-segmentation adaptation.** The three first columns present target-domain images of the GTA5 → Cityscapes benchmark (a) along with their ground-truth segmentation maps (b) and the predicted maps before self-training (c). We compare pseudo-labels collected with MCP (d) and with ConDA (e). Green (resp. red) pixels are correct (resp. erroneous) predictions selected by the method and black pixels are discarded predictions. ConDA retains fewer errors while preserving approximately the same amount of correct predictions.

While most methods of the literature rely on MCP as a measure of confidence for the pseudo-label extraction, I argue that MCP tends to show overconfidence in some prediction mistakes of the model due to the behavior of DNN training. First, I introduced ESL that exploits the entropy of the predictions instead of the MCP, one to take into account the probability distribution over all the classes, which reduces the tendency of overconfidence of this criterion compared to MCP. While ESL proposes a simple yet effective ST method based on entropy, the TCP

would be an ideal measure of confidence for pseudo-label estimation. Neverthless, since the TCP is not directly accessible from the model predictions, the proposed method ConDA learns with adversarial training an auxiliary Neural Network (NN) with dedicated architecture that predicts the TCP of a semantic segmentation model. While more complex to implement than ESL due to the training of an additional NN, ConDA proves to improve significantly the performance of UDA models compared to ST methods based on the MCP.

The ST framework and pseudo-label estimation methods presented are extremely flexible. While the experiments mostly show ST on synthetic-to-real with adversarial UDA models, they may be applied on virtually any UDA setting and approach. For instance, I demonstrated the efficiency of ConDA in a PI information setting based on DADA (Vu et al. 2019b). Furthermore, Chapter 5 develops UDA to multiple target domains and also shows the addition of ST strategies is impactful when dealing with multiple unlabeled domains.

CHAPTER 5

ADAPTATION TO MULTIPLE DOMAINS

Chapter abstract

Beyond the traditional scope of Unsupervised Domain Adaptation (UDA) with a single source domain and a single target domain, real-world autonomous driving systems are confronted to a variety of scenarios to handle, may it be varying light conditions, numerous cities from all over the world or diverse weathers. In this context, one is confronted to UDA to multiple domains, which further increases the challenges of traditional UDA with the addition of distribution shifts existing within the different target domains. While also mentioning other extensions of traditional UDA, this chapter focuses on multi-target UDA and continual UDA and introduces our solutions to both these problems. Multi-Discriminator (Multi-Dis.) extends single-target adversarial UDA approaches to multi-target UDA by performing adversarial distribution alignment both between source-target pairs and between targets. Multi-Target Knowledge Transfer (MTKT) is another strategy to multi-target UDA that introduces multiple specialized segmentation branches, one for each target, and performs knowledge transfer from these experts to a domain-agnostic segmentation branch in a multi-teacher single-student fashion. Finally, Continual Target Knowledge Transfer (CTKT) extends this last approach to continual UDA by additionally distilling at each training step the knowledge from the previous model into the current model.

The first part of this chapter has led to the publication of a conference paper:

- Antoine Saporta, Tuan-Hung Vu, Matthieu Cord, and Patrick Pérez (2021). "Multi-Target Adversarial Frameworks for Domain Adaptation in Semantic Segmentation". In: *IEEE/CVF International Conference on Computer Vision (ICCV)*.

Contents

The Unsupervised Domain Adaptation (UDA) methods presented in Chapter 2 rely on one of two core ideas: either a distribution alignment between the source domain and the target domain on some intermediate features of the model using various strategies (direct, adversarial, etc.), or the transformation of the input images from one domain to the other, most often the source images translated into the target domain.

These strategies are effective in the traditional UDA setting as defined in Chapter 2, and Chapter 3 and Chapter 4 discussed how to leverage extra source data as Privileged Information (PI) and how to design new Self-Training (ST) schemes in this context.

We explore here another aspect of UDA. All these strategies rely on the fact that this standard setting only deals with two domains – one source and one target – and focuses on matching the representations or the inputs of one domain onto those of the other domain. This constitutes an important limitation of UDA since real-world applications are significantly more complex: for instance, an autonomous vehicle should operate in multiple countries, or under various weather conditions, all constituting different target domains from a UDA perspective. Furthermore, autonomous systems should be able to improve and adapt to new target domains, for instance when deploying autonomous vehicles to a new country.

These problems are still open in the UDA community, especially for the semantic segmentation task. This section explores both multi-target UDA and continual UDA for semantic segmentation, which are both novel UDA settings. On the former, Section 5.2 proposes two frameworks to extend standard adversarial single-target

UDA to multi-target UDA: Multi-Discriminator (Multi-Dis.), relying on source-target and target-targets alignment, and Multi-Target Knowledge Transfer (MTKT), introducing a multi-teacher single-student approach with multiple target-specific segmentation branches and a single target-agnostic segmentation branch. These extensions of adversarial UDA approaches may be combined with ST strategies discussed in Chapter 4, and we propose some experiments using Entropy-based Self-supervised Learning (ESL). Finally, with the Continual Target Knowledge Transfer (CTKT) framework, Section 5.3 extends the multi-target UDA approach MTKT to the more complex continual UDA in which the target domains are discovered sequentially and one at a time.

5.1 Extending the Standard Unsupervised Domain Adaptation Setting

This section extends the literature review of Chapter 2 with works related to traditional UDA but involving more than two domains. Then, it specifically focuses on multi-target UDA and continual learning, the lines of research I tackle in my contributions. I first give a brief overview of multi-domain learning settings in the Computer Vision (CV) literature.

Domain Generalization. Domain generalization (H. Li et al. 2018; Matsuura and Harada 2020; Shanshan Zhao et al. 2020; Pandey et al. 2021) is a challenging task close to the Domain Adaptation (DA) problem. Its objective is to learn a model from one or several training (source) domains that will be able to generalize to unseen testing (target) domains. While very close to a multi-domain UDA setting, their difference lies in the total absence of target data during training, leading to drastically different approaches to tackle these tasks.

Multi-Source Unsupervised Domain Adaptation. Multi-source UDA (Sicheng Zhao et al. 2019; J. He et al. 2021; Nguyen et al. 2021) aims at training using an arbitrary number of source domains to learn better, more generalizable features for a single target domain. While one could train a model with a standard UDA approach on the combination of all the considered source domains, it proves to be inefficient due to the discrepancy between the domains. Multi-source approaches effectively account for these distribution shifts and leverage them to produce more generalizable features.

Open-Compound Domain Adaptation. In Open-Compound Domain Adaptation (OCDA) (Z. Liu et al. 2020; Park et al. 2020), the target domain may be considered as a combination of multiple homogeneous target domains – for instance, similar weather conditions such as 'sunny', 'foggy', etc. – where the domain labels are not known during training. Moreover, previously unseen target domains may be encountered during inference.

Multi-Target Unsupervised Domain Adaptation. Multi-target UDA is still a fairly recent setting in the literature and mostly tackles classification tasks. While multi-target UDA is close to the OCDA setting in that it considers multiple target domains instead of a single one, multi-target UDA differs from this last setting in that it assumes that the domain of origin is known during training and that no new domains are faced at test time. Thus, for instance, multi-target UDA would be better suited to DA to multiple cities than OCDA since the origin of the training data should be easily accessible. Two main scenarios emerge in the works on this task. In the first one, even though the target is considered composed of multiple domains with gaps and misalignments, the domain labels are unknown during training and test. Peng et al. (2019) propose an architecture that extracts domain-invariant features by performing source-target domain disentanglement. Moreover, it also removes class-irrelevant features by adding a class disentanglement loss. In a similar setting, Z. Chen et al. (2019) present an adversarial meta-adaptation network that both aligns source with mixed-target features and uses an unsupervised meta-learner to cluster the target inputs into k clusters, which are adversarially aligned. In the second scenario, the target identities are labeled on the training samples but remain unknown during inference. To handle it, Yu et al. (2018) learn a common parameter dictionary from the different target domains and extract the target model parameters by sparse representation; Gholami et al. (2020) adopt a disentanglement strategy by separately capturing both domain-specific private features and feature representations by learning a domain classifier and a class label predictor, and train a shared decoder to reconstruct the input sample from those disentangled representations.

Tackling multi-target UDA in semantic segmentation has been proposed in a concurrent work to ours (Isobe et al. 2021). Isobe et al. (2021) train multiple semantic segmentation models, each one expert on a specific domain. These domain-specific expert models collaborate by being trained on images from the other domains stylized in the domain of expertise while making sure that the predicted map are coherent between the experts for a same original image. Finally, the knowledge of all these experts is transferred to another model, which serves as domain-generic student. Figure 5.1 illustrates their Collaborative Consistency Learning method.

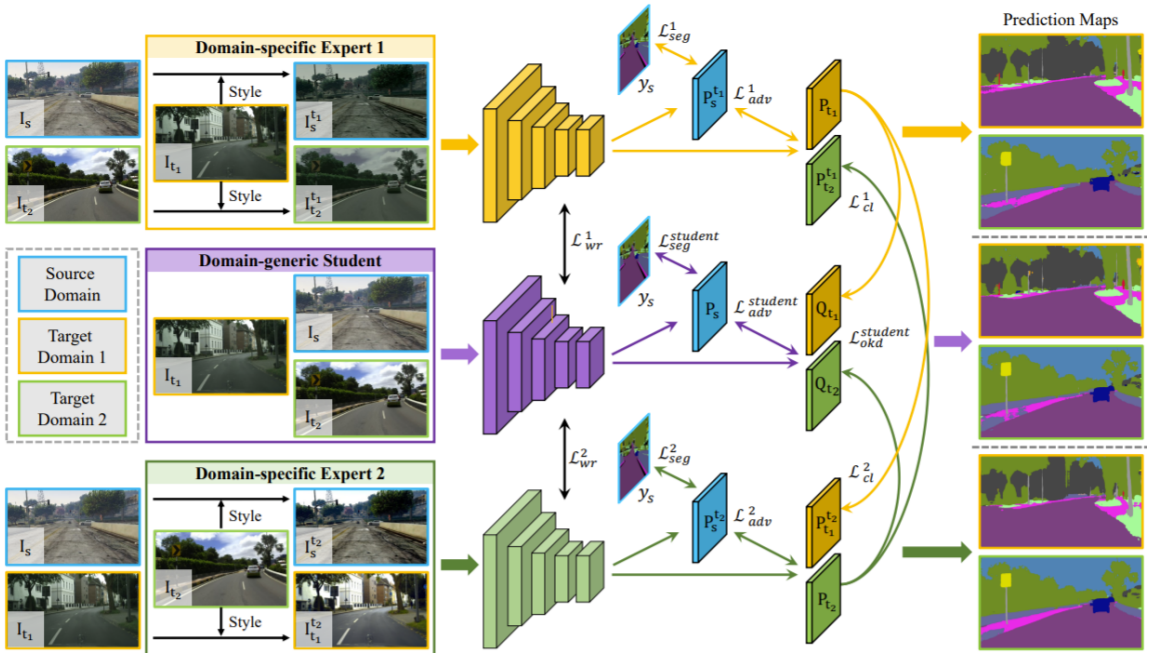


Figure 5.1 – **Collaborative Consistency Learning approach to multi-target UDA.** Each expert model is trained on images stylized in the domain of expertise and the expert models collaborate by ensuring the consistency of their predictions for a same original image. A domain-generic model is trained by ensuring its consistency with the domain experts. Illustration from (Isobe et al. 2021).

5.2 Multi-Target Adversarial Frameworks for Domain Adaptation in Semantic Segmentation

As discussed in the previous section, standard UDA strategies are not well suited to tackle adaptation to multiple target domains. This section develops frameworks that extend standard UDA adversarial methods to a multi-target setting.

5.2.1 Multi-Target Unsupervised Domain Adaptation

Problem Setting. We consider a different UDA scenario where $T \geq 2$ distinct target domains $\mathfrak{X}_{t,n}$, $n \in [T]$, must be jointly handled. These target domains are represented by unlabeled training sets $\mathcal{X}_{t,n}$, $n \in [T]$. Similar to the standard UDA setting, we assume that the annotated training examples $(\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{X}_s \times \mathcal{Y}_s$ stem from a single source domain $\mathfrak{X}_s$, for instance, a specific synthetic environment. The main goal is to train a single segmenter F that achieves equally good results on all target-domain test sets. While the target domain of origin is known for

all unlabeled training examples, we assume, as in classification approaches of (Gholami et al. 2020; Yu et al. 2018), that this information is not accessible at test time.

Revisiting Adversarial Unsupervised Domain Adaptation Approach. Many recent state-of-the-art approaches for UDA in semantic segmentation rely on output-space adversarial alignment (see Section 2.2.3.2).

Approaches like (Tsai et al. 2018) and (Vu et al. 2019a) handle only one source domain and one target domain. Such approaches fail to generalize to multiple domains, as illustrated in Figure 5.2, and require proper handling of the multiple domains during training to produce good results on every target domain.

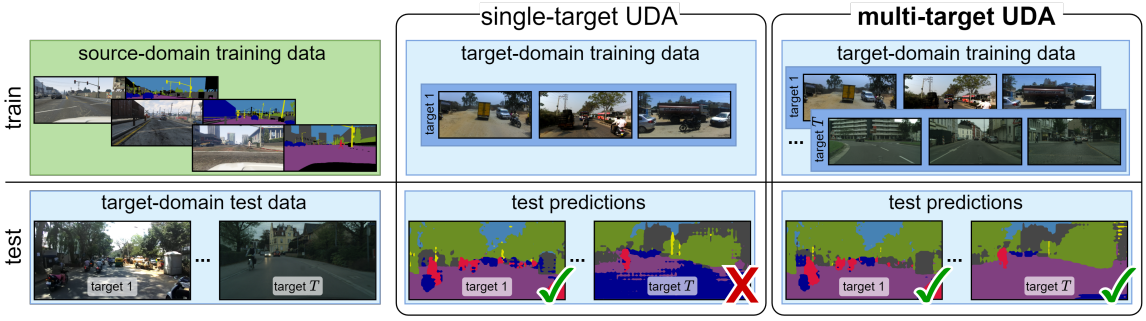


Figure 5.2 – **Multi-target unsupervised domain adaptation for semantic segmentation.** In the standard single-target setting, UDA methods produce good segmentation in the target domain they are trained on, but generalize poorly to other unseen domains. Multi-target UDA aims at excelling in the multiple domains the model is trained on. (*Top*) The information available during training is composed of source-domain RGB images with ground-truth semantic maps (green), here from GTA5, and unannotated RGB images from target domain(s) (blue), here from IDD ('target 1') and Cityscapes ('target T'). (*Bottom*) Test-time segmentation is on new images from the target domains, without knowing which domain they stem from.

In our setting with multiple target domains, a simple strategy is to merge all target datasets into a single one and then to utilize an existing single-source single-target UDA framework. The UDA model is trained on the source data $\mathcal{X}_s$ and the aggregated target data $\mathcal{X}_t = \bigcup_{n \in [T]} \mathcal{X}_{t,n}$. However, such a strategy disregards the inherent discrepancy among target domains. As shown in the experiments, this "multi-target baseline" is less effective than the proposed strategies, which explicitly handle inter-target domain shifts. The following sections describe these two novel frameworks.

5.2.2 Multi-Dis.: Multi-Discriminator Framework

Our first strategy for multi-target UDA, called Multi-Dis., relies on two types of discriminators to align each target domain with the source (source-target discriminators) and with other targets (target-target discriminators). Figure 5.3 illustrates this first approach.

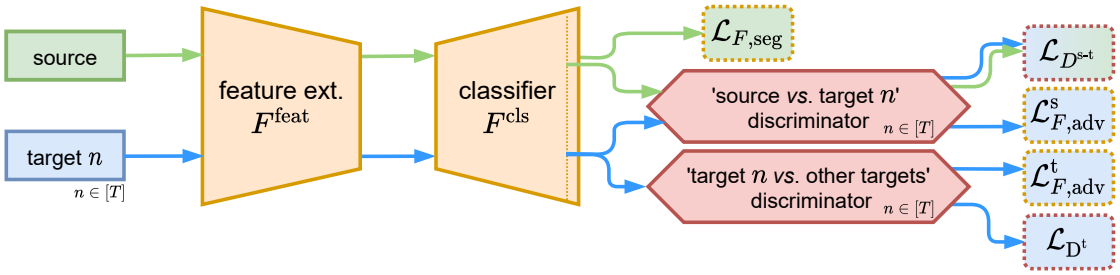


Figure 5.3 – **Multi-discriminator approach to multi-target UDA**. With Multi-Dis., the segmenter is trained against two types of adversaries that discriminate respectively source *vs.* one target and one target *vs.* all other targets. The multiple losses are defined in Equation 2.16, Equation 5.1, Equation 5.2, Equation 5.4 and Equation 5.5.

Source-target adversarial alignment. We introduce a discriminator D_n^{s-t} with parameters ϕ_n^{s-t} for each target domain n. It learns to discriminate $\mathcal{X}_{t,n}$ from the source set $\mathcal{X}_s$. By denoting $\mathcal{L}_{D_n^{s-t}}$ the minimization objective of this discriminator, defined as in Equation 2.15 on domain n, we train these T source-target discriminators with the mean objective:

$$\mathcal{L}_{D^{s-t}}(\phi_{1:T}^{s-t}) = \frac{1}{T}\sum_{n=1}^{T} \mathcal{L}_{D_n^{s-t}}(\phi_n^{s-t}). \tag{5.1}$$

Concurrently, the segmenter F is trained to fool these T discriminators with the adversarial objective:

$$\mathcal{L}^{\text{s}}_{F,\text{adv}}(\theta) = \frac{1}{T}\sum_{n=1}^{T} \frac{1}{|\mathcal{X}_{t,n}|} \sum_{\boldsymbol{x}_t \in \mathcal{X}_{t,n}} \mathcal{L}_{\text{BCE}}(D_n^{s-t}(\boldsymbol{Q}_{\boldsymbol{x}}), 1), \tag{5.2}$$

with the Binary Cross-Entropy (BCE) loss defined in Equation 2.4.

Target-target adversarial alignment. In the above source-target alignment, the source acts as an anchor for each target to "pull" closer to the other targets. However, as this alignment is imperfect, there remain gaps across targets, which we

propose to reduce further by additional target-target alignments. To this end, we introduce for each target domain n a discriminator D_n^t with parameters ϕ_n^t that classifies $\mathfrak{X}_{t,n}$ (class 1) vs all other target domains $\mathfrak{X}_{t,k}$, $k \neq n$ (class 0), resulting in T 1-vs-all discriminators. The target-target discriminator D_n^t is trained by minimizing the loss

$$\mathcal{L}_{D_n^t}(\phi_n^t) = \frac{1}{|\mathcal{X}_{t,n}|} \sum_{\boldsymbol{x}_t \in \mathcal{X}_{t,n}} \mathcal{L}_{\text{BCE}}(D_n^t(\boldsymbol{Q}_{\boldsymbol{x}_t}), 1) + \frac{1}{\sum\limits_{k \neq n} |\mathcal{X}_{t,k}|} \sum_{\boldsymbol{x}_t \in \bigcup\limits_{k \neq n} \mathcal{X}_{t,k}} \mathcal{L}_{\text{BCE}}(D_n^t(\boldsymbol{Q}_{\boldsymbol{x}_t}), 0). \tag{5.3}$$

The collective objective of all target-target discriminators now reads:

$$\mathcal{L}_{D^t}(\phi_{1:T}^t) = \frac{1}{T} \sum_{n=1}^{T} \mathcal{L}_{D_n^t}(\phi_n^t). \tag{5.4}$$

The segmenter F tries to fool all the target-target discriminators by minimizing the adversarial loss:

$$\mathcal{L}_{F,\text{adv}}^t(\theta) = \frac{1}{T} \sum_{n=1}^{T} \frac{1}{\sum\limits_{k \neq n} |\mathcal{X}_{t,k}|} \sum_{\boldsymbol{x}_t \in \bigcup\limits_{k \neq n} \mathcal{X}_{t,k}} \mathcal{L}_{\text{BCE}}(D_n^t(\boldsymbol{Q}_{\boldsymbol{x}_t}), 1). \tag{5.5}$$

To sum up, the segmenter F is trained by minimizing over θ the objective:

$$\mathcal{L}_F = \mathcal{L}_{F,\text{seg}} + \lambda_{\text{adv}}^{\text{s}} \mathcal{L}_{F,\text{adv}}^{\text{s}} + \lambda_{\text{adv}}^t \mathcal{L}_{F,\text{adv}}^t, \tag{5.6}$$

with weights $\lambda_{\text{adv}}^{\text{s}}$ and λ_{adv}^t to balance the adversarial terms and $\mathcal{L}_{F,\text{seg}}$ defined in Equation 2.16.

5.2.3 MTKT : Multi-Target Knowledge Transfer Framework

The main driving force in prediction-level adversarial approaches (Tsai et al. 2018; Vu et al. 2019a) is the adjustment of the decision boundaries. Alignment in feature space then follows to comply with adjusted boundaries. We thus stress the importance of classifier design in the multi-target UDA scenario. In our Multi-Dis. approach, one classifier simultaneously handles multiple domain shifts, either source-target or target-target. The main challenge is the instability of adversarial training, which is amplified if several adversarial losses are jointly minimized. Such an issue is particularly problematic in the early training phase when most target predictions are very noisy. To address this challenge, we propose the MTKT framework, with a novel network design and learning scheme which do not rely

on the joint minimization of multiple adversarial losses over the same classifier module, hopefully reducing the instability of the training. Figure 5.4 shows the MTKT architecture.

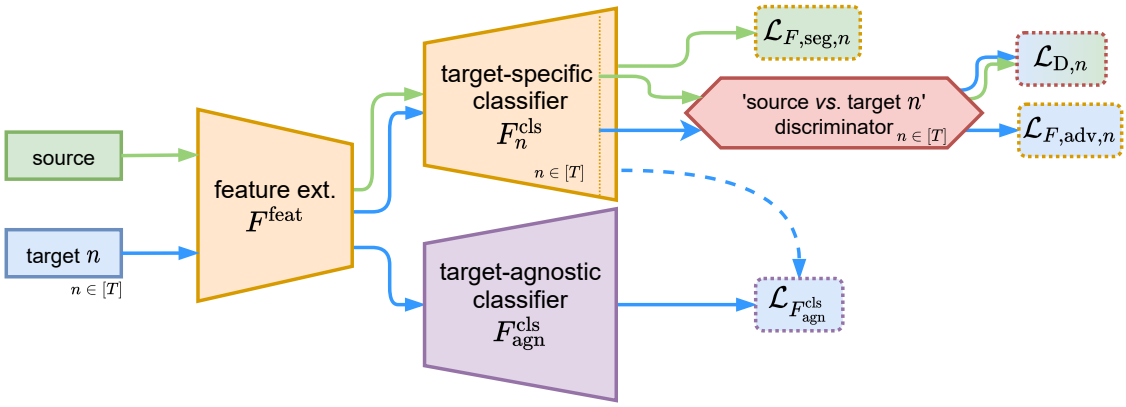


Figure 5.4 – **Multi-target knowledge transfer approach to multi-target UDA.** With MTKT, a set of target-specific segmenters is first trained adversarially. Their knowledge is then jointly distilled to the target-agnostic segmenter whose loss defined in Equation 5.8 is not back-propagated into the target-specific branches (as indicated by the dotted arrow).

The classification part of the network is first re-designed with T *target-specific* instrumental classifiers, $F_n^{\text{cls}},\ n \in [T]$, based on the same feature extractor F^{feat}, each handling one specific source-target domain shift. Such an architecture allows separate output-space adversarial alignment for each specific source-target pair, alleviating the instability problem. For each target-specific classifier F_n^{cls}, we introduce a domain discriminator D_n^{t} as to classify source vs target n. The training objectives are similar to those used in single-target models (Equation 2.15 and Equation 2.16).

We then introduce a *target-agnostic* classification branch $F_{\text{agn}}^{\text{cls}}$ that fuses all the knowledge transferred from the T target-specific classifiers. This target-agnostic classifier is the final product of the approach, i.e., the one used at test time when domain knowledge is not available.

The knowledge from the T "teachers" is transferred to the target-agnostic "student" via minimizing the Kullback-Leibler (KL) divergence (Hinton et al. 2015) between teachers' and student's predictions on target domains. In details, for a given sample $\boldsymbol{x}_t \in \mathcal{X}_{t,n}$, we compute the KL loss

$$\mathcal{L}_{\text{KL},n}(\boldsymbol{x}_t) = \sum_{h=1}^{H}\sum_{w=1}^{W}\sum_{c=1}^{C} \boldsymbol{P}_{n,\boldsymbol{x}_t}[h,w,c] \log \frac{\boldsymbol{P}_{n,\boldsymbol{x}_t}[h,w,c]}{\boldsymbol{P}_{\boldsymbol{x}_t}[h,w,c]}, \tag{5.7}$$

where $\boldsymbol{P}_{n,\boldsymbol{x}_t}$ and $\boldsymbol{P}_{\boldsymbol{x}_t}$ are soft-segmentation predictions coming from the target-specific F_n^{cls} and the target-agnostic $F_{\text{agn}}^{\text{cls}}$ respectively. The minimization objective of the target-agnostic classifier $F_{\text{agn}}^{\text{cls}}$ over the segmenter's parameters (including feature extractor's) then reads:

$$\mathcal{L}_{F_{\text{agn}}^{\text{cls}}}(\theta) = \frac{1}{T}\sum_{n=1}^{T}\frac{1}{|\mathcal{X}_{t,n}|}\sum_{\boldsymbol{x}_t\in\mathcal{X}_{t,n}}\mathcal{L}_{\text{KL},n}(\boldsymbol{x}_t). \tag{5.8}$$

Minimizing KL losses helps $F_{\text{agn}}^{\text{cls}}$ adjust its decision boundaries toward good behavior in all T target domains. As the KL loss is back-propagated through the feature extractor, such an adjustment results in implicit alignment in target feature space, which overall mitigates the distribution shifts between the T domains.

Discussion. Unlike Multi-Dis., the multi-teacher/single-student mechanism in MTKT avoids direct alignment between unlabeled parts. The target-agnostic classifier is encouraged to adjust its decision boundaries to favor all the target-specific teachers, thus helping cross-target alignment.

Although we build our frameworks over output-space alignment (Vu et al. 2019a; Tsai et al. 2018), note that they could be adapted to other adversarial feature-alignment methods (Hoffman et al. 2016). Moreover, orthogonal approaches like pseudo-labeling (Chapter 4) can also be included in our frameworks, and we show some experiments with such addition in Section 5.2.4.4.

5.2.4 Experimental Results

5.2.4.1 Datasets

We build our experiments on four urban driving datasets, one being synthetic and the three other being recorded in various geographic locations: GTA5, Cityscapes, IDD, Mapillary Vistas. The datasets are described in Section 2.3.2. Though all containing urban scenes, the four datasets have different labeling policies and semantic granularity.

We follow the protocol used in (K.-H. Lee et al. 2019; Vu et al. 2019b) and standardize the label set with 7 super classes, common to all four datasets: *flat*, *construction*, *object*, *nature*, *sky*, *human* and *vehicle*. The mapping from original classes to these super classes is given in Table 2.2, Table 2.1, Table 2.5 and Table 2.4.

When Cityscapes, IDD, or Mapillary are used as target domains, only unlabeled images from them are used for training, by definition of the UDA problem.

5.2.4.2 Implementation Details

The experiments are conducted with PyTorch (Paszke et al. 2017). The adversarial framework is based on Adversarial Entropy Minimization (AdvEnt)'s published code. [1] The semantic segmentation model is DeepLab-V2 (L.-C. Chen et al. 2018a), built upon the ResNet-101 (K. He et al. 2016) backbone initialized with ImageNet (Deng et al. 2009) pre-trained weights. The segmenters are trained by Stochastic Gradient Descent (SGD) (Bottou 2010) with learning rate 2.5×10^{-4}, momentum 0.9 and weight decay 10^{-4}. We train the discriminators using an Adam optimizer (Diederik P. Kingma 2015) with learning rate 10^{-4}. All experiments are conducted at the 640×320 resolution.

For MTKT, we "warm up" the target-specific branches for 20,000 iterations before training the target-agnostic branch. The warm-up step avoids the distillation of noisy target predictions in the early phase, which helps stabilize target-agnostic training.

5.2.4.3 Main results

We consider four setups, varying the type of domain-shift ('syn-2-real' or 'city-2-city') or the number T of targets (two to three domains). For multi-target performance, we report the mean Intersection over Union (mIoU) averaged over the target domains. Using the average helps mitigate the potential bias caused by target evaluation sets with substantially different sizes.

GTA5 → Cityscapes + Mapillary. Table 5.1 reports segmentation results on the two target validation sets of Cityscapes and Mapillary; GTA5 is the source domain in this setup. For comparison, we consider the single-target AdvEnt models, i.e. trained on either Cityscapes or Mapillary unlabeled images. We also have the multi-target AdvEnt model, denoted as 'Multi-Target Baseline' in Table 5.1, which is trained on the merging of the two targets. For all models, including the single-target ones, we report both per-target and average mIoUs. The two rows marked with '(*)' indicate results of the single-target models on the same domains used for training, regarded as per-target baselines.

Single-target baselines achieve worse average mIoU than those trained on both domains, which indicates the benefit of having access to diverse data from multiple domains during training. Our proposed approaches outperform the multi-target baseline with mIoU gains of $+0.6\%$ for multi-discriminator and $+2.0\%$ for MTKT. Looking closer at the per-target results, we observe unfavorable performance if one directly transfers single-target models to a new domain. Indeed,

1. https://github.com/valeoai/ADVENT

Table 5.1 – **Multi-target semantic segmentation performance on GTA5 → Cityscapes + Mapillary.** Per-class Intersection over Union (IoU) (%), per-domain mIoU ('mIoU') and mIoU averaged over domains ('mIoU Avg.'); mIoU gain (green) or loss (red) w.r.t. corresponding per-target baselines (marked as '*'); 'train': indication of the unlabeled target data used for training.

GTA5 → Cityscapes + Mapillary											
Method	Target	Train	flat	constr.	object	nature	sky	human	vehicle	mIoU-7	mIoU-7 Avg.
Single-Target Baselines (Vu et al. 2019a)	Cityscapes	✓	93.5	80.5	26.0	78.5	78.5	55.1	76.4	69.8 (*)	66.6
	Mapillary	-	86.8	69.0	30.2	71.2	91.5	35.3	59.5	63.4$_{\downarrow 6.2}$	
	Cityscapes	-	89.3	79.3	19.5	76.9	84.6	47.7	63.0	65.8$_{\downarrow 4.0}$	67.7
	Mapillary	✓	89.5	72.6	31.0	75.3	94.1	50.7	73.8	69.6 (*)	
Multi-Target Baseline (Vu et al. 2019a)	Cityscapes	✓	93.1	80.5	24.0	77.9	81.0	52.5	75.0	69.1$_{\downarrow 0.7}$	68.9
	Mapillary	✓	90.0	71.3	31.1	73.0	92.6	46.6	76.6	68.7$_{\downarrow 0.9}$	
Multi-Dis.	Cityscapes	✓	94.5	80.8	22.2	79.2	82.1	47.0	79.0	69.3$_{\downarrow 0.5}$	69.5
	Mapillary	✓	89.4	71.2	29.5	76.2	93.6	50.4	78.3	69.8$_{\uparrow 0.2}$	
MTKT	Cityscapes	✓	95.0	81.6	23.6	80.1	83.6	53.7	79.8	**71.1**$_{\uparrow 1.3}$	**70.9**
	Mapillary	✓	90.6	73.3	31.0	75.3	94.5	52.2	79.8	**70.8**$_{\uparrow 1.2}$	

testing the Cityscapes-only model on Mapillary results in a drop of -6.2% mIoU compared to the reference performance, and a similar drastic drop is seen for Mapillary-only model on Cityscapes. Especially we notice important degradation on safety-critical classes like *human* or *vehicle* using those single-target models. The Multi-Dis. model achieves comparable mIoUs as the per-target baselines. The MTKT model improves over the per-target baselines by a significant margin, i.e. $+1.3\%$ on Cityscapes and $+1.2\%$ on Mapillary. Such results highlight the merit of the proposed strategies, especially MTKT. Note that adding adversarial training on the target-agnostic branch of MTKT hinders the alignment effect, reducing the performance by 0.9% mIoU Avg.

GTA5 → Cityscapes + IDD. We experiment with another synthetic-to-real setup in which the two target datasets have noticeably different landscapes, i.e. European cities in Cityscapes and Indian ones in IDD. Results are reported in Table 5.2. Here also, multi-target models outperform single-target ones. In this setup, the performance of Multi-Dis. is comparable to the multi-target baseline's. We conjecture that the complex and unstable optimization problem in the Multi-Dis. framework makes it difficult to achieve good alignment across targets, especially when the two target domains are more noticeably different. With a dedicated architecture and learning scheme that alleviates such an optimization issue, the MTKT model achieves the best results, in terms of both per-target and average mIoUs.

Table 5.2 – **Multi-target semantic segmentation performance on GTA5 → Cityscapes + IDD.** Organization as in Table 5.2.

GTA5 → Cityscapes + IDD											
Method	Target	Train	flat	constr.	object	nature	sky	human	vehicle	mIoU-7	mIoU-7 Avg.
Single-Target Baselines (Vu et al. 2019a)	Cityscapes	✓	93.5	80.5	26.0	78.5	78.5	55.1	76.4	69.8 (*)	66.5
	IDD	-	91.3	52.3	13.3	76.1	88.7	46.7	74.8	$63.3_{\downarrow 1.8}$	
	Cityscapes	-	78.6	79.2	24.8	77.6	83.6	48.7	44.8	$62.5_{\downarrow 7.3}$	63.8
	IDD	✓	91.2	53.1	16.0	78.2	90.7	47.9	78.9	65.1 (*)	
Multi-Target Baseline (Vu et al. 2019a)	Cityscapes	✓	93.9	80.2	26.2	79.0	80.5	52.5	78.0	$70.0_{\uparrow 0.2}$	67.4
	IDD	✓	91.8	54.5	14.4	76.8	90.3	47.5	78.3	$64.8_{\downarrow 0.3}$	
Multi-Dis.	Cityscapes	✓	94.3	80.7	20.9	79.3	82.6	48.5	76.2	$68.9_{\downarrow 0.9}$	67.3
	IDD	✓	92.3	55.0	12.2	77.7	92.4	51.0	80.2	$65.7_{\uparrow 0.6}$	
MTKT	Cityscapes	✓	94.5	82.0	23.7	80.1	84.0	51.0	77.6	**70.4**$_{\uparrow 0.5}$	**68.2**
	IDD	✓	91.4	56.6	13.2	77.3	91.4	51.4	79.9	**65.9**$_{\uparrow 0.8}$	

We visualize some qualitative results in Figure 5.5.

GTA5 → Cityscapes + Mapillary + IDD. We consider a more challenging setup involving three target domains – Cityscapes, Mapillary and IDD – and show results in Table 5.3. With more target domains, the same conclusions hold. In terms of average mIoU, the Multi-Dis. model marginally improves over the multi-target baseline. The MTKT model significantly outperforms all other models with 69.1% mIoU Avg. Moreover, when compared to the per-target baselines, MTKT is the only model to show improvement on every target domain.

Cityscapes → Mapillary + IDD. Finally, we experiment on a realistic city-to-city setup with Cityscapes as the source and Mapillary and IDD as target domains. The results are shown in Table 5.4. Interestingly, on Mapillary, the single-target model trained on IDD achieves better results than the one trained only on Mapillary. We conjecture that the domain gap between Cityscapes and Mapillary is less than the one between Cityscapes and IDD; The extra data diversity coming from IDD improves the single-target IDD-only model generalization and helps mitigate the small Cityscapes-Mapillary domain gap. Another observation is that the IDD-only model outperforms the multi-target baseline. This indicates the disadvantage of the naive dataset merging strategy: not only complementary signals but also conflicting/negative ones get transferred. The two proposed models outperform the multi-target baseline; MTKT obtains the best performance overall. Again in this realistic setup, we showcase the advantages of our methods, especially the MTKT model.

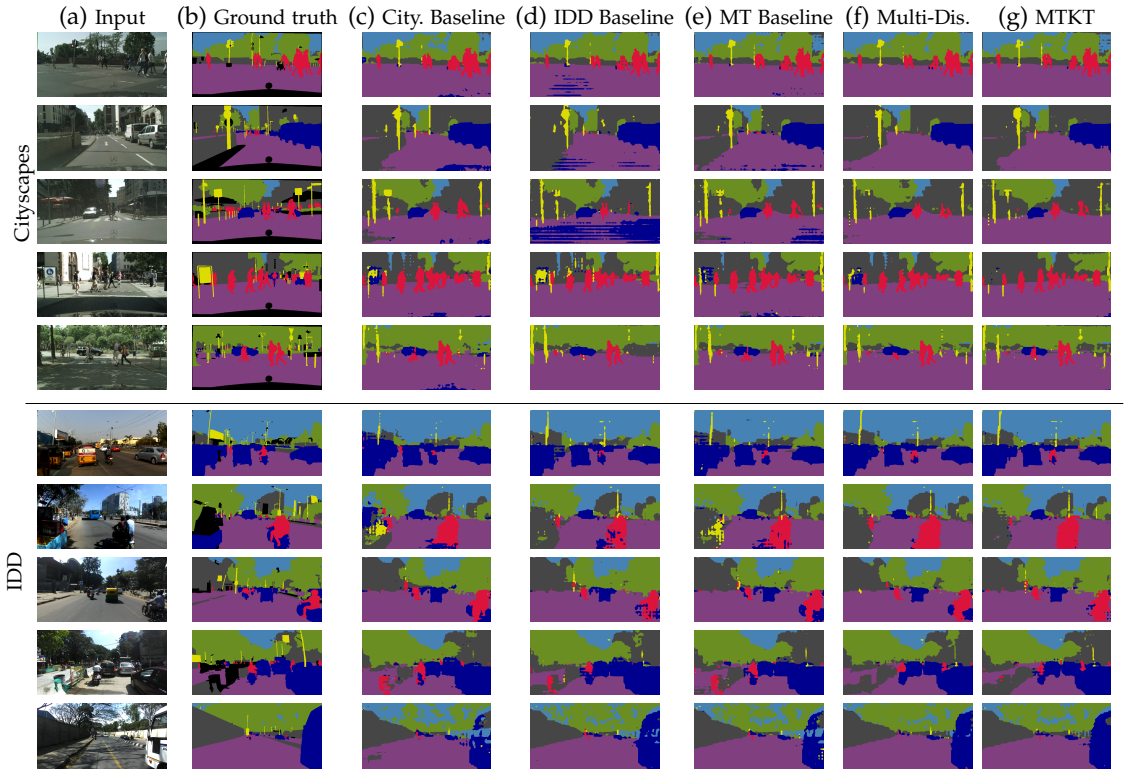


Figure 5.5 – **Multi-target qualitative results in the GTA5 → Cityscapes + IDD setup.** (a) Test images from Cityscapes and IDD; (b) Ground-truth segmentation maps; Results of (c) single-target baseline trained on Cityscapes target, (d) single-target baseline trained on IDD target, (e) multi-target baseline, (f) proposed Multi-Dis. and (g) proposed MTKT. Both proposed multi-target frameworks give overall cleaner segmentation maps compared to the baselines.

Conclusions. These four sets of experiments demonstrate that the proposed multi-target frameworks consistently deliver competitive performance on the multiple target domains on which they are trained. MTKT always gives the best performance, both in per-target and average mIoUs, compared to the baselines and the Multi-Dis. model. Note that our models are compatible with techniques such as image translation (Hoffman et al. 2018; Wu et al. 2018; Y. Yang and Soatto 2020) or ST with pseudo-labeling (Chapter 4), from which they could benefit. In particular, we show next with additional experiments how to use ESL pseudo-labeling (Section 4.2) with MTKT.

5.2.4.4 Combining with Self-Training.

We proposed in Chapter 4 some extensions of the AdvEnt method used here as UDA strategy. In particular, ST can easily be combined with our multi-target

Table 5.3 – **Multi-target segmentation performance on GTA5 → Cityscapes + Mapillary + IDD ($T = 3$).** Organization as in Table 5.1.

GTA5 → Cityscapes + Mapillary + IDD											
Method	Target	Train	flat	constr.	object	nature	sky	human	vehicle	mIoU-7	mIoU-7 Avg.
Single-Target Baselines (Vu et al. 2019a)	Cityscapes	✓	93.5	80.5	26.0	78.5	78.5	55.1	76.4	69.8 (*)	65.5
	Mapillary	-	86.8	69.0	30.2	71.2	91.5	35.3	59.5	63.3$_{\downarrow 6.3}$	
	IDD	-	91.3	52.3	13.3	76.1	88.7	46.7	74.8	63.3$_{\downarrow 1.8}$	
	Cityscapes	-	89.3	79.3	19.5	76.9	84.6	47.7	63.0	65.8$_{\downarrow 4.0}$	66.7
	Mapillary	✓	89.5	72.6	31.0	75.3	94.1	50.7	73.8	69.6 (*)	
	IDD	-	91.7	54.3	13.0	77.3	92.3	47.4	76.8	64.7$_{\downarrow 0.4}$	
	Cityscapes	-	78.6	79.2	24.8	77.6	83.6	48.7	44.8	62.5$_{\downarrow 7.3}$	65.5
	Mapillary	-	88.5	71.2	32.4	72.8	92.8	51.3	73.7	69.0$_{\downarrow 0.6}$	
	IDD	✓	91.2	53.1	16.0	78.2	90.7	47.9	78.9	65.1 (*)	
Multi-Target Baseline (Vu et al. 2019a)	Cityscapes	✓	93.6	80.6	26.4	78.1	81.5	51.9	76.4	69.8$_{-}$	67.8
	Mapillary	✓	89.2	72.4	32.4	73.0	92.7	41.6	74.9	68.0$_{\downarrow 1.6}$	
	IDD	✓	92.0	54.6	15.7	77.2	90.5	50.8	78.6	65.6$_{\uparrow 0.5}$	
Multi-Dis.	Cityscapes	✓	94.6	80.0	20.6	79.3	84.1	44.6	78.2	68.8$_{\downarrow 1.0}$	68.2
	Mapillary	✓	89.0	72.5	29.3	75.5	94.7	50.3	78.9	70.0$_{\uparrow 0.4}$	
	IDD	✓	91.6	54.2	13.1	78.4	93.1	49.6	80.3	65.8$_{\uparrow 0.7}$	
MTKT	Cityscapes	✓	94.6	80.7	23.8	79.0	84.5	51.0	79.2	**70.4**$_{\uparrow 0.6}$	**69.1**
	Mapillary	✓	90.5	73.7	32.5	75.5	94.3	51.2	80.2	**71.1**$_{\uparrow 1.5}$	
	IDD	✓	91.7	55.6	14.5	78.0	92.6	49.8	79.4	**65.9**$_{\uparrow 0.8}$	

frameworks. Taking for instance ESL (Section 4.2), we consider three ways to adapt its pseudo-labeling strategy to the MTKT architecture. In all of them, we collect pseudo-labels in each target domain using the corresponding target-specific classifier and use them as additional self-supervision for these target-specific heads; In the second method we also use these pseudo-labels to restrict the back-propagation of the KL losses to pixels that are correctly classified according to these pseudo-labels; In the third method, they are also used to refine the target-agnostic classifier. We report in Table 5.5 the results of the models trained with these three pseudo-labeling-based refinement strategies on GTA5 → Cityscapes + IDD and compare them to the baseline trained with ESL. The three ways of extending MTKT with pseudo-labeling result in similar performance gains of at least $+1.6\%$ mIoU Avg. This demonstrates that knowledge transfer is complementary to pseudo-labeling. Moreover, MTKT with ESL outperforms the baseline with ESL by $+1.7\%$ mIoU Avg.

5.2.4.5 Direct Transfer to a New Dataset.

We consider a direct transfer setup: the models see no images from the test domain during training. This experiment highlights how well the models can generalize to new previously-unseen domains. Table 5.6 reports the results of

Table 5.4 – **Multi-target segmentation performance of city-2-city multi-target UDA on Cityscapes → Mapillary + IDD.** Organization as in Tab. 5.1.

Cityscapes → Mapillary + IDD											
Method	Target	Train	flat	constr.	object	nature	sky	human	vehicle	mIoU-7	mIoU-7 Avg.
Single-Target Baselines (Vu et al. 2019a)	Mapillary	✓	87.4	65.9	28.2	72.8	92.1	46.9	72.7	66.6 (*)	65.8
	IDD	-	91.8	52.2	15.9	80.2	91.1	45.7	77.6	65.0$_{\downarrow 2.3}$	
	Mapillary	-	88.2	70.0	28.5	75.4	93.6	49.1	76.7	68.8$_{\uparrow 2.2}$	68.0
	IDD	✓	93.2	53.4	16.5	83.4	93.4	51.4	79.5	67.3 (*)	
Multi-Target Baseline (Vu et al. 2019a)	Mapillary	✓	87.7	65.9	29.0	73.2	91.5	47.9	75.7	67.3$_{\uparrow 0.7}$	67.0
	IDD	✓	93.3	53.0	17.2	82.8	92.2	49.3	79.6	66.8$_{\downarrow 0.5}$	
Multi-Dis.	Mapillary	✓	88.6	70.9	29.6	75.8	94.7	49.2	76.1	69.3$_{\uparrow 2.7}$	67.9
	IDD	✓	92.8	52.8	17.0	83.1	94.2	48.5	77.4	66.5$_{\downarrow 0.8}$	
MTKT	Mapillary	✓	88.3	70.4	31.6	75.9	94.4	50.9	77.0	**69.8**$_{\uparrow 3.2}$	**69.0**
	IDD	✓	93.6	54.9	18.6	84.0	94.5	53.4	79.2	**68.3**$_{\uparrow 1.0}$	

Table 5.5 – **Additional impact of pseudo-labeling on multi-target UDA.** Trained models are refined with one step of ESL (Section 4.2). For MTKT, pseudo-labels are extracted for each target domain with the associated teacher head, and used either (1) to refine this head only, (2) to refine this head and to back-propagate KL-loss only on the pixels with predictions compliant with pseudo-labels or (3) to refine both this head and the target-agnostic model.

GTA5 → Cityscapes + IDD						
Method	M-T base.	M-T base. + PL	MTKT	MTKT + PL (1)	MTKT + PL (2)	MTKT + PL (3)
mIoU-7 Avg.	67.4	68.9	68.2	69.8	69.7	**69.9**

such a direct transfer to a new dataset in different setups. The models are trained on GTA5 → Cityscapes + IDD (resp. on GTA5 → Cityscapes + Mapillary) and tested on Mapillary (resp. IDD). On both setups, MTKT shows better performance in terms of mIoU compared to the baselines on the new domain. In the first one, in particular, with Mapillary as the new test domain, MTKT outperforms the multi-target baseline by $+3.9\%$. What is particularly noticeable in this setup is the performance on the *human* class: while we observe an IoU of around 50% in the main results on domain adaptation to Mapillary (e.g. in Table 5.1), the direct transfer results of the multi-target baseline and of Multi-Dis. drop under 38% on this class; Differently, MTKT manages to get similar performance with 52.8% IoU on *human*. This experiment hints at the ability of MTKT to better generalize to new unseen domains.

Table 5.6 – **Direct transfer to new target.** Multi-target models are tested on a new unseen domain: (Top) GTA5 → Cityscapes + IDD, tested on Mapillary; (Bottom) GTA5 → Cityscapes + Mapillary, tested on IDD.

setup	Method	Test set	flat	constr.	object	nature	sky	human	vehicle	mIoU-7
G→C+I	M-T Baseline	Mapillary	88.4	71.0	**31.0**	72.4	92.0	37.4	74.7	66.7
	Multi-Dis.		89.2	72.1	21.7	73.8	**94.0**	34.8	75.9	65.9
	MTKT		**89.8**	**74.0**	30.4	**74.1**	93.6	**52.6**	**79.4**	**70.6**
G→C+M	M-T Baseline	IDD	**91.6**	54.7	**13.9**	76.5	90.9	48.3	77.5	64.8
	Multi-Dis.		91.2	54.6	12.9	**77.7**	**92.5**	50.3	78.6	**65.4**
	MTKT		91.5	**56.1**	12.3	76.1	90.9	**51.4**	**79.2**	**65.4**

Limits. Nonetheless, multi-target settings have their drawbacks, especially in the context of urban scene segmentation. As already discussed in Chapter 2, UDA for semantic segmentation is an expensive task in terms of computing power, and specifically requires a lot of Graphics Processing Unit (GPU) memory. This becomes exponentially more problematic when considering more than two domains.

Moreover, while showing significant improvements on multi-target settings, the methods proposed in this section rely on the simultaneous availability of all the target domain data during training. This limits the practical usage of multi-target UDA strategies as one may not have access to all the target domains at the same time due to privacy or may want to improve a pre-trained model on a new target domain without re-training on the previous target domains.

In these scenarios, a continual learning perspective in which new target domains are discovered sequentially would be more relevant than a multi-target setting.

5.3 Continual Target Knowledge Transfer for Continual Unsupervised Domain Adaptation

5.3.1 Continual Unsupervised Domain Adaptation

Continual Learning. The task of continual learning aims at learning a constantly changing distribution. A naive mitigation is to re-train the model from scratch on the updated dataset. However, it assumes that previous data is kept, which is

often unfeasible for multiple reasons, including privacy. Thus, a continual model has to learn solely on the new data while remembering the previous data. As a result, the model faces the challenge of "catastrophic forgetting" (Robins 1995; Thrun 1998; French 1999) where the performance on previous samples drops. This problem can be mitigated by different approaches: rehearsal of a limited amount of previous data (Rebuffi et al. 2017; Hayes et al. 2020) can reduce forgetting, but is memory-costly for high-resolution images required for semantic segmentation (Douillard et al. 2021b). A second approach is to constrain the new model to be "similar" to the previous model. This similarity can be defined on the weights (Kirkpatrick et al. 2017), the gradients (Lopez-Paz and Ranzato 2017), or even the probabilities (Z. Li and Hoiem 2016) and the intermediary features (Douillard et al. 2020). More recently, continual models were adapted for semantic segmentation (Michieli and Zanuttigh 2019; Cermelli et al. 2020; Douillard et al. 2021a) with success, but they restricted themselves to supervised tasks on a single dataset and not unsupervised adaptation across multiple domains.

Problem Setting. As in Section 5.2.1, $T \geq 2$ distinct target domains $\mathfrak{X}_{t,n}, n \in [T]$, must be jointly handled by the model. They are represented by unlabeled training sets $\mathcal{X}_{t,n}$, $n \in [T]$. As in standard UDA settings, we assume that the annotated training examples $(\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{X}_s \times \mathcal{Y}_s$ stem from a single source domain, a specific synthetic environment for instance.

This setting differs from the multi-target unsupervised domain adaptation setting of Section 5.2.1: we assume here that the different target domains can only be accessed sequentially and one at a time during training. More precisely, during training, the model only has access to a single target dataset $\mathcal{X}_{t,n}$ with $n \in [T]$ and cannot access ever again the target datasets $\mathcal{X}_{t,k}, k \in [n-1]$ it has previously learned. Nevertheless, we consider that the source domain dataset $(\mathcal{X}_s, \mathcal{Y}_s)$ is always accessible during training.

However, the objective is the same as the multi-target setting: train a single segmenter F that achieves equally good results on all target-domain test sets. Also, while the target domain of origin is known for all unlabeled training examples, we assume, as in multi-target classification approaches of (Gholami et al. 2020; Yu et al. 2018), that this information is not accessible at test time.

Overall, this setting brings a new challenge compared to the multi-target one: when training on new target domains, the model must not forget the old target domains it has trained before.

Revisiting Adversarial Unsupervised Domain Adaptation Approach. As in Section 5.2.1, we adapt the training procedure of state-of-the-art UDA approaches like (Tsai et al. 2018) or (Vu et al. 2019a) to this new setting.

In continual unsupervised domain adaptation, the model always has access during training to a single source domain and a single target domain. Standard UDA approaches can easily be adapted to this setting: at each iteration $n \in [T]$, the model is trained on the source dataset $\mathcal{X}_s$ and the current target dataset $\mathcal{X}_{t,n}$, initialized from the model trained at the previous iteration. This way, the model is trained in a UDA fashion sequentially on all the target domains. However, this "continual baseline" doesn't directly tackle catastrophic forgetting of old domains. The following section describes our strategy to explicitly prevent catastrophic forgetting.

5.3.2 CTKT: Continual Target Knowledge Transfer Framework

The proposed approach extends the MTKT framework presented in Section 5.2.3 to the continual setting. The proposed CTKT framework is described in Figure 5.6.

As in MTKT, the classification part of the network is designed with multiple branches. At iteration $n \geq 2$, the segmentation model $F_{(1:n)}$, composed of a feature extractor $F^{\text{feat}}_{(1:n)}$ and a pixel-wise classifier $F^{\text{cls}}_{(1:n)}$, is trained on the source domain $\mathfrak{X}_s$ and the target domain $\mathfrak{X}_{t,n}$ to perform on the target domains 1 to n. In this step n, the network has one additional *target-specific* instrumental classifier F^{cls}_n based on the feature extractor $F^{\text{feat}}_{(1:n)}$. This classifier F^{cls}_n handles the specific source-target n domain shift. This separated branch allows a proper output-space adversarial alignment between the source domain and the target domain n. This classification head is associated with a domain discriminator D to classify source vs. target n. The training objectives are similar to those used in single-target models (Equation 2.15 and Equation 2.16).

Target-agnostic architecture. The *target-agnostic* head $F^{\text{cls}}_{(1:n)}$, which is eventually kept as classification head of the model, is trained to perform on all the target domains from 1 to n. The knowledge from the target-specific branch is transferred to the target-agnostic branch via a teacher-student strategy by minimizing the KL divergence between the predictions of the two classification heads on the target domain n. For a given sample $\boldsymbol{x}_t \in \mathcal{X}_{t,n}$, we compute the KL loss

$$\mathcal{L}_{\text{KL},n}(\boldsymbol{x}_t) = \sum_{h=1}^{H}\sum_{w=1}^{W}\sum_{c=1}^{C} \boldsymbol{P}_{n,\boldsymbol{x}_t}[h,w,c] \log \frac{\boldsymbol{P}_{n,\boldsymbol{x}_t}[h,w,c]}{\boldsymbol{P}_{(1:n),\boldsymbol{x}_t}[h,w,c]}, \tag{5.9}$$

where $\boldsymbol{P}_{n,\boldsymbol{x}_t}$ and $\boldsymbol{P}_{(1:n),\boldsymbol{x}_t}$ are soft-segmentation predictions coming from the target-specific F^{cls}_n and the target-agnostic $F^{\text{cls}}_{(1:n)}$ respectively.

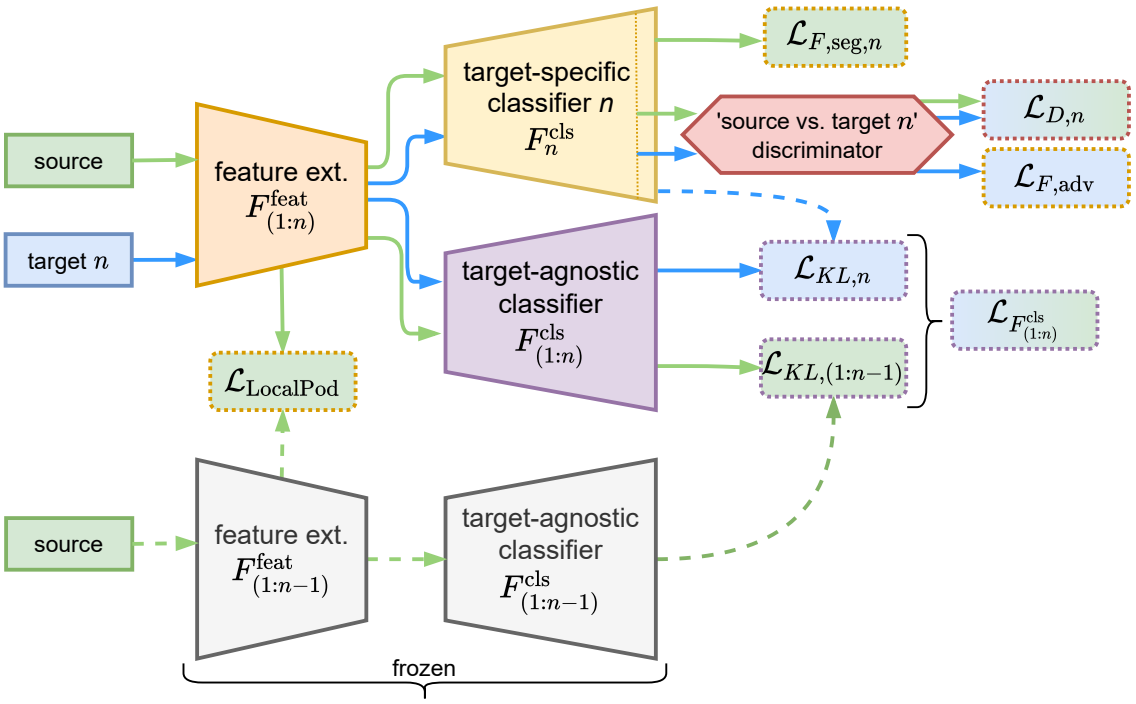


Figure 5.6 – **Continual-target knowledge transfer approach to continual UDA.** When discovering a new target domain n, CTKT learns its model $(F^{\text{feat}}_{(1:n)}, F^{\text{cls}}_{(1:n)})$ using knowledge transfer from both a target-specific segmenter F^{cls}_{n} trained adversarially for this target domain, and the frozen segmentation model from the previous training step $(F^{\text{feat}}_{(1:n-1)}, F^{\text{cls}}_{(1:n-1)})$. In combination with this architectural design, the training losses are indicated and further developed in the text. In particular, the Local POD loss $\mathcal{L}_{\text{LocalPod}}$ is introduced to further prevent catastrophic forgetting. The losses are not back-propagated into dotted arrows.

Preventing catastrophic forgetting. Furthermore, the model must not forget what it has learned in the previous training iterations about the other target domains. Without proper constraint, the model may be subject to catastrophic forgetting, hindering its performance on previous target domains. Thus, additional losses based on the model's previous iteration are considered when training the model on the new target domain. The aim of these losses is to make sure that the new model keeps similar features to its previous iteration in order to keep similar results on the previous target domains, on which the old model was performing well.

Ideally, one would want to use images from the previous target domains to get the features and results of the old model and make sure they are as close as possible as those of the currently trained model. However, in this continual setting,

one cannot access the previous target domain images anymore. Nevertheless, since the models are trained with adversarial alignment, we assume that the features on the source domain are close enough to the features of the previous target domains to use them as proxy. Under this assumption, we contrain source domain features of the current model with source domain features of the previous model.

Practically, we perform a similar knowledge transfer with KL distillation between the previously trained target-agnostic classifier $F^{\text{cls}}_{(1:n-1)}$, supposed to perform on targets 1 to $n-1$, and the currently trained target-agnostic classifier $F^{\text{cls}}_{(1:n)}$ we want to perform on all targets 1 to n. Without access to images from domains 1 to $n-1$, we use source images as proxy. For a given source sample $\boldsymbol{x}_s \in \mathcal{X}_s$, we compute the KL loss

$$\mathcal{L}_{\text{KL},(1:n-1)}(\boldsymbol{x}_s) = \sum_{h=1}^{H}\sum_{w=1}^{W}\sum_{c=1}^{C} \boldsymbol{P}_{(1:n-1),\boldsymbol{x}_s}[h,w,c] \log \frac{\boldsymbol{P}_{(1:n-1),\boldsymbol{x}_s}[h,w,c]}{\boldsymbol{P}_{(1:n),\boldsymbol{x}_s}[h,w,c]}, \tag{5.10}$$

where $\boldsymbol{P}_{(1:n-1),\boldsymbol{x}_s}$ and $\boldsymbol{P}_{(1:n),\boldsymbol{x}_s}$ are soft-segmentation predictions coming from the previous iteration of the target-agnostic classifier $F^{\text{cls}}_{(1:n-1)}$, previously trained for targets 1 to $n-1$ and now frozen, and the currently trained target-agnostic $F^{\text{cls}}_{(1:n)}$, respectively.

With the addition of this loss, the minimization objective of the target-agnostic classifier $F^{\text{cls}}_{(1:n)}$ over the segmenter's parameters (including feature extractor's), or knowledge transfer loss, then reads:

$$\mathcal{L}_{F^{\text{cls}}_{(1:n)}}(\theta) = \frac{1}{|\mathcal{X}_{t,n}|}\sum_{\boldsymbol{x}_t \in \mathcal{X}_{t,n}} \mathcal{L}_{\text{KL},n}(\boldsymbol{x}_t) + \lambda_{\text{prev}} \frac{1}{|\mathcal{X}_s|}\sum_{\boldsymbol{x}_s \in \mathcal{X}_s} \mathcal{L}_{\text{KL},(1:n-1)}(\boldsymbol{x}_s), \tag{5.11}$$

with weight λ_{prev} to balance the knowledge transfer from the previous model.

Along with this knowledge transfer loss, we also want to specifically enforce the current feature extractor F^{feat} to produce features at each layer close to the previous iteration of the feature extractor. This kind of approach helps alleviate the catastrophic forgetting in continual learning problems. Practically, we propose to use the Local POD distillation $\mathcal{L}_{\text{LocalPod}}(\theta^{\text{feat}})$ proposed in (Douillard et al. 2021a). By noting $\psi(\boldsymbol{u})$ the concatenation of width-pooled slices and height-pooled slices over multiple scales of $\boldsymbol{u}$, the Local POD distillation is defined, on each sample $\boldsymbol{x}_s$ of the source dataset $\mathcal{X}_s$, and over each layer activation $F^{\text{feat}\,(l)}_{(1:n)}(\boldsymbol{x}_s)$ and $F^{\text{feat}\,(l)}_{(1:n-1)}(\boldsymbol{x}_s)$, $l \in [L]$, of the the feature extractors $F^{\text{feat}}_{(1:n)}$ and $F^{\text{feat}}_{(1:n-1)}$, by the formula:

$$\mathcal{L}_{\text{LocalPod}}(\theta^{\text{feat}}) = \frac{1}{L}\sum_{l=1}^{L} \left\| \psi(F^{\text{feat}\,(l)}_{(1:n)}) - \psi(F^{\text{feat}\,(l)}_{(1:n-1)}) \right\|^2_F. \tag{5.12}$$

It is a multi-scale pooling distillation method that aims at preserving spatial relationships on the intermediate features. A more in-depth definition of the loss is developed in (Douillard et al. 2021a).

Note that this distillation loss was originally proposed in (Douillard et al. 2021a) in a continual semantic segmentation setting where new classes were added at each iteration while staying in the same domain. The setting considered here is notably different since the classes don't change throughout training, but the model encounters new target domains at each iteration. Furthermore, the target domains are not annotated, while segmentation maps are available for all images during training in the setting of (Douillard et al. 2021a). We use the authors' implementation of the LocalPod in our experiments [2].

Overall, the minimization objective of the semantic segmentation model F over θ can be written as:

$$\mathcal{L}_F = \mathcal{L}_{F_n^{\text{cls}},seg} + \lambda_{\text{adv}}\mathcal{L}_{F_n^{\text{cls}},adv} + \lambda_{\text{kt}}\mathcal{L}_{F_{(1:n)}^{\text{cls}}} + \lambda_{\text{dist}}\mathcal{L}_{\text{LocalPod}}, \tag{5.13}$$

with weights $\lambda_{\text{adv}}, \lambda_{\text{kt}}, \lambda_{\text{dist}}$ balancing adversarial training, knowledge transfer and distillation, respectively.

5.3.3 Experimental Results

5.3.3.1 Datasets

To be able to compare the results between multi-target and continual UDA, we build our experiments on the same datasets as the previous section on multi-target UDA (Section 5.2): GTA5, Cityscapes, IDD, Mapillary Vistas. These datasets are further described in Section 2.3.2. Yet again, we standardize the label set with the 7 super classes common to all four datasets.

5.3.3.2 Implementation Details

The experiments are conducted with PyTorch (Paszke et al. 2017). The adversarial framework is based on AdvEnt's published code. [3] The semantic segmentation model is DeepLab-V2 (L.-C. Chen et al. 2018a), built upon the ResNet-101 (K. He et al. 2016) backbone first initialized with ImageNet (Deng et al. 2009) pre-trained weights. In a continual setting, when considering a new target domain, the model is initialized with the weights of the previously trained model. All semantic segmentation models are trained by SGD (Bottou 2010) with learning

2. https://github.com/arthurdouillard/CVPR2021_PLOP
3. https://github.com/valeoai/ADVENT

rate 2.5×10^{-4}, momentum 0.9 and weight decay 10^{-4}. We train the discriminators using an Adam optimizer (Diederik P. Kingma 2015) with learning rate 10^{-4}. All experiments are conducted at the 640×320 resolution. For CTKT, the weights λ_{prev} (Equation 5.11) and λ_{dist} (Equation 5.13), balancing the knowledge transfered from the previously trained model, are set to 10^{-5} in all experiments.

5.3.3.3 Results

Ablation Study. We first consider the two-step continual task GTA5 → Cityscapes → IDD: the models are first trained on GTA5 → Cityscapes, then on GTA5 → IDD. We perform an ablation study of this setting of the proposed method CTKT after this second training step.

The results are displayed in Table 5.7. In this table, we analyze the impact of the

Table 5.7 – **Ablation study of the CTKT architecture for continual UDA.** The models are decomposed into multiple blocks: 'Prev. KT' denotes knowledge transfer from the previous model; 'Feat. Dist.' denotes feature distillation with LocalPod (Douillard et al. 2021a); 'Multi Heads' denotes training both a target-specific head with adversarial training and a target-agnostic head with knowledge transfer. The best results for each metric are in **bold**, the second best are underlined.

GTA5 → Cityscapes → IDD						
Method	Prev. KT	Feat. Dist.	Multi Heads	Target	mIoU-7	mIoU-7 Avg.
GTA5 → Cityscapes Baseline (Vu et al. 2019a)				Cityscapes IDD	**69.0** 61.5	65.2
Continual Baseline (Vu et al. 2019a)				Cityscapes IDD	63.6 65.4	64.5
Previous model Knowledge Transfer	✓			Cityscapes IDD	65.3 65.3	65.3
LocalPod (Douillard et al. 2021a) Feature Distillation		✓		Cityscapes IDD	66.4 **65.5**	65.9
Prev. Knowledge Transfer + Feature Distillation	✓	✓		Cityscapes IDD	66.8 **65.5**	66.2
CTKT	✓	✓	✓	Cityscapes IDD	68.0 65.3	**66.7**

multiple elements of CTKT: knowledge transfer from the previous model ('Prev. KT'), feature distillation from the previous model with $\mathcal{L}_{\text{LocalPod}}$ ('Feat. Dist.'), and the decomposition into target-specific and target-agnostic branches ('Multi Heads').

First, we note that all the continual experiments exhibit comparable performance on IDD, on which they were trained last: from 65.2% to 65.5% mIoU on

IDD, much higher than the 61.5% mIoU on IDD of the previous model, not trained on this domain.

Then, we note that the continual baseline, not implementing any of these elements, suffers catastrophic forgetting on the previous Cityscapes target domain, dropping from 69.0% to 63.6% mIoU. Despite the improvement on IDD, on which it was trained last, the overall performance is lower than the previous model (difference of -0.7% mIoU Avg.).

Implemented on their own, both the knowledge transfer from the previous model and the feature distillation with LocalPod help limit the catastrophic forgetting on Cityscapes. Furthermore, they prove to be complementary, exhibiting greater performance on Cityscapes when combined.

Finally, CTKT adds the decomposition into target-specific and target-agnostic branches, which lessens catastrophic forgetting even further. CTKT performance on Cityscapes is only 1.0% mIoU lower than the previous model trained only on Cityscapes while being competitive on IDD. Overall, the performance of CTKT in terms of mIoU Avg. is significantly higher than the other models, notably $+2.2\%$ greater than the continual baseline.

GTA5 → Cityscapes → IDD → Mapillary. We consider the challenging setup involving three target domains – Cityscapes, Mapillary and IDD – discovered sequentially in a three-step continual learning setting and show results in Table 5.8. We compare results of continual UDA with multi-target UDA results from the previous section (Section 5.2). Due to the simultaneous availability of data for all the target domains, the multi-target setting is easier than the continual setting and the performance of multi-target experiments are expected to be higher than those of continual UDA experiments.

The continual baseline performs worse than all the multi-target experiments with at least a -0.8% mIoU Avg. decrease. In particular, the results on the Cityscapes and IDD datasets, which have been seen in early continual training steps, are significantly lower than those of all the other models due to catastrophic forgetting. Moreover, its performance is notably degraded on the *human* and *vehicle* classes compared to the better performing multi-target models, which is especially critical for autonomous driving applications.

On the other hand, CTKT exhibits comparable performance to the rather competitive Multi-Dis. multi-target model with a 68.2% mIoU Avg., proving the efficiency of the proposed continual learning framework.

Table 5.8 – **Continual UDA segmentation performance on GTA5 → Cityscapes → IDD → Mapillary (three steps).** 'Setting' indicates if the experiment is multi-target (simultaneous training on all the target domains) or continual (target domains discovered sequentially, three-step training).

GTA5 → Cityscapes → IDD → Mapillary											
Method	Setting	Target	flat	constr.	object	nature	sky	human	vehicle	mIoU-7	mIoU-7 Avg.
Multi-Target Baseline (Vu et al. 2019a)	Multi-Target	Cityscapes	93.6	80.6	26.4	78.1	81.5	51.9	76.4	69.8	67.8
		IDD	92.0	54.6	15.7	77.2	90.5	50.8	78.6	65.6	
		Mapillary	89.2	72.4	32.4	73.0	92.7	41.6	74.9	68.0	
Multi-Dis.	Multi-Target	Cityscapes	94.6	80.0	20.6	79.3	84.1	44.6	78.2	68.8	68.2
		IDD	91.6	54.2	13.1	78.4	93.1	49.6	80.3	65.8	
		Mapillary	89.0	72.5	29.3	75.5	94.7	50.3	78.9	70.0	
MTKT	Multi-Target	Cityscapes	94.6	80.7	23.8	79.0	84.5	51.0	79.2	**70.4**	**69.1**
		IDD	91.7	55.6	14.5	78.0	92.6	49.8	79.4	**65.9**	
		Mapillary	90.5	73.7	32.5	75.5	94.3	51.2	80.2	**71.1**	
Continual Baseline (Vu et al. 2019a)	Continual	Cityscapes	92.9	79.0	18.7	76.9	84.1	47.3	72.9	67.4	67.0
		IDD	91.8	51.1	11.6	79.0	91.6	47.5	72.5	63.6	
		Mapillary	90.3	71.7	30.1	76.1	93.9	50.2	77.3	70.0	
CTKT	Continual	Cityscapes	94.9	80.2	19.3	79.4	80.7	53.2	78.2	69.4	68.2
		IDD	92.5	54.1	12.0	79.2	92.7	48.0	76.6	65.0	
		Mapillary	91.0	73.2	29.2	76.0	94.1	50.0	78.0	70.2	

5.4 Conclusion

Practical applications of UDA require models that perform on a multitude of different domains, such as multiple cities or various weather conditions. While effective in the traditional single-target setting, standard UDA strategies do not easily extend to multi-target domains scenarios.

I first tackled the multi-target UDA problem for semantic segmentation by introducing two frameworks to extend adversarial alignment UDA strategies to this task. The first one, Multi-Dis., actively enforces the pairwise alignment between the source and the target domains as well as between the alignment between each target and the other targets using multiple discriminators. The second one, MTKT, introduces a multi-teacher single-student strategy based on multiple target-specific segmentation heads and a single target-agnostic segmentation head. This strategy alleviates some instability of the concurrent adversarial training procedures of Multi-Dis..

Then, I proposed CTKT to tackle the novel continual UDA problem. Close to MTKT, CTKT adapts to each new target domain using a teacher-student strategy from a target-specific head to a target-agnostic head while transferring the knowledge from the previous model, trained to perform on all the previous target

domains. The proposed benchmarks and architectures deliver competitive baselines for developments of real-world use-cases UDA scenarios.

Chapter 6

CONCLUSION

In this chapter, we first summarize the contributions of this thesis before discuting directions for future work on Domain Adaptation (DA) for urban scene segmentation.

6.1 Summary of Contributions

This thesis focuses on improving Deep Learning (DL) architectures and training procedures for Unsupervised Domain Adaptation (UDA) in semantic segmentation on three themes: leveraging source domain Privileged Information (PI) (Chapter 3), estimating the confidence in the target domain segmentation predictions for Self-Training (ST) (Chapter 4), and simultaneously or continuously adapting to multiple target domains (Chapter 5).

Bilinear multimodal discriminator for adversarial UDA with PI. Chapter 3 studies UDA scenarios in which additional information is available on the source domain, which is especially common when considering a synthetic source domain. More specifically, we discuss depth information as PI for UDA by considering depth prediction as an additional task to further constrain the semantic segmentation model. We propose Bilinear Multimodal Domain Adaptation (BerMuDA) that aims at explicitly discovering multimodal interactions by learning a bilinear fusion between semantic segmentation and depth predictions in the domain discriminator. The tensor-based bilinear fusion allows modelling fine interactions between the two modalities and builds discriminative multimodal representations that help improve adversarial UDA, as shown in the experiments on synthetic-to-real urban scene UDA.

Estimation of confidence for target pseudo-label selection and ST. Chapter 4 tackles ST on the unlabeled target domain based on pseudo-label selection, a strategy borrowed from the semi-supervised learning community. In particular, we discuss the measure of confidence employed in the literature, generally revolving around the Maximum Class Probability (MCP) of the segmentation model predic-

tions. We first propose Entropy-based Self-supervised Learning (ESL), a simple yet efficient improvement over MCP-based approaches, that considers the entropy of the prediction as measure of confidence. Since the entropy takes into consideration the complete distribution of the prediction for each pixel, this measure is more reliable than MCP in assessing the confidence of the semantic segmentation network. Experiments on multiple UDA models and settings demonstrate that the introduction of ESL substantially improves the performance of the ST procedure compared to the MCP criterion.

To go further, we propose the more complex Confidence Learning for Domain Adaptation (ConDA) that aims at estimating the True Class Probability (TCP), best suited as measure of confidence. ConDA learns an auxiliary neural network dedicated to confidence estimation, with an architecture specifically designed for semantic segmentation and an adversarial training procedure tailored to UDA. ConDA proved to significantly improve over strong baselines on various benchmarks.

Adversarial frameworks for multi-target and continual UDA. Chapter 5 discusses UDA to multiple target domains in semantic segmentation, which represents a practical challenge for UDA applications such as autonomous driving. We propose two adversarial frameworks to tackle multi-target UDA, Multi-Discriminator (Multi-Dis.) and Multi-Target Knowledge Transfer (MTKT), that extend output-based adversarial approaches of standard UDA to this new setting.

Furthermore, we address continual UDA where target domains are discovered sequentially, adding the challenge of catastrophic forgetting to the UDA problem. We propose Continual Target Knowledge Transfer (CTKT) that allows the continuous training of output-based adversarial UDA approaches on new target domains by distilling the knowledge of previous iterations.

Finally, we additionally propose unified benchmarks for adaptation to multiple urban scene datasets on which we test our different approaches and baselines, setting competitive standards for these novel tasks.

6.2 Perspectives for Future Work

Let us now discuss some directions that could be addressed in future work in relation to our contributions.

Extending the DA setting to more practical scenarios

Solving DA settings closer to practical applications has been one of the main focuses of this thesis, Chapter 3 addressing UDA with additional PI in the source domain and Chapter 5 tackling UDA to more than one target domain. Designing architectures that can leverage additional data or perform in more complex scenarios is one of the major research directions toward practical use cases of UDA.

Source-free UDA. While we considered in Chapter 5 that it may be unfeasible to keep previous data in continous UDA settings for diverse reasons such as privacy, the same assessment could be made about the source data. Source-free UDA (Kundu et al. 2021; Y. Liu et al. 2021) considers settings in which only a source pretrained model is made available for adaptation to the target domain, making the task significantly more challenging. Nonetheless, building architectures which can continuously learn on new target domains (like the proposed CTKT in Chapter 5), while remembering their knowledge of previous target domains and not requiring to memorize the source data to adapt on this new target data would consitute a major breakthrough for practical applications such as autonomous vehicles.

Semi-supervised DA. In this thesis, we specifically considered UDA, which is the most commonly tackled DA setting in the literature. However, even if manual semantic segmentation annotation is costly, annotating a few real urban scene images is conceivable. Semi-supervised DA (B. Li et al. 2021; Singh et al. 2021) studies DA scenarios in which the target training dataset has a few labeled examples. Specific architectures and strategies must be employed to properly leverage this additional target information and improve the performance of the DA. Nonetheless, the confidence estimation method proposed in Chapter 4 would perfectly fit semi-supervised DA and enhance the target dataset with confident pseudo-labels for the unlabeled samples.

Adaptation with new target labels. Despite the many differences and gaps between the considered domains, UDA generally assumes that the label spaces of the domains are identical. However, some domains may show some very specific classes, for example "autorickshaw" in Indian urban scenes in IDD. Ideally, for practical applications, UDA models should be able to, at least, detect new classes, if not actually predict their class with marginal supervision. Such incremental or boundless DA settings have been tackled on various Computer Vision (CV) tasks by a few works in the literature (Kundu et al. 2020; Bucher et al. 2020). The continual UDA setting proposed in Chapter 5 could be extended with the

more traditional continual learning paradigm in mind: when discovering new target domains, the model must also learn to predict new classes, maybe with a few annotated examples of this new class. Such a setting would combine the challenges of the continual UDA problem discussed in Chapter 5 with those of the continual semantic segmentation task, studied for example in (Douillard et al. 2021a).

Adaptation with other sensors

In this thesis and in the vast majority of the literature, UDA is considered on traditional camera inputs. However, in the context of autonomous vehicles, the system has access to a variety of different sensors, each providing unique and informative signals about their environment. As for semantic segmentation of images, the annotation of LiDAR point clouds, for instance, is expansive and UDA is a compelling solution to leverage these signals, especially when combined with traditional cameras.

Traditional camera to fish-eye camera. While the UDA literature, and more generally the semantic segmentation community, mostly consider traditional cameras, fisheye cameras are commonly employed in practice for capturing a large field of view, in particular in automotive applications. Solving CV tasks on fisheye camera data would have a massive impact on autonomous driving, and recent datasets such as Woodscape (Yogamani et al. 2019) promote the use of such data. One may consider fisheye camera input as a new target domain in a UDA strategy and leverage traditional camera data as source domain. Adapting multi-target UDA strategies like those we propose in Chapter 5 by considering a synthetic source domain and the multiple video inputs of an autonomous car, coming from traditional and fisheye cameras, as multiple target domains to build a unified semantic segmentation model for all the cameras of the vehicle could have a tremendous impact on autonomous driving systems.

UDA for 3D CV by leveraging LiDAR and radar. Apart from cameras, autonomous vehicles employ other sensors to acquire a 3D knowledge of their environment, in particular radar and LiDAR. While Chapter 3 discussed leveraging depth as source PI in UDA settings, one could practically have access to depth information or 3D point clouds in the target domain and in real-time. (Jaritz et al. 2020) first proposes to perform UDA for 3D semantic segmentation on point clouds by leveraging both camera and LiDAR inputs. In such a context, approaches like the proposed BerMuDA (Chapter 3) could help build discriminative feature accross

the multiple modalities to improve the cooperation between the modalities and better adapt on the target domain.

Applying UDA to practical driving assistance systems

Valeo is firmly committed to improving advanced driving assistance systems (ADAS) and to developing autonomous vehicles. Experiments on prototypes of autonomous cars or on controlled circuits in collaboration with other research and development teams of the company would prove to be the perfect testing ground for the proposed UDA methods. For instance, we could check the impact of integrating day-to-night adaptation with UDA methods on an experimental autonomous driving car. This applied context would also allow us to evaluate the algorithms on more practical metrics than the Intersection over Union (IoU), such as counting the number of times the driver takes over the wheel.

BIBLIOGRAPHY

Ben-Younes, Hedi, Rémi Cadene, Nicolas Thome, and Matthieu Cord (2019). "BLOCK: Bilinear Superdiagonal Fusion for Visual Question Answering and Visual Relationship Detection". In: *Proc. AAAI Conf. Artifical Intelligence* (cit. on pp. 51, 53, 55).

Ben-Younes, Hedi, Rémi Cadène, Nicolas Thome, and Matthieu Cord (2017). "MUTAN: Multimodal Tucker Fusion for Visual Question Answering". In: *Proc. Int. Conf. Computer Vision* (cit. on pp. 51, 55).

Bishop, Christopher M (2006). *Pattern recognition and machine learning*. Springer (cit. on p. 8).

Bottou, Léon (2010). "Large-scale machine learning with stochastic gradient descent". In: *Proc. Int. Conf. Computational Statistics* (cit. on pp. 12, 69, 97, 108).

Bousmalis, Konstantinos, Nathan Silberman, David Dohan, Dumitru Erhan, and Dilip Krishnan (2017). "Unsupervised Pixel-Level Domain Adaptation with Generative Adversarial Networks". In: *Proc. Conf. Comp. Vision Pattern Rec.* (cit. on pp. 20, 21, 49).

Bousmalis, Konstantinos, George Trigeorgis, Nathan Silberman, Dilip Krishnan, and Dumitru Erhan (2016). "Domain Separation Networks". In: *Int. Conf. Neural Inf. Proc. Systems* (cit. on pp. 24, 25).

Bucher, Maxime, Tuan-Hung Vu, Mathieu Cord, and Patrick Pérez (2020). "BUDA: Boundless Unsupervised Domain Adaptation in Semantic Segmentation". In: (cit. on pp. 14, 115).

Caruana, Rich (1997). "Multitask learning". In: *Machine learning* 28.1, pp. 41–75 (cit. on pp. 14, 47).

Cermelli, Fabio, Massimiliano Mancini, Samuel Rota Bulò, Elisa Ricci, and Barbara Caputo (2020). "Modeling the Background for Incremental Learning in Semantic Segmentation". In: *Proc. Conf. Comp. Vision Pattern Rec.* (Cit. on p. 104).

Chang, Wei-Lun, Hui-Po Wang, Wen-Hsiao Pengg, and Wei-Chen Chiu (2019). "All about Structure: Adapting Structural Information across Domains for Boosting Semantic Segmentation". In: *Proc. Conf. Comp. Vision Pattern Rec.* (Cit. on pp. 25, 26, 60, 82).

Chen, Liang-Chieh, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan Yuille (2018a). "DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs". In: *IEEE Trans. Pattern Anal. Machine Intell.* (cit. on pp. 56, 69, 80, 97, 108).

Chen, Liang-Chieh, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille (2014). "Semantic image segmentation with deep convolutional nets and fully connected crfs". In: *arXiv* (cit. on p. 11).

Chen, Liang-Chieh, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille (2017). "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs". In: *IEEE Trans. Pattern Anal. Machine Intell.* (cit. on pp. 11, 12).

Chen, Liang-Chieh, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam (2018b). "Encoder-decoder with atrous separable convolution for semantic image segmentation". In: *Proc. Europ. Conf. Computer Vision* (cit. on p. 11).

Chen, Tian Qi and Mark Schmidt (2016). "Fast Patch-based Style Transfer of Arbitrary Style". In: *arXiv* (cit. on pp. 23, 24).

Chen, Ziliang, Jingyu Zhuang, Xiaodan Liang, and Liang Lin (2019). "Blending-target domain adaptation by adversarial meta-adaptation networks". In: *Proc. Conf. Comp. Vision Pattern Rec.* (Cit. on p. 90).

Corbière, Charles, Nicolas Thome, Avner Bar-Hen, Matthieu Cord, and Patrick Pérez (2019). "Addressing failure prediction by learning model confidence". In: (cit. on pp. 66, 74).

Corbière, Charles, Nicolas Thome, Antoine Saporta, Tuan-Hung Vu, Matthieu Cord, and Patrick Perez (2021). "Confidence Estimation via Auxiliary Models". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* (cit. on pp. 5, 61, 74, 77).

Cord, Matthieu and Pádraig Cunningham (2008). *Machine learning techniques for multimedia: case studies on organization and retrieval*. Springer Science & Business Media (cit. on p. 8).

Cordts, Marius, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele (2016a). "The Cityscapes Dataset for Semantic Urban Scene Understanding". In: *Proc. Conf. Comp. Vision Pattern Rec.* (Cit. on pp. 34, 56).

Cordts, Marius, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele (2016b). "The Cityscapes dataset for semantic urban scene understanding". In: *Proc. Conf. Comp. Vision Pattern Rec.* (Cit. on p. 3).

Courty, Nicolas, Rémi Flamary, Amaury Habrard, and Alain Rakotomamonjy (2017). "Joint distribution optimal transportation for domain adaptation". In: *Int. Conf. Neural Inf. Proc. Systems* (cit. on p. 19).

Courty, Nicolas, Rémi Flamary, Devis Tuia, and Alain Rakotomamonjy (2016). "Optimal transport for domain adaptation". In: (cit. on p. 19).

Csurka, Gabriela (2017). "A comprehensive survey on domain adaptation for visual applications". In: *Domain adaptation in computer vision applications* (cit. on p. 14).

Damodaran, Bharath Bhushan, Benjamin Kellenberger, Rémi Flamary, Devis Tuia, and Nicolas Courty (2018). "Deepjdot: Deep joint distribution optimal transport for unsupervised domain adaptation". In: *Proc. Europ. Conf. Computer Vision* (cit. on pp. 19, 20).

Deng, Jia, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei (2009). "ImageNet: A Large-Scale Hierarchical Image Database". In: *Proc. Conf. Comp. Vision Pattern Rec.* (Cit. on pp. 56, 69, 97, 108).

Diederik P. Kingma, Jimmy Ba (2015). "Adam: A Method for Stochastic Optimization". In: *Proc. Int. Conf. Learning Representations* (cit. on pp. 12, 70, 97, 109).

Douillard, Arthur, Yifu Chen, Arnaud Dapogny, and Matthieu Cord (2021a). "Plop: Learning without forgetting for continual semantic segmentation". In: *Proc. Conf. Comp. Vision Pattern Rec.* (Cit. on pp. 104, 107–109, 116).

Douillard, Arthur, Yifu Chen, Arnaud Dapogny, and Matthieu Cord (2021b). "Tackling Catastrophic Forgetting and Background Shift in Continual Semantic Segmentation". In: (cit. on p. 104).

Douillard, Arthur, Matthieu Cord, Charles Ollion, Thomas Robert, and Eduardo Valle (2020). "PODNet: Pooled Outputs Distillation for Small-Tasks Incremental Learning". In: *Proc. Europ. Conf. Computer Vision* (cit. on p. 104).

French, Robert (1999). "Catastrophic forgetting in connectionist networks". In: *Trends in cognitive sciences* (cit. on p. 104).

Fukui, Akira, Dong Huk Park, Daylen Yang, Anna Rohrbach, Trevor Darrell, and Marcus Rohrbach (2016). "Multimodal Compact Bilinear Pooling for Visual Question Answering and Visual Grounding". In: *Proc. Conf. Empirical Methods Nat. Lang. Proc.* (Cit. on p. 51).

Ganin, Yaroslav and Victor Lempitsky (2015). "Unsupervised Domain Adaptation by Backpropagation". In: *Proc. Int. Conf. Machine Learning* (cit. on pp. 27, 28).

Gholami, Behnam, Pritish Sahu, Ognjen Rudovic, Konstantinos Bousmalis, and Vladimir Pavlovic (2020). "Unsupervised multi-target domain adaptation: An information theoretic approach". In: *IEE Trans. Image Processing* (cit. on pp. 90, 92, 104).

Girshick, Ross (2015). "Fast R-CNN". In: *Proc. Int. Conf. Computer Vision* (cit. on p. 54).

Goodfellow, Ian, Yoshua Bengio, and Aaron Courville (2016). *Deep learning*. MIT Press (cit. on p. 10).

Goodfellow, Ian, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio (2014). "Generative adversarial nets". In: *Int. Conf. Neural Inf. Proc. Systems* (cit. on pp. 12, 20).

Grandvalet, Yves and Yoshua Bengio (2005). "Semi-supervised Learning by Entropy Minimization". In: *Int. Conf. Neural Inf. Proc. Systems* (cit. on p. 62).

Gretton, Arthur, Karsten M. Borgwardt, Malte J. Rasch, Bernhard Schölkopf, and Alexander Smola (2012). "A Kernel Two-Sample Test". In: *J. Mach. Learn. Res.* (cit. on p. 15).

Guo, Chuan, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger (2017). "On Calibration of Modern Neural Networks". In: *Proc. Int. Conf. Machine Learning* (cit. on p. 76).

Hastie, Trevor (2009). *The elements of statistical learning: data mining, inference, and prediction*. Springer (cit. on p. 8).

Hayes, Tyler L., Kushal Kafle, Robik Shrestha, Manoj Acharya, and Christopher Kanan (2020). "REMIND Your Neural Network to Prevent Catastrophic Forgetting". In: *Proc. Europ. Conf. Computer Vision* (cit. on p. 104).

He, Jianzhong, Xu Jia, Shuaijun Chen, and Jianzhuang Liu (2021). "Multi-source domain adaptation with collaborative learning for semantic segmentation". In: *Proc. Conf. Comp. Vision Pattern Rec.* (Cit. on pp. 24, 89).

He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun (2016). "Deep Residual Learning for Image Recognition". In: *Proc. Conf. Comp. Vision Pattern Rec.* (Cit. on pp. 11, 56, 69, 97, 108).

Hinton, Geoffrey, Oriol Vinyals, and Jeff Dean (2015). "Distilling the knowledge in a neural network". In: *arXiv preprint arXiv:1503.02531* (cit. on p. 95).

Hoffman, Judy, Eric Tzeng, Taesung Park, Jun-Yan Zhu, Phillip Isola, Kate Saenko, Alexei A. Efros, and Trevor Darrell (2018). "CyCADA: Cycle Consistent Adversarial Domain Adaptation". In: *Proc. Int. Conf. Machine Learning* (cit. on pp. 21, 22, 82, 100).

Hoffman, Judy, Dequan Wang, Fisher Yu, and Trevor Darrell (2016). "FCNs in the Wild: Pixel-level Adversarial and Constraint-based Adaptation". In: *arXiv:1612.02649* (cit. on pp. 29, 96).

Huang, Xun and Serge J. Belongie (2017). "Arbitrary Style Transfer in Real-time with Adaptive Instance Normalization". In: *Proc. Int. Conf. Computer Vision* (cit. on p. 22).

Hull, Jonathan (1994). "Database for handwritten text recognition research". In: *IEEE Trans. Pattern Anal. Machine Intell.* (cit. on p. 34).

Isobe, Takashi, Xu Jia, Shuaijun Chen, Jianzhong He, Yongjie Shi, Jianzhuang Liu, Huchuan Lu, and Shengjin Wang (2021). "Multi-target domain adaptation with collaborative consistency learning". In: *Proc. Conf. Comp. Vision Pattern Rec.* (Cit. on pp. 90, 91).

Jaritz, Maximilian, Tuan-Hung Vu, Raoul de Charette, Emilie Wirbel, and Patrick Pérez (2020). "XMUDA: Cross-modal unsupervised domain adaptation for 3d semantic segmentation". In: *Proc. Conf. Comp. Vision Pattern Rec.* (Cit. on p. 116).

Kim, Jin-Hwa, Kyoung-Woon On, Woosang Lim, Jeonghee Kim, Jung-Woo Ha, and Byoung-Tak Zhang (2016). "Hadamard Product for Low-Rank Bilinear Pooling". In: *Proc. Int. Conf. Learning Representations* (cit. on p. 51).

Kim, Myeongjin and Hyeran Byun (2020). "Learning texture invariant representation for domain adaptation of semantic segmentation". In: *Proc. Conf. Comp. Vision Pattern Rec.* (Cit. on pp. 23, 24).

Kirkpatrick, James, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell (2017). "Overcoming catastrophic forgetting in neural networks". In: *Proc. National Academy of Sciences* (cit. on p. 104).

Kokkinos, Iasonas (2017). "UberNet: Training a Universal Convolutional Neural Network for Low-, Mid-, and High-Level Vision Using Diverse Datasets and Limited Memory". In: *Proc. Conf. Comp. Vision Pattern Rec.* (Cit. on p. 54).

Krizhevsky, Alex and Geoffrey Hinton (2009). "Learning multiple layers of features from tiny images". In: *Master's thesis, Department of Computer Science, University of Toronto* (cit. on p. 76).

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey Hinton (2012). "ImageNet Classification with Deep Convolutional Neural Networks". In: *NIPS* (cit. on pp. 11, 16–18, 81).

Kundu, Jogendra Nath, Akshay Kulkarni, Amit Singh, Varun Jampani, and R Venkatesh Babu (2021). "Generalize then adapt: Source-free domain adaptive semantic segmentation". In: *Proc. Int. Conf. Computer Vision* (cit. on p. 115).

Kundu, Jogendra Nath, Nishank Lakkakula, and R Venkatesh Babu (2019). "Um-adapt: Unsupervised multi-task adaptation using adversarial cross-task distillation". In: *Proc. Int. Conf. Computer Vision* (cit. on pp. 47, 48).

Kundu, Jogendra Nath, Rahul Mysore Venkatesh, Naveen Venkat, Ambareesh Revanur, and R Venkatesh Babu (2020). "Class-incremental domain adaptation". In: *Proc. Europ. Conf. Computer Vision* (cit. on pp. 14, 115).

Laina, Iro, Christian Rupprecht, Vasileios Belagiannis, Federico Tombari, and Nassir Navab (2016). "Deeper Depth Prediction with Fully Convolutional Residual Networks". In: *Proc. IEEE Int. Conf. 3D Vision* (cit. on p. 54).

LeCun, Yann, Bernhard Boser, John Denker, Donnie Henderson, Richard Howard, Wayne Hubbard, and Lawrence Jackel (1989). "Handwritten digit recognition with a back-propagation network". In: *Adv. Neural Inf. Proc. Systems* (cit. on pp. 11, 34).

Lee, Dong-Hyun (2013). "Pseudo-Label : The Simple and Efficient Semi-Supervised Learning Method for Deep Neural Networks". In: *Int. Conf. Machine Learning Workshop* (cit. on pp. 4, 62).

Lee, Kuan-Hui, German Ros, Jie Li, and Adrien Gaidon (2019). "SPIGAN: Privileged Adversarial Learning from Simulation". In: *Proc. Int. Conf. Learning Representations* (cit. on pp. 49, 50, 52, 58, 59, 96).

Li, Bo, Yezhen Wang, Shanghang Zhang, Dongsheng Li, Kurt Keutzer, Trevor Darrell, and Han Zhao (2021). "Learning invariant representations and risks for semi-supervised domain adaptation". In: *Proc. Conf. Comp. Vision Pattern Rec.* (Cit. on pp. 14, 115).

Li, Haoliang, Sinno Jialin Pan, Shiqi Wang, and Alex C Kot (2018). "Domain generalization with adversarial feature learning". In: *Proc. Conf. Comp. Vision Pattern Rec.* (Cit. on p. 89).

Li, Yunsheng, Lu Yuan, and Nuno Vasconcelos (2019a). "Bidirectional Learning for Domain Adaptation of Semantic Segmentation". In: *Proc. Conf. Comp. Vision Pattern Rec.* (Cit. on pp. 60, 64, 66, 71, 82).

Li, Yunsheng, Lu Yuan, and Nuno Vasconcelos (2019b). "Bidirectional Learning for Domain Adaptation of Semantic Segmentation". In: *Proc. Conf. Comp. Vision Pattern Rec.* (Cit. on pp. 65, 69, 70).

Li, Z. and D. Hoiem (2016). "Learning without Forgetting". In: *Proc. Europ. Conf. Computer Vision* (cit. on p. 104).

Liu, Xiao, Fanjin Zhang, Zhenyu Hou, Li Mian, Zhaoyu Wang, Jing Zhang, and Jie Tang (2021). "Self-supervised learning: Generative or contrastive". In: *IEEE Trans. on Knowledge and Data Eng.* (cit. on p. 62).

Liu, Yuang, Wei Zhang, and Jun Wang (2021). "Source-Free Domain Adaptation for Semantic Segmentation". In: *Proc. Conf. Comp. Vision Pattern Rec.* (cit. on p. 115).

Liu, Ziwei, Zhongqi Miao, Xingang Pan, Xiaohang Zhan, Dahua Lin, Stella X. Yu, and Boqing Gong (2020). "Open Compound Domain Adaptation". In: *Proc. Conf. Comp. Vision Pattern Rec.* (Cit. on p. 90).

Long, Mingsheng, Yue Cao, Jianmin Wang, and Michael Jordan (2015). "Learning Transferable Features with Deep Adaptation Networks". In: *Proc. Int. Conf. Machine Learning* (cit. on pp. 16, 17).

Long, Mingsheng, Zhangjie Cao, Jianmin Wang, and Michael I. Jordan (2018). "Conditional Adversarial Domain Adaptation". In: *Int. Conf. Neural Inf. Proc. Systems* (cit. on p. 52).

Long, Mingsheng, Han Zhu, Jianmin Wang, and Michael I. Jordan (2017). "Deep Transfer Learning with Joint Adaptation Networks". In: *Proc. Int. Conf. Machine Learning* (cit. on p. 17).

Lopez-Paz, David and Marc'Aurelio Ranzato (2017). "Gradient Episodic Memory for Continual Learning". In: *Int. Conf. Neural Inf. Proc. Systems* (cit. on p. 104).

Matsuura, Toshihiko and Tatsuya Harada (2020). "Domain generalization using a mixture of multiple latent domains". In: *Proc. AAAI Conf. Artifical Intelligence* (cit. on p. 89).

Mei, Ke, Chuang Zhu, Jiaqi Zou, and Shanghang Zhang (2020). "Instance adaptive self-training for unsupervised domain adaptation". In: *Proc. Europ. Conf. Computer Vision* (cit. on p. 66).

Michieli, Umberto and Pietro Zanuttigh (2019). "Incremental Learning Techniques for Semantic Segmentation". In: *Proc. Int. Conf. Computer Vision Workshop* (cit. on p. 104).

Mordan, Taylor, Antoine Saporta, Alexandre Alahi, Matthieu Cord, and Patrick Pérez (2020). "Bilinear Multimodal Discriminator for Adversarial Domain Adaptation with Privileged Information". In: *Symposium of the European Association for Research in Transportation (hEART)* (cit. on pp. 5, 45).

Mordan, Taylor, Nicolas Thome, Gilles Henaff, and Matthieu Cord (2018). "Revisiting Multi-Task Learning with ROCK: a Deep Residual Auxiliary Block for Visual Detection". In: *Int. Conf. Neural Inf. Proc. Systems* (cit. on pp. 50, 54, 55, 59).

Morsing, Lukas Hedegaard, Omar Ali Sheikh-Omar, and Alexandros Iosifidis (2021). "Supervised domain adaptation using graph embedding". In: *Proc. Int. Conf. Pattern Recognition* (cit. on p. 14).

Motiian, Saeid, Marco Piccirilli, Donald A Adjeroh, and Gianfranco Doretto (2017). "Unified deep supervised domain adaptation and generalization". In: *Proc. Int. Conf. Computer Vision* (cit. on p. 14).

Murez, Zak, Soheil Kolouri, David J. Kriegman, Ravi Ramamoorthi, and Kyungnam Kim (2018). "Image to Image Translation for Domain Adaptation". In: *Proc. Conf. Comp. Vision Pattern Rec.* (cit. on pp. 26, 27).

Netzer, Yuval, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng (2011). "Reading Digits in Natural Images with Unsupervised Feature Learning". In: *Int. Conf. Neural Inf. Proc. Systems Workshop* (cit. on p. 34).

Neuhold, Gerhard, Tobias Ollmann, Samuel Rota Bulò, and Peter Kontschieder (2017). "The Mapillary Vistas Dataset for Semantic Understanding of Street Scenes". In: *Proc. Int. Conf. Computer Vision* (cit. on p. 38).

Nguyen, Van-Anh, Tuan Nguyen, Trung Le, Quan Hung Tran, and Dinh Phung (2021). "Stem: An approach to multi-source domain adaptation with guarantees". In: *Proc. Int. Conf. Computer Vision* (cit. on p. 89).

Pandey, Prashant, Mrigank Raman, Sumanth Varambally, and Prathosh AP (2021). "Generalization on Unseen Domains via Inference-Time Label-Preserving Target Projections". In: *Proc. Conf. Comp. Vision Pattern Rec.* (Cit. on p. 89).

Park, Kwanyong, Sanghyun Woo, Inkyu Shin, and In-So Kweon (2020). "Discover, hallucinate, and adapt: Open compound domain adaptation for semantic segmentation". In: *Int. Conf. Neural Inf. Proc. Systems* (cit. on p. 90).

Paszke, Adam, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer

(2017). "Automatic differentiation in PyTorch". In: *Workshop at Int. Conf. Neural Inf. Proc. Systems* (cit. on pp. 69, 81, 97, 108).

Peng, Xingchao, Zijun Huang, Ximeng Sun, and Kate Saenko (2019). "Domain agnostic learning with disentangled representations". In: *Proc. Int. Conf. Machine Learning* (cit. on p. 90).

Qi, Fan, Xiaoshan Yang, and Changsheng Xu (2018). "A Unified Framework for Multimodal Domain Adaptation". In: *Proc. ACM Int. Conf. Multimedia* (cit. on p. 52).

Qian, Ning (1999). "On the momentum term in gradient descent learning algorithms". In: *Neural networks* (cit. on p. 12).

Rebuffi, Sylvestre-Alvise, Alexander Kolesnikov, Georg Sperl, and Christoph H. Lampert (2017). "iCaRL: Incremental Classifier and Representation Learning". In: *Proc. Conf. Comp. Vision Pattern Rec.* (Cit. on p. 104).

Ren, Zhongzheng and Yong Jae Lee (2018). "Cross-Domain Self-supervised Multi-task Feature Learning using Synthetic Imagery". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (cit. on pp. 47, 48).

Richter, Stephan R., Vibhav Vineet, Stefan Roth, and Vladlen Koltun (2016). "Playing for Data: Ground Truth from Computer Games". In: *Proc. Europ. Conf. Computer Vision* (cit. on p. 35).

Robins, Anthony (1995). "Catastrophic Forgetting, Rehearsal and Pseudorehearsal". In: *Connection Science* (cit. on p. 104).

Ros, German, Laura Sellart, Joanna Materzynska, David Vazquez, and Antonio M. Lopez (2016). "The SYNTHIA Dataset: A Large Collection of Synthetic Images for Semantic Segmentation of Urban Scenes". In: *Proc. Conf. Comp. Vision Pattern Rec.* (Cit. on pp. 35, 56).

Rumelhart, David E, Richard Durbin, Richard Golden, and Yves Chauvin (1995). "Backpropagation: The basic theory". In: *Backpropagation: Theory, architectures and applications* (cit. on p. 12).

Saenko, Kate, Brian Kulis, Mario Fritz, and Trevor Darrell (2010). "Adapting Visual Category Models to New Domains". In: *Proc. Europ. Conf. Computer Vision* (cit. on p. 34).

Saito, Kuniaki, Donghyun Kim, Stan Sclaroff, Trevor Darrell, and Kate Saenko (2019). "Semi-supervised domain adaptation via minimax entropy". In: *Proc. Int. Conf. Computer Vision* (cit. on p. 14).

Saporta, Antoine, Tuan-Hung Vu, Matthieu Cord, and Patrick Pérez (2020). "ESL: Entropy-guided self-supervised learning for domain adaptation in semantic segmentation". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshop on Scalability in Autonomous Driving* (cit. on pp. 5, 61).

Saporta, Antoine, Tuan-Hung Vu, Matthieu Cord, and Patrick Pérez (2021). "Multi-Target Adversarial Frameworks for Domain Adaptation in Semantic Segmen-

tation". In: *IEEE/CVF International Conference on Computer Vision (ICCV)* (cit. on pp. 5, 87).

Simonyan, Karen and Andrew Zisserman (2014). "Very deep convolutional networks for large-scale image recognition". In: *arXiv* (cit. on p. 11).

Singh, Anurag, Naren Doraiswamy, Sawa Takamuku, Megh Bhalerao, Titir Dutta, Soma Biswas, Aditya Chepuri, Balasubramanian Vengatesan, and Naotake Natori (2021). "Improving semi-supervised domain adaptation using effective target selection and semantics". In: *Proc. Conf. Comp. Vision Pattern Rec.* (Cit. on p. 115).

Sun, Baochen, Jiashi Feng, and Kate Saenko (2016). "Return of frustratingly easy domain adaptation". In: *Proc. AAAI Conf. Artifical Intelligence* (cit. on p. 18).

Sun, Baochen and Kate Saenko (2016). "Deep coral: Correlation alignment for deep domain adaptation". In: *Proc. Europ. Conf. Computer Vision* (cit. on p. 18).

Szegedy, Christian, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich (2015). "Going deeper with convolutions". In: *Proc. Conf. Comp. Vision Pattern Rec.* (Cit. on p. 11).

Szegedy, Christian, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna (2016). "Rethinking the inception architecture for computer vision". In: *Proc. Conf. Comp. Vision Pattern Rec.* (Cit. on p. 11).

Szeliski, Richard (2010). *Computer vision: algorithms and applications*. Springer Science & Business Media (cit. on p. 8).

Tan, Chuanqi, Fuchun Sun, Tao Kong, Wenchang Zhang, Chao Yang, and Chunfang Liu (2018). "A survey on deep transfer learning". In: *Int. Conf. Artificial Neural Networks* (cit. on p. 13).

Tan, Shuhan, Xingchao Peng, and Kate Saenko (2020). "Class-imbalanced domain adaptation: an empirical odyssey". In: *Proc. Europ. Conf. Computer Vision* (cit. on p. 14).

Thrun, Sebastian (1998). "Lifelong Learning Algorithms". In: *Springer Learning to Learn* (cit. on p. 104).

Tieleman, Tijmen, Geoffrey Hinton, et al. (2012). "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude". In: *COURSERA: Neural networks for machine learning* (cit. on p. 12).

Tsai, Yi-Hsuan, Wei-Chih Hung, Samuel Schulter, Kihyuk Sohn, Ming-Hsuan Yang, and Manmohan Chandraker (2018). "Learning to Adapt Structured Output Space for Semantic Segmentation". In: *Proc. Conf. Comp. Vision Pattern Rec.* (Cit. on pp. 30–33, 46, 57–59, 69, 71, 82, 92, 94, 96, 104).

Tzeng, Eric, Judy Hoffman, Kate Saenko, and Trevor Darrell (2017). "Adversarial Discriminative Domain Adaptation". In: *Proc. Conf. Comp. Vision Pattern Rec.* (cit. on pp. 27, 28).

Tzeng, Eric, Judy Hoffman, Ning Zhang, Kate Saenko, and Trevor Darrell (2014). "Deep Domain Confusion: Maximizing for Domain Invariance". In: *CoRR* (cit. on pp. 16, 18).

Vapnik, Vladimir and Rauf Izmailov (2015). "Learning Using Privileged Information: Similarity Control and Knowledge Transfer". In: *J. Mach. Learn. Res.* (cit. on pp. 4, 46).

Varma, Girish, Anbumani Subramanian, Anoop Namboodiri, Manmohan Chandraker, and C V Jawahar (2019). "DD: A Dataset for Exploring Problems of Autonomous Navigation in Unconstrained Environments". In: *Winter Conf. Appl. of Comp. Vision* (cit. on p. 38).

Vu, Tuan-Hung, Himalaya Jain, Maxime Bucher, Matthieu Cord, and Patrick Pérez (2019a). "ADVENT: Adversarial Entropy Minimization for Domain Adaptation in Semantic Segmentation". In: *Proc. Conf. Comp. Vision Pattern Rec.* (Cit. on pp. 31–33, 46, 50, 59, 69, 71–74, 81–83, 92, 94, 96, 98, 99, 101, 102, 104, 109, 111).

Vu, Tuan-Hung, Himalaya Jain, Maxime Bucher, Matthieu Cord, and Patrick Pérez (2019b). "DADA: Depth-aware Domain Adaptation in Semantic Segmentation". In: *Proc. Int. Conf. Computer Vision* (cit. on pp. 50–52, 57–59, 82, 83, 86, 96).

Wang, Yaqing, Quanming Yao, James T Kwok, and Lionel M Ni (n.d.). "Generalizing from a few examples: A survey on few-shot learning". In: *ACM Computing Surveys* () (cit. on p. 13).

Wu, Zuxuan, Xintong Han, Yen-Liang Lin, Mustafa Gokhan Uzunbas, Tom Goldstein, Ser Nam Lim, and Larry S Davis (2018). "Dcan: Dual channel-wise alignment networks for unsupervised scene adaptation". In: *Proc. Europ. Conf. Computer Vision* (cit. on pp. 21, 22, 100).

Yang, Xiangli, Zixing Song, Irwin King, and Zenglin Xu (2021). "A Survey on Deep Semi-supervised Learning". In: *arXiv* (cit. on p. 62).

Yang, Yanchao and Stefano Soatto (2020). "Fda: Fourier domain adaptation for semantic segmentation". In: *Proc. Conf. Comp. Vision Pattern Rec.* (Cit. on pp. 23, 24, 100).

Yogamani, Senthil, Ciarán Hughes, Jonathan Horgan, Ganesh Sistu, Padraig Varley, Derek O'Dea, Michal Uricár, Stefan Milz, Martin Simon, Karl Amende, et al. (2019). "WoodScape: A multi-task, multi-camera fisheye dataset for autonomous driving". In: *Proc. Int. Conf. Computer Vision* (cit. on p. 116).

Yu, Huanhuan, Menglei Hu, and Songcan Chen (2018). "Multi-target unsupervised domain adaptation without exactly shared categories". In: *arXiv preprint arXiv:1809.00852* (cit. on pp. 90, 92, 104).

Zhao, Hengshuang, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia (2017). "Pyramid scene parsing network". In: *Proc. Conf. Comp. Vision Pattern Rec.* (Cit. on p. 11).

Zhao, Shanshan, Mingming Gong, Tongliang Liu, Huan Fu, and Dacheng Tao (2020). "Domain generalization via entropy regularization". In: *Int. Conf. Neural Inf. Proc. Systems* (cit. on p. 89).

Zhao, Sicheng, Bo Li, Xiangyu Yue, Yang Gu, Pengfei Xu, Runbo Hu, Hua Chai, and Kurt Keutzer (2019). "Multi-source Domain Adaptation for Semantic Segmentation". In: *Int. Conf. Neural Inf. Proc. Systems* (cit. on p. 89).

Zhu, Jun-Yan, Taesung Park, Phillip Isola, and Alexei A. Efros (2017). "Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks". In: *Proc. Int. Conf. Computer Vision* (cit. on pp. 21, 22, 69).

Zhuang, Fuzhen, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He (2020). "A comprehensive survey on transfer learning". In: *Proc. IEEE* (cit. on p. 13).

Zou, Yang, Zhiding Yu, Xiaofeng Liu, B.V.K. Vijaya Kumar, and Jinsong Wang (2019). "Confidence Regularized Self-Training". In: *Proc. Int. Conf. Computer Vision* (cit. on pp. 66, 82).

Zou, Yang, Zhiding Yu, BVK Vijaya Kumar, and Jinsong Wang (2018). "Unsupervised domain adaptation for semantic segmentation via class-balanced self-training". In: *Proc. Europ. Conf. Computer Vision* (cit. on pp. 65, 66, 82).