Synth-to-real semi-supervised learning for visual tasks

Jose Luis Gómez Zurita

Universitat Autònoma de Barcelona

Synth-to-real semi-supervised learning for visual tasks

A dissertation submitted by **Jose Luis Gómez Zurita** at Universitat Autònoma de Barcelona to fulfil the degree of **Doctor of Philosophy**.

Bellaterra, December 19, 2022

This document was typeset by the author using LaTeX 2_ε.

ISBN: 978-84-124793-9-3

Printed by Imprima

Acknowledgements

After five years of intense working and research, I can conclude the last chapter of my academic stage. Of course, I was not alone during all this time and there are a lot of people that I must show my gratitude for their inestimable support.

First, I would like to thank those who made directly possible to accomplish academically this thesis. In order of relevance, first I would like to specially thank my director and tutor Dr. Antonio Lopez, I have been with you since my degree thesis in Computer Science and basically, you marked my entire academic career introducing me to the autonomous driving field. Even with your tight schedule, you had time to support and guide me in all the work done. Next, I would like to thank Dr. Gabriel Villalonga, I followed your steps working in the same topics, we shared research lines and worked hand by hand. Without any doubt, you were key in my development as a researcher. In addition, I would like to thank Dr. Jose Antonio Iglesias and their team in Galicia. With you and your team, and the dataset that you made, I was able to finish my last chapter of this thesis. Lastly, I would like to thank Dr. German Ros, you are direct, exigent and also a well of knowledge. You helped me expand my knowledge in several topics.

Next, I would like to thank the people from the Computer Vision Center (CVC) with who I had fun and exchanged a lot of knowledge. Starting with the members of the ADAS group that I shared my time and exchange opinions, Dr. Joan Serrat, Dra. Idoia Ruiz, Diego Porres and Yi Xiao, you have my gratitude. And of course, all the other people from the group that we shared some time together like the SYNTHIA and CARLA members, thanks to all of you. Moreover, there are a lot more people that I am grateful from the CVC. I want to specially mention Dr. Fei Yang and Dr. Kai Wang, I shared a lot of funny moment with you, and you made more enjoyable my Ph.D., thank you. Also, I would like to thank Dr. Andres Mafla, we shared wonderful gym sessions. In general, I am grateful with all the CVC staff, IT and Ph.D. students that I shared my time with.

Finally, outside the academic and the research circle, I would like to thank my wife first. You made possible my last step of my academic life, supporting and helping me in home. Next, my three cats, that I love and allow me to release all my stress petting them. Finally, my group of close friends with I share my hobbies and good moments outside the academic world.

Abstract

The curse of data labeling is a costly bottleneck in supervised deep learning, where large amounts of labeled data are needed to train intelligent systems. In onboard perception for autonomous driving, this cost corresponds to the labeling of raw data from sensors such as cameras, LiDARs, RADARs, etc. Therefore, synthetic data with automatically generated ground truth (labels) has aroused as a reliable alternative for training onboard perception models. However, synthetic data commonly suffers from synth-to-real domain shift, i.e., models trained on the synthetic domain do not show their achievable accuracy when performing in the real world. This shift needs to be addressed by techniques falling in the realm of domain adaptation (DA).

The semi-supervised learning (SSL) paradigm can be followed to address DA. In this case, a model is trained using source data with labels (here synthetic) and leverages minimal knowledge from target data (here the real world) to generate pseudo-labels. These pseudo-labels help the training process to reduce the gap between the source and the target domains. In general, we can assume accessing both, pseudo-labels and a few amounts of human-provided labels for the target-domain data. However, the most interesting and challenging setting consists in assuming that we do not have human-provided labels at all. This setting is known as unsupervised domain adaptation (UDA). This PhD focuses on applying SSL to the UDA setting, for onboard visual tasks related to autonomous driving.

We start by addressing the synth-to-real UDA problem on onboard vision-based object detection (pedestrians and cars), a critical task for autonomous driving and driving assistance. In particular, we propose to apply an SSL technique known as co-training, which we adapt to work with deep models that process a multi-modal input. The multi-modality consists of the visual appearance of the images (RGB) and their monocular depth estimation. The synthetic data we use as the source domain contains both, object bounding boxes and depth information. This prior knowledge is the starting point for the co-training technique, which iteratively labels unlabeled real-world data and uses such pseudo-labels (here bounding boxes with an assigned object class) to progressively improve the labeling results. Along this process, two models collaborate to automatically label the images, in a way that one model compensates for the errors of the other, so avoiding error drift. While this automatic labeling process is done offline, the resulting pseudo-labels can be used to train object detection models that must perform in real-time onboard a vehicle. We show that multi-modal co-training improves the labeling results compared to single-modal co-training, remaining competitive compared to human labeling.

Given the success of co-training in the context of object detection, we have also adapted this technique to a more crucial and challenging visual task, namely, onboard semantic segmentation. In fact, providing labels for a single image can take from 30 to 90 minutes for a human labeler, depending on the content of the image. Thus, developing automatic labeling techniques for this visual task is of great interest to the automotive industry. In particular, the new co-training framework addresses synth-to-real UDA by an initial stage of self-training. Intermediate models arising from this stage are used to start the co-training procedure, for which we have elaborate an accurate collaboration policy between the two models performing the automatic labeling. Moreover, our co-training seamlessly leverages datasets from different synthetic domains. In addition, the co-training procedure is agnostic to the loss function used to train the semantic segmentation models which perform the automatic labeling. We achieve state-of-the-art results on publicly available benchmark datasets, again, remaining competitive compared to human labeling.

Finally, on the ground of our previous experience, we have designed and implemented a new SSL technique for UDA in the context of visual semantic segmentation. In this case, we mimic the labeling methodology followed by human labelers. In particular, rather than labeling full images at a time, categories of semantic classes are defined and only those are labeled in a labeling pass. In fact, different human labelers can become specialists in labeling different categories. Afterward, these per-category-labeled layers are combined to provide fully labeled images. Our technique is inspired by this methodology since we perform synth-to-real UDA per category, using the self-training stage previously developed as part of our co-training framework. The pseudo-labels obtained for each category are finally fused to obtain fully automatically labeled images. In this context, we have also contributed to the development of a new photo-realistic synthetic dataset based on path-tracing rendering. Our new SSL technique seamlessly leverages publicly available synthetic datasets as well as this new one to obtain state-of-the-art results on synth-to-real UDA for semantic segmentation. We show that the new dataset allows us to reach better labeling accuracy than previously existing datasets, at the same time that it complements well them when combined. Moreover, we also show that the new human-inspired SSL technique outperforms co-training.

Resumen

El etiquetado de datos supone un cuello de botella en el aprendizaje profundo supervisado, donde grandes cantidades de datos etiquetados son necesarios para entrenar sistemas inteligentes. En percepción aplicada a la conducción autónoma, este coste corresponde a etiquetar datos sin tratar procedentes de sensores como cámaras, LiDARs, RADARs, etc. En consecuencia, los datos sintéticos con etiquetas generadas automáticamente han surgido como alternativa para entrenar modelos de percepción. Sin embargo, los datos sintéticos sufren comúnmente de discrepancias de dominio entre datos sintéticos y reales. Estas discrepancias necesitan ser abordadas por técnicas presentes en el ámbito de la adaptación de dominio (DA).

El paradigma del aprendizaje semi-supervisado (SSL) puede ser usado para resolver DA. En este caso, un modelo es entrenado usando datos de partida etiquetados (en nuestro caso datos sintéticos) y aprovecha el mínimo conocimiento posible de los datos objetivo (en nuestro caso datos reales) para generar pseudo-etiquetas. Estas pseudo-etiquetas ayudan al proceso de entrenamiento reduciendo la distancia entre los dominios sintético y real. En general, podemos asumir el acceso a ambos, tanto a pseudo-etiquetas como a pequeñas cantidades de datos objetivo anotados por humanos. Sin embargo, el escenario más interesante y desafiante consiste en asumir que no tenemos acceso a etiqueta humana alguna. Este escenario es conocido como adaptación de dominio sin supervisión (UDA). Esta tesis se centra en aplicar SSL en UDA para tareas visuales destinadas a la conducción autónoma.

Empezamos abordando el problema de sintético a real en UDA para detección de objetos (peatones y coches) en sistema de visión a bordo, que es una tarea crítica en conducción autónoma y sistemas de conducción asistida. En particular, proponemos la aplicación de una técnica de SSL conocida como co-training (entrenamiento cooperativo), el cual adaptamos para trabajar con modelos profundos que procesan datos de entrada multi-modo. La multi-modalidad consiste en la apariencia visual de imágenes (RGB) y su estimación monocular de profundidad. Los datos sintéticos que usamos como dominio de origen contienen ambos, información de profundidad y anotaciones de objetos encuadrados. Este conocimiento a priori es el punto de partida de la técnica de co-training, que iterativamente etiqueta datos reales sin etiquetar (pseudo-etiquetas) y las usa (en este caso cuadrículas alrededor de objetos con clase asignada) para progresivamente mejorar el resultado del etiquetado. A lo largo de este proceso, dos modelos colaboran para etiquetar automáticamente las imágenes, de modo que un modelo compensa las carencias del otro y viceversa, evitando propagación de errores. Mientras que este proceso automático de etiquetado no se hace en caliente, las pseudo-etiquetas resultantes pueden ser usadas para entrenar modelos de detección de objetos que rinden a tiempo real a bordo de un vehículo. Además, mostramos que el co-training multi-modo mejora la etiquetación en comparación al modo único (solo una vista RGB), manteniéndose competitivo con la etiquetación por humanos.

Debido al éxito del co-training en detección de objetos, también hemos adaptado esta técnica a una tarea visual más crucial y desafiante, llamada segmentación semántica. De hecho, etiquetar una sola imagen puede llevar de 30 a 90 minutos para un anotador humano, dependiendo del contenido de la imagen. Por este motivo, desarrollar técnicas de etiquetado automatizado para tareas visuales es de gran interés para la industria automovilística. En particular, el nuevo framework de co-training aborda sintético a real en UDA mediante una fase inicial de auto etiquetado. Modelos intermedios son creados a partir de esta fase que se utilizan para empezar el proceso de co-training, para el cual hemos elaborado una política de colaboración entre los dos modelos que realizan el etiquetado automático. Nuestro método de co-training aprovecha perfectamente datasets de diferentes dominios sintéticos. Además, este método es agnóstico a la función de coste usada para entrenar modelos de segmentación semántica que realizan la anotación automáticamente. Finalmente, mostramos que conseguimos el estado del arte en datasets disponibles públicamente y seguimos mostrando que nos mantenemos competitivos con el etiquetado humano.

Finalmente, con la experiencia obtenida previamente, hemos diseñado e implementado un nuevo método de SSL para UDA en el contexto de la segmentación semántica. En este caso, imitamos la metodología de etiquetado que utilizaría un humano. En particular, en vez de etiquetar toda la imagen de golpe, definimos categorías de clases semánticas y solo estas son etiquetadas de una pasada. De hecho, diferentes etiquetadores humanos se pueden convertir en especialistas para etiquetar diferentes categorías. A continuación, estas capas etiquetadas son combinadas para obtener una etiquetación global de las imágenes. Nuestra técnica se inspira en esta metodología debido a que aplicamos el marco de sintético a real en UDA a cada categoría, usando la etapa del self-training previamente desarrollada como parte de nuestro framework de co-training. Las pseudo-etiquetas obtenidas para cada categoría son finalmente fusionadas para obtener automáticamente la imagen totalmente etiquetada. En este contexto, también hemos contribuido al desarrollo de un nuevo dataset foto-realista de imágenes sintéticas renderizado con path-tracing. Nuestro nuevo método de SSL aprovecha perfectamente datasets sintéticos disponibles públicamente junto al nuestro para obtener el estado del arte en resultados para UDA de sintético a real para segmentación semántica. Mostramos que nuestro nuevo dataset nos permite alcanzar mejor precisión en el etiquetado que con previos datasets existentes, al mismo tiempo que los complementa adecuadamente cuando los combinamos. Además, también demostramos que nuestra nueva técnica SSL inspirada en humanos supera al co-training.

Resum

L'etiquetatge de dades suposa una limitació en l'aprenentatge profund supervisat, on grans quantitats de dades etiquetades són necessàries per entrenar sistemes intel·ligents. En percepció aplicada a la conducció autònoma, aquest cost correspon a etiquetar dades sense tractar procedents de sensors com càmeres, LiDARS, RADARs, etc. En conseqüència, les dades sintètiques amb etiquetes generades automàticament han sorgit com a alternativa per entrenar models de percepció. No obstant això, les dades sintètiques pateixen habitualment de discrepàncies de domini entre dades sintètiques i reals. Aquestes discrepàncies necessiten ser abordades per tècniques presents en l'àmbit de l'adaptació de domini (DA).

El paradigma de l'aprenentatge semisupervisat (SSL) pot ser utilitzat per resoldre DA. En aquest cas, un model és entrenat usant dades d'inici etiquetades (en el nostre cas dades sintètiques) i aprofita el mínim coneixement possible de les dades objectiu (en el nostre cas dades reals) per generar pseudo-etiquetes. Aquestes pseudo-etiquetes ajuden al procés d'entrenament reduint la distancia entre els dominis sintètic i real. En general, podem assumir l'accés ambdós, tant a pseudo-etiquetes com a petites quantitats de dades objectiu etiquetades per humans. No obstant això, l'escenari més interessant i desafiant consisteix a assumir que no tenim accés a cap etiqueta humana. Aquest escenari és conegut com a adaptació de domini sense supervisió (UDA). Aquesta tesi se centra a aplicar SSL en UDA per tasques visuals destinades a la conducció autònoma.

Comencem adreçant el problema de sintètic a real en UDA per detecció d'objectes (vianants i cotxes) en sistemes de visió a bord, que és una tasca crítica en conducció autònoma i sistemes de conducció assistida. En particular, proposem l'aplicació d'una tècnica de SSL coneguda com a co-training (entrenament cooperatiu), el qual adaptem per treballar amb models profunds que processen dades d'entrada multimode, La multimodalitat consisteix en l'aparença visual d'imatges (RGB) i l'estimació monocular de profunditat. Les dades sintètiques que utilitzem com a domini d'origen contenen ambdós, informació de profunditat i etiquetes d'objectes enquadrats. Aquest coneixement previ és el punt d'inici de la tècnica de co-training, que iterativament etiqueta dades reals sense etiquetar (pseudo-etiquetes) i les utilitza (en aquest cas quadrícules al voltant d'objectes amb classe assignada) progressivament per millorar el resultat de l'etiquetatge. Durant el transcurs d'aquest procés, dos models col·laboren per etiquetar automàticament les imatges, de mode que un model compensa les carències de l'altre i al revés, evitant propagació d'errors. Mentre que aquest procés automàtic d'etiquetatge no es fa en calent, les pseudo-etiquetes resultants poden ser utilitzades per entrenar models de detecció d'objectes que rendeixen a temps real a bord d'un vehicle. A més a més, mostrem que el co-training multi-mode millora l'etiquetació en comparació al mode únic (només vista RGB), mantenint-se competitiu amb l'etiquetació per humans.

Gràcies a l'èxit del co-training en detecció d'objectes, també hem adaptat aquesta tècnica a una tasca visual més crucial i desafiant, anomenada segmentació semàntica. De fet, un humà pot trigar a etiquetar una sola imatge de 30 a 90 minuts, depenent del contingut d'aquesta. Per aquest motiu, desenvolupar tècniques d'etiquetatge automatitzat per a tasques visuals és de gran rellevància per a la indústria automobilística. En particular, el nou framework de co-training adreça sintètic a real en UDA per mitjà d'una fase inicial d'auto etiquetatge. Models intermedis són creats a partir d'aquesta fase que s'utilitza per començar el procés de co-training, pel qual hem elaborat una política de col·laboració entre tots dos models que realitzen l'etiquetatge automàtic. A més a més, aquest mètode és agnòstic a la funció de cost utilitzada per entrenar models de segmentació semàntica que realitzen etiquetatges automàticament. Finalment, mostrem que aconseguim l'estat de l'art en datasets disponibles públicament i seguim mostrant que ens mantenim competitius amb l'etiquetatge humà.

Finalment, amb l'experiència obtinguda prèviament, hem dissenyat i implementat un nou mètode de SSL per UDA en el context de la segmentació semàntica. En aquest cas, imitem la metodologia d'etiquetatge que faria servir un humà. En particular, en comptes d'etiquetar tota la imatge de cop, definim categories de classes semàntiques i tan sols etiquetem aquestes d'una passada. De fet, diferent etiquetadors humans es poden convertir en experts per etiquetar diferents categories. A continuació, aquestes capes etiquetades són combinades per assolir un etiquetatge global de les imatges. Aquesta tècnica s'inspira en aquesta metodologia perquè apliquem UDA en el marc de sintètic a real a cada categoria, utilitzant l'etapa de self-training prèviament desenvolupada com a part del nostre framework de co-training. Les pseudo-etiquetes obtingudes per cada categoria són finalment fusionades per obtenir automàticament la imatge totalment etiquetada. En aquest context, també hem contribuït al desenvolupament d'un nou dataset foto-realista d'imatges sintètiques renderitzades amb path-tracing. El nostre mètode de SSL aprofita perfectament datasets sintètics disponibles públicament junts al nostre, per assolir l'estat de l'art en resultats en UDA de sintètic a real per segmentació semàntica. Mostrem que el nostre nou dataset ens permet assolir millor precisió en l'etiquetatge que amb previs datasets existents, al mateix temps que els complementa adequadament quan els combinem. A més a més, també demostrem que la nostra nova tècnica de SSL inspirada en humans supera al co-training.

Contents

List of Figures

List of Tables

1 Introduction

Since the industrial revolution, humanity advanced technologically from baby steps to giant strides. The mechanization introduced to alleviate hand labor and improve production processes was propagated to our daily lives affecting transportation, sanitation, housing, etc. Furthermore, the digital revolution was the milestone that allowed us to start automatizing tasks effectively in every aspect of our lives. The most iconic tool created to automatize tasks was the computer. Computers allow us to solve complex problems which would be impossible to solve by hand with a reasonable effort and in a reasonable time. In addition, with electronic sensors, we can parameterize the inputs required by a computer to automatize tasks. Since the introduction of the first computers and sensors to nowadays, the improvements are astonishing. In the beginning, entire rooms were required by computers to execute basic tasks. Advances in manufacturing electronics enabled reducing computer size while increasing their performance. These improvements opened new doors in every aspect of our lives, from our houses (air conditioning, smartphones, televisions, etc) to our work (industrial manufacturing, administrative procedures, etc).

It is not surprising that with the irruption of computers in our society, a branch dedicated to the study of their applications and development was created. We refer to *Computer Science* (CS). Nowadays, this is a very broad field with many branches. This Ph.D. dissertation falls in the CS branch known as Artificial Intelligence (AI) [117]. Broadly speaking, AI studies how to automatize human intelligence (strong AI), or at least approach it in specific domains (weak IAs). One important aspect of intelligent behaviors is the capacity to learn from data or experiences, as living beings do (genetic heritage, individual learning). Accordingly, we can find Machine Learning (ML) [69] as a branch of AI. This Ph.D. is highly circumscribed in the ML branch. In addition, we will focus on vision-based perception for autonomous driving and driver assistance. Thus, Computer Vision (CV) [23], another branch of IA, is essential in this Ph.D. In fact, nowadays, CV tasks are performed by training so-called deep models (e.g., neural networks, transformers), where Deep Learning (DL) [56] is a branch of ML. In short, we will use DL models to perform CV tasks

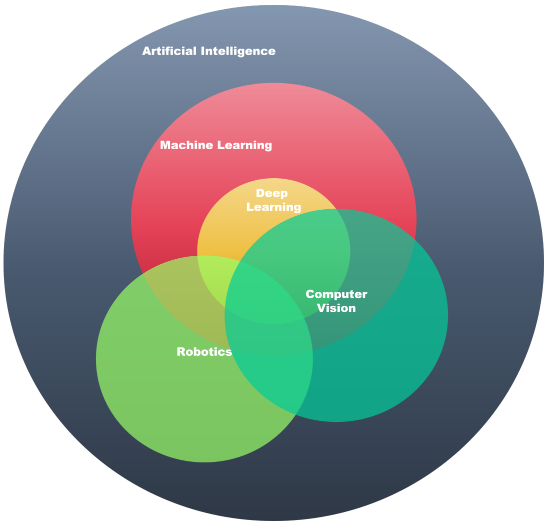


Figure 1.1: Computer science topics most related to this Ph.D.

(e.g., object detection and semantic segmentation) and general ML to elaborate procedures that aim at simplifying the training of such deep models. The mentioned application context of this Ph.D., i.e., autonomous driving and driver assistance can be considered as falling in the field of Robotics. Figure 1.1 summarizes how the aforementioned branches are related.

1.1 Scene understanding in autonomous driving

The necessity to automatize tasks embraces our daily life and one of the most challenging necessities to automatize is transportation. As we are aware, urban and metropolitan areas expand at a high pace due to the technological development of societies. Hence, traveling distances increase making the citizens the necessity to use a transportation system. A large number of transport vehicles becomes an important issue because increases the accident rate [115], travel times due to traffic jams, and pollution. Furthermore, the human factor is the principal cause of accidents while driving due to distractions, inadequate experience, mental state, or unexpected situations. Nowadays, these reasons generate the necessity to increase security while driving and are one of the fundamental aspects that the automotive industry is working on and improving constantly. Step by step, better

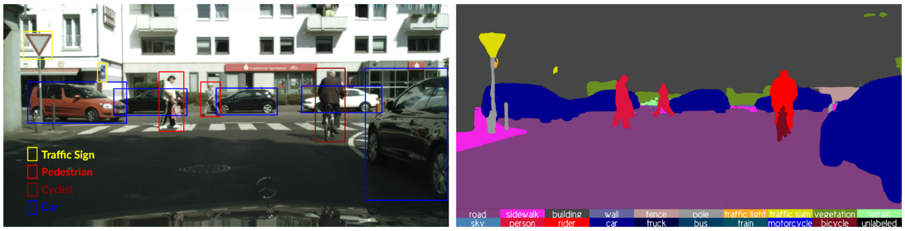


Figure 1.2: Example of a 2D object detection displayed on the RGB image on the left and its respective semantic segmentation prediction on the right.

systems are introduced in vehicles to improve, on the one hand, security from direct accidents and protect their occupants and, on the other hand, reduce the possibility of accidents. In addition, more traffic elements and regulations in urban environments improve cohesion between drivers and pedestrians and lower the danger while driving. Still, accidents continue to happen due to human factors. Hence, the best solution to the aforementioned issues would be to replace the human factor and completely automatize transportation with autonomous driving vehicles [118].

Unfortunately, the current technology does not allow us to achieve complete autonomous driving yet. Nevertheless, we are going in a good direction where companies in collaboration with research centers are developing new systems and techniques to fulfill this objective. In fact, the scope of this dissertation is to help industries with novel techniques that would help onboard visual perception for autonomous driving.

In order to drive autonomously, a multi-sensor suite is needed to perceive the surrounding environment. Examples of sensors are RGB cameras placed in front of the ego vehicle and around to cover blind spots and obtaining visual information of the environment; RADAR and LIDARs sensors that cover 360 degrees around the ego vehicle, providing the distance to other traffic participants and the velocity of vehicles (RADAR); IMUs providing heading and displacement information from the ego vehicle; and GNSS systems for geolocalization. All these data can be used to generate an AI able to drive the ego vehicle. However, the complexity of the information received makes this task really challenging. You need to follow the traffic rules, maintain your vehicle on the right path, and respect the rest of the traffic participants. The most difficult part of the task is to avoid accidents in front of unexpected situations, e.g., unexpected elements in your path, other vehicles not following traffic rules, sensors not working, errors on input data, etc.

As we observe, there are several aspects to take into account that are determined with a proper scene understanding, which is the capacity of the vehicle to perceive and understand the intentions and state of all the elements that are involved in the surrounding traffic scenario. In our opinion, one of the most important type of sensor to obtain a proper scene understanding are RGB cameras, others give support to specific tasks and are complementary to the system as a security measure. The RGB camera sensors are the most similar to the human eyes. Hence, CV methods would be necessary for a proper scene understanding. Among the numerous applications of CV, this Ph.D. dissertation focus on two key tasks to perform onboard vision-based perception for autonomous driving, namely, object detection [62] and semantic segmentation [17] (see Figure 1.2).

On the one hand, object detection aims to identify elements of interest inside an image or sequence and frame them as precisely as possible in a rectangular bounding box (BB). This task comprises the localization of an object inside the BB and also the proper classification of the detected object in case of multiple classes. We observe how this task helps in scene understanding because is able to highlight objects of interest related to autonomous driving. To implement this method we use convolutional neural network (CNN) [51] architectures composed by a backbone [20, 38, 40, 96, 98, 126, 135] to extract feature descriptors from the images and a head to perform the object detection [61, 63, 80–83]. On the other hand, semantic segmentation aims to assign each pixel of an image to a class. This task is complex in comparison to object detection due to the necessity to classify each of the pixels of the whole image, where normally a high amount of categories are involved and is dependent on the image size. Analogous to object detection, we implement this method using the same CNN backbones and a head to apply semantic segmentation [3, 11, 65, 132, 134, 139].

1.2 Synth-to-real domain adaptation

The aforementioned tasks, where we use CNNs to address them, need large amounts of labeled data to train successfully a model [55]. The most common causes of a model performing poorly in these tasks are due to inadequate data, e.g. , insufficient amounts, variability, inadequate labels, different domains, etc. In autonomous driving, the vehicle needs to respond properly to innumerable amounts of situations and environments. Thus, datasets incorporating the maximum possible scenarios are required to train adequate vision-based perception models. Note that, in this field, any error could be fatal, causing an accident. In our case, the most desirable data would be from urban scenes composed of all the traffic elements, related to the geographic area to apply autonomous driving (this could be all over the world), with different weather and illumination conditions. As we observe, trying to supply

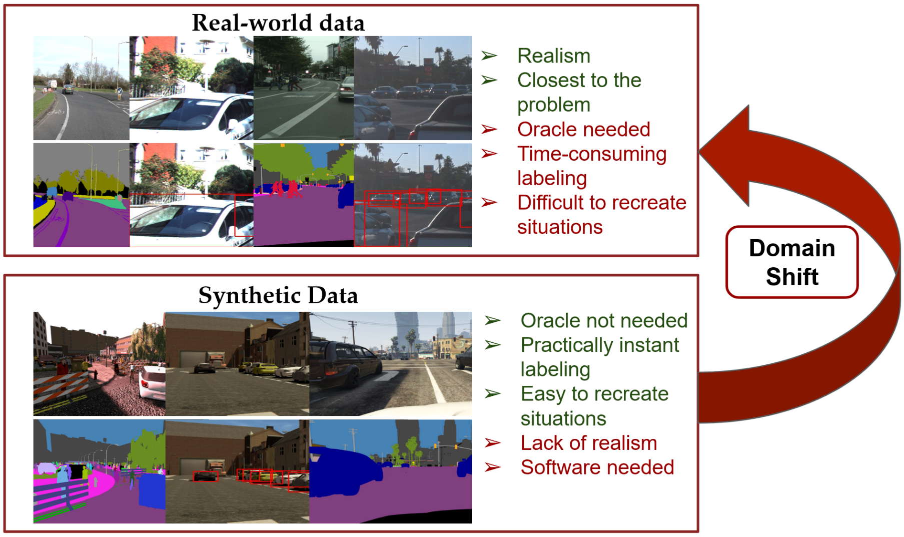


Figure 1.3: Summary of the properties of acquiring real and synthetic data and a visual representation of the notorious domain shift existent between both.

these data demands in the real world is going to be challenging. One of the principal problems, in gathering new real data, is the time needed by a human oracle to label properly the information captured by the sensors (labeling one image could range from 15 to 60 minutes). In addition, these labeled data have a certain bias due to the human factor. Another problem is to recreate situations of interest, where most of them depend on random factors (e.g. , weather conditions, uncommon vehicles, near accident cases, etc.). Thus, obtaining real-world data, oriented to autonomous driving, is time-consuming and expensive [15, 28, 29, 70, 99, 131]. Synthetic data [25, 85, 86, 88, 101, 119] are an alternative of great interest that could overcome the stated problems and is the focus of this dissertation. Synthetic data have the properties of being easy to generate having practically no limits in amounts, with labels obtained almost instantly (current hardware takes a few seconds or less) without the need for a task-oriented human labeler. Moreover, we can create specific scenes of interest useful to address our task, which may be difficult to acquire in the real world. Figure 1.3 summarizes the aforementioned properties of real-world and synthetic data. Nevertheless, synthetic data are still far from representing perfectly realistic scenarios compared to real-world data. Even when actual software tools generate more and more photo-realistic images, creating rich scenarios similar to the real world is not trivial. Hence, still, a noticeable difference between real and synthetic domains exists creating what we call a domain shift.

Domain shifts are present between different datasets, it does not matter if they are real or synthetic, since parameters like different camera resolutions, daytime, weather, environment illumination, etc, provoke domain discrepancies when training a CV model with one dataset and testing it in a different one. Normally speaking, the domain shifts result in poor performance or bad generalization. To address the domain shift problems we apply domain adaptation techniques.

Domain adaptation (DA) [16] techniques aim at increasing the generalization capacity of the CV models to perform in different domains. The most popular scenario is to leverage synthetic data to perform a real-world task (synth-to-real). Hence, a model trained on synthetic data and tested on a real one need to obtain useful results. This dissertation addresses synth-to-real DA by proposing new Semi-supervised learning (SSL) [145] techniques. SSL methods aim at creating automatic labeling systems, i.e., without human intervention. These methods are of great interest to labeling real-world data and address the curse of labeling. Moreover, when in synth-to-real DA we assume that there are no labels from real-world data, the problem is known as unsupervised domain adaptation (UDA) [26], which is the actual setting we are going to face along this Ph.D. dissertation.

1.3 Ph.D. objectives and outline

After introducing all the ingredients involved in automatizing a task as complex as driving, identifying the elements involved, and the main challenges, we are able to properly define the objectives and methodology of this Ph.D. Performing scene understanding is essential for autonomous driving. Hence, we aim at improving two core related tasks, namely, object detection and semantic segmentation. Overall, we focus on their need for supervised data, which we tackle via synthetic data and, consequently, by addressing the synth-to-real UDA challenge. More specifically, **the overall goal of this Ph.D. dissertation is to develop new SSL methods to address the synth-to-real UDA challenge in the context of onboard vision-based perception for autonomous driving, so contributing to automatizing data labeling**. We address our goal along different research lines organized in chapters that are self-contained and follow a paper structure composed of an abstract, introduction, related work, proposed method, experimental results, and conclusions.

In Chapter 2, we design and implement an SSL technique, inspired by the so-called co-training, to obtain automatically labeled object bounding boxes (pseudo-labels) relying on multi-modal data views: appearance (RGB) and depth (D). Our method is purely data-driven, so treats the involved CNN models as a black box, which strives for generality. We demonstrate the effectiveness of such a multi-modal co-training to obtain pseudo-labels and how outperforms single-modal co-training in the standard SSL and the synth-to-real UDA settings.

In Chapter 3, we design and implement a procedure inspired by co-training to address onboard semantic segmentation, i.e., to obtain automatically provide per-pixel semantic labels (pseudo-labels) given an image. Again, we focus on a synth-to-real setting. Again, our procedure treats the involved CNN models as a black box. The results show the effectivity of the generated pseudo-labels, obtaining results close to a human labeler. In addition, we achieve state-of-the-art in several target datasets.

In Chapter 4, we focus on a two-fold contribution to improve the work done for the previous chapter. On the one hand, we collaborate with a team of experts on simulation to generate a new photo-realistic dataset for onboard semantic segmentation. On the other hand, we propose a new SSL procedure to tackle the synth-to-real UDA setting, where we produce pseudo-labels inspired by human labeling protocols. The combination of our new method and dataset produces high-quality pseudo-labels. It improves co-training, enabling the training of semantic segmentation models whose accuracy is really close to the case of using human-provided labels. In other words, obtaining results that have not been seen before in the synth-to-real UDA setting.

Finally, Chapter 5 reviews the work and main contributions of this Ph.D. dissertation, drawing some near-future research actions.

2 Co-training for deep object detection: comparing single-modal and multi-modal approaches

Top-performing computer vision models are powered by convolutional neural networks (CNNs). Training an accurate CNN highly depends on both the raw sensor data and their associated ground truth (GT). Collecting such GT is usually done through human labeling, which is time-consuming and does not scale as we wish. This data-labeling bottleneck may be intensified due to domain shifts among image sensors, which could force per-sensor data labeling. In this chapter, we focus on the use of co-training, a semi-supervised learning (SSL) method, for obtaining pseudo-labeled object bounding boxes (BBs), i.e., the GT to train deep object detectors. In particular, we assess the goodness of multi-modal co-training by relying on two different views of an image, namely, appearance (RGB) and estimated depth (D). Moreover, we compare appearance-based single-modal co-training with multi-modal. Our results suggest that in a standard SSL setting (no domain shift, a few human-labeled data) and under virtual-to-real domain shift (many virtual-world labeled data, no human-labeled data) multi-modal co-training outperforms single-modal. In the latter case, by performing GAN-based domain translation both co-training modalities are on par, at least when using an off-the-shelf depth estimation model not specifically trained on the translated images.

2.1 Introduction

Supervised deep learning enables accurate computer vision models. Key for this success is the access to raw sensor data (i.e., images) with ground truth (GT) for the visual task at hand (e.g., image classification [95], object detection [83] and recognition [100], pixel-wise instance/semantic segmentation [111,124], monocular depth estimation [18], 3D reconstruction [50], etc.). The supervised training of such computer vision models, which are based on convolutional neural networks (CNNs), is known to require very large amounts of images with GT [97]. While, until one decade ago, acquiring representative images was not easy for many computer vision

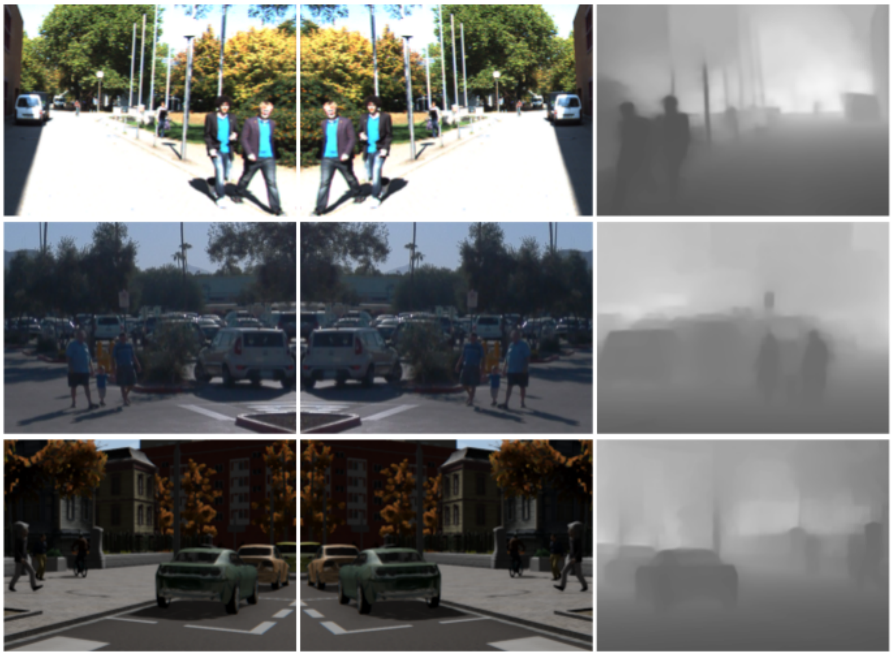


Figure 2.1: From top to bottom: samples from KITTI ($\mathcal{K}$), Waymo ($\mathcal{W}$), and Virtual-world ($\mathcal{V}$) datasets. Middle column: cropped patch from an original image. Left column: horizontal mirror of the original patch. Right column: monocular depth estimation [130] from the original patch. Left-middle columns are the views used for co-training in [110]. Right-middle columns are the views also used in this chapter.

applications (e.g., for onboard perception), nowadays, the bottleneck has shifted to the acquisition of the GT. The reason is that this GT is mainly obtained through human labeling, whose difficulty depends on the visual task. In increasing order of labeling time, we see that image classification requires image-level tags, object detection requires object bounding boxes (BBs), instance/semantic segmentation requires pixel-level instance/class silhouettes, and depth GT cannot be manually provided. Therefore, manually collecting such GT is time-consuming and does not scale as we wish. Moreover, this data labeling bottleneck may be intensified due to domain shifts among different image sensors, which could drive to per-sensor data labeling.

To address the curse of labeling, different meta-learning paradigms are being explored. In self-supervised learning

(SfSL) the idea is to train the desired models with the help of auxiliary tasks related to the main task. For instance, solving automatically generated jigsaw puzzles helps to obtain more accurate image recognition models [54], while stereo and structure-from-motion (SfM) principles can provide self-supervision to train monocular depth estimation models [31]. In active learning (AL) [87, 94], there is a human—model collaborative loop, where the model proposes data labels, known as pseudo-labels, and the human corrects them so that the model learns from the corrected labels too; thus, aiming at a progressive improvement of the model accuracy. In contrast to AL, semi-supervised learning (SSL) [10, 107] does not require human intervention. Instead, it is assumed the availability of a small set of off-the-shelf labeled data and a large set of unlabeled data, and both datasets must be used to obtain a more accurate model than if only the labeled data were used. In SfSL, the model trained with the help of the auxiliary tasks is intended to be the final model of interest. In AL and SSL, it is possible to use any model with the only purpose of pseudo-labeling the data, i.e., producing the pseudo-labels, and then use labels and pseudo-labels for training the final model of interest.

In this chapter we focus on co-training [5, 33], a type of SSL algorithm. Co-training pseudo-labels data through the mutual improvement of two models. These models analyze the unlabeled data according to their different views of these data. Our work focuses on onboard vision-based perception for driver assistance and autonomous driving. In this context, vehicle and pedestrian detection are key functionalities. Accordingly, we apply co-training to significantly reduce human intervention when labeling these objects (in computer vision terminology) for training the corresponding deep object detector. Therefore, the labels are BBs locating the mentioned traffic participants in the onboard images. More specifically, we consider two settings. On the one hand, as is usual in SSL, we assume the availability of a small set of human-labeled images (i.e., with BBs for the objects of interests), and a significantly larger set of unlabeled images. On the other hand, we do not assume human labeling at all, but we have a set of virtual-world images with automatically generated BBs.

This work is the natural continuation of the work presented by Villalonga & López [110]. In this previous work, a co-training algorithm for deep object detection is presented, addressing the two above-mentioned settings too. In [110], the two views of an image consist of the original RGB representation and its horizontal mirror; thus, it is a single-modal co-training based on appearance. However, a priori, the higher difference among data views the more accurate pseudo-labels can be expected from co-training. Therefore, as a major novelty of this work, we explore the use of two image modalities in the role of co-training views. In particular, one view is the appearance (i.e., the original RGB), while the other view is the corresponding depth (D) as estimated by a state-of-the-art monocular depth estimation model [130]. Thus, we term this approach as multi-modal co-training; however, it can still be considered a single-sensor because still relies only on RGB images. Figure 2.1 illustrates these different views for images that we use in our experiments.

In this setting, the research questions that we address are two: (Q1) Is multi-modal (RGB/D) co-training effective on the task of providing pseudo-labeled object BBs?; (Q2) How does perform multi-modal (RGB/D) co-training compared to single-modal (RGB)?. After adapting the method presented in [110] to work with both, the single and the multi-modal data views, we ran a comprehensive set of experiments for answering these two questions. Regarding (Q1), we conclude that, indeed, multi-modal co-training is rather effective. Regarding (Q2), we conclude that in a standard SSL setting (no domain shift, a few human-labeled data) and under virtual-to-real domain shift (many virtual-world labeled data, no human-labeled data) multi-modal co-training outperforms single-modal. In the latter case, when GAN-based virtual-to-real image translation is performed [144] (i.e., as image-level domain adaptation) both co-training modalities are on par; at least, by using an off-the-shelf monocular depth estimation model not specifically trained on the translated images.

We organize the rest of the chapter as follows. Section 2.2 reviews related works. Section 2.3 draws the co-training algorithm. Section 2.4 details our experimental setting, discussing the obtained results in terms of (Q1) and (Q2). Section 2.5 summarizes the presented work, suggesting lines of continuation.

2.2 Related work

As we have mentioned before, co-training falls in the realm of SSL. Thus, here we summarize previous related works applying SSL methods. The input to these methods consists of a labeled dataset, $\mathscr{X}^l$, and an unlabeled one, $\mathscr{X}^u$, with $\#\mathscr{X}^u \gg \#\mathscr{X}^l$ and $\mathscr{D}_{\mathscr{X}^u} = \mathscr{D}_{\mathscr{X}^l}$, where $\#\mathscr{X}$ is the cardinality of the set $\mathscr{X}$ and $\mathscr{D}_{\mathscr{X}}$ refers to the domain from which $\mathscr{X}$ has been drawn. Note that, when the latter requirement does not hold, we are under a domain shift setting. The goal of a SSL method is to use both $\mathscr{X}^l$ and $\mathscr{X}^u$ to allow the training of a predictive model, ϕ, so that its accuracy is higher than if only $\mathscr{X}^l$ is used for its training. In other words, the goal is to leverage unlabeled data.

A classical SSL approach is the so-called self-training, introduced by Yarowsky [128] in the context of language processing. Self-training is an incremental process that starts by training ϕ on $\mathscr{X}^l$; then, ϕ runs on $\mathscr{X}^u$, and its predictions are used to form a pseudo-labeled set $\mathscr{X}^{\hat{l}}$, further used together with $\mathscr{X}^l$ to retrain ϕ. This is repeated until convergence, and the accuracy of ϕ, as well as the quality of $\mathscr{X}^{\hat{l}}$, are supposed to become higher as the cycles progress. Jeong et al. [47] used self-training for deep object detection (on PASCAL VOC and MS-COCO datasets). To collect $\mathscr{X}^{\hat{l}}$, a consistency loss is added while training ϕ, which is a CNN for object detection in this case, together with a mechanism for removing predominant backgrounds. The consistency loss is based on the idea that $\phi(\mathrm{I}^u) \sim \phi(\mathrm{I}^{u^{\Lsh}})^{\Lsh}$, where I^u is an unlabeled image, and " $\Lsh$ " refers to performing horizontal mirroring. Lokhande et al. [64] used self-training for deep image classification. In this case, the original activation functions of ϕ, a CNN for image classification, must be changed to Hermite polynomials. Note that these two examples of self-training involve modifications either in the architecture of ϕ [64] or in its training framework [47]. However, we aim at using a given ϕ together with its training framework as a black box, so performing SSL only at the data level. In this way, we can always benefit from state-of-the-art models and training frameworks, i.e., avoiding changing the SSL approach if those change. In this way, we can also decouple the model used to produce pseudo-labels from the model that would be trained with them for deploying the application of interest.

A major challenge when using self-training is to avoid drifting to erroneous pseudo-labels. Note that, if $\mathscr{X}^{\hat{l}}$ is biased to some erroneous pseudo-labels, when using this set to retrain ϕ incrementally, a point can be reached where $\mathscr{X}^{l}$ cannot compensate the errors in $\mathscr{X}^{\hat{l}}$, and ϕ may end learning wrong data patterns and so producing more erroneous pseudo-labels. Thus, as alternative to the self-training of Yarowsky [128], Blum and Mitchell proposed co-training [5]. Briefly, co-training is based on two models, ϕ_{v_1} and ϕ_{v_2}, each one incrementally trained on different data features, termed as views. In each training cycle, ϕ_{v_1} and ϕ_{v_2} collaborate to form $\mathscr{X}^{\hat{l}} = \mathscr{X}^{\hat{l}}_{v_1} \cup \mathscr{X}^{\hat{l}}_{v_2}$. Where, $\mathscr{X}^{\hat{l}}_{v_i}$ and $\mathscr{X}^{l}$ are used to retrain $\phi_{v_i}, i \in \{1,2\}$. This is repeated until convergence. It is assumed that each view, v_i, is discriminant enough as to train an accurate ϕ_{v_i}. Different implementations of co-training, may differ in the collaboration policy. Our approach follows the disagreement idea introduced by Guz et al. [33] in the context of sentence segmentation, later refined by Tur [106] to address domain shifts in the context of natural language processing. In short, only pseudo-labels of high confidence for ϕ_{v_i} but of low confidence for ϕ_{v_j}, $i, j \in \{1,2\}, i \neq j$, are considered as part of $\mathscr{X}^{\hat{l}}_{v_j}$ in each training cycle. Soon, disagreement-based SSL attracted much interest [143].

In general, ϕ_{v_1} and ϕ_{v_2} can be based on different data views by either training on different data samples ($\mathscr{X}^{\hat{l}}_{v_1} \neq \mathscr{X}^{\hat{l}}_{v_2}$) or being different models (e.g., ϕ_{v_1} and ϕ_{v_2} can be based on two different CNN architectures). The disagreement-based co-training falls in the former case. In this line, Qiao et al. [76] used co-training for deep image classification, where the two different views are achieved by training on mutually adversarial samples. However, this implies linking the training of the ϕ_{v_i}'s at the level of the loss function, while, as we have mentioned before, we want to use these models as black boxes.

The most similar work to this chapter is the co-training framework that we introduced in [110] since we work on top of it. In [110], two single-modal views are considered. These consist of using ϕ_{v_1} to process the original images from $\mathcal{X}^u$ while using ϕ_{v_2} to process their horizontally mirrored counterparts, and analogously for $\mathcal{X}^l$. A disagreement-based collaboration is applied to form $\mathcal{X}^{\hat{l}}_{v_1}$ and $\mathcal{X}^{\hat{l}}_{v_2}$. Moreover, not only the setting where $\mathcal{X}^l$ is based on human labels is considered, but also when it is based on virtual-world data. In the latter case, a GAN-based virtual-to-real image translation [144] is used as pre-processing for the virtual-world images, i.e., before taking them for running the co-training procedure. Very recently, Díaz et al. [21] presented co-training for visual object recognition. In other words, the paper addresses a classification problem, while we address both localization and classification to perform object detection. While the different views proposed in [21] rely on self-supervision (e.g., forcing image rotations), here, these rely on data multi-modality. In fact, in our previous work [110], we used mirroring to force different data views, which can be considered as a kind of self-supervision too. Here, after adapting and improving the framework used in [110], we confront this previous setting to a new multi-modal single-sensor version (Algorithm 1 and Figure 2.2). We focus on the case where ϕ_{v_1} works with the original images while ϕ_{v_2} works with their estimated depth. Analyzing this setting is quite interesting because appearance and depth are different views of the same data.

To estimate depth, we need an out-of-the-shelf monocular depth estimation (MDE) model, so that we can keep the co-training as a single-sensor even being multi-modal. MDE can be based on either LiDAR supervision, or stereo/SfM self-supervision, or combinations; where, both LiDAR and stereo data, and SfM computations, are only required at training time, but not at testing time. We refer to [18] for a review on MDE state-of-the-art. In this chapter, to isolate the multi-modal co-training performance assessment as much as possible from the MDE performance, we have chosen the top-performing supervised method proposed by Yin et al. [130].

Finally, we would like to mention that there are methods in the literature that may be confused with co-training, so it is worth introducing a clarification note. This is the case of the co-teaching proposed by Han et al. [35] and the co-teaching+ of Yu et al. [133]. These methods have been applied to deep image classification to handle noisy labels on $\mathcal{X}^l$. However, citing Han et al. [35], *co-training is designed for SSL, and co-teaching is for learning with noisy (ground truth) labels (LNL); as LNL is not a special case of SSL, we cannot simply translate co-training from one*

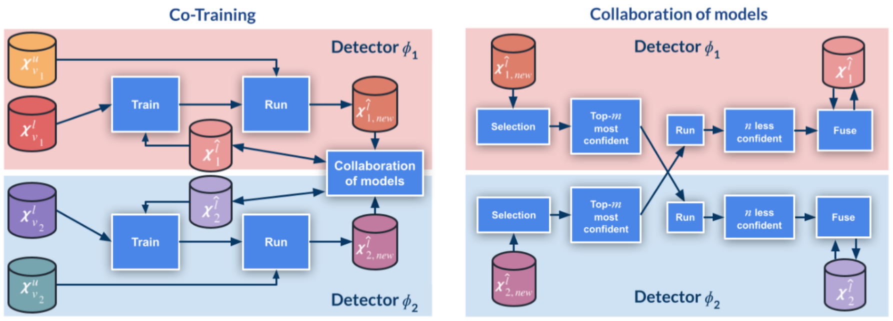


Figure 2.2: Co-training pipeline: the **left** diagram shows the global block structure, while the **right** diagram details the collaboration of models block. Symbols and procedures are based on Algorithm 1. We refer to this algorithm and the main text for a detailed explanation.

problem setting to another problem setting.

2.3 Method

In this section, we explain our co-training procedure with the support of Figure 2.2 and Algorithm 1. Up to a large extent, we follow the same terminology as in [110].
Input: The specific sets

of labeled $(\mathcal{X}^l_{v_1}, \mathcal{X}^l_{v_2})$ and unlabeled $(\mathcal{X}^u_{v_1}, \mathcal{X}^u_{v_2})$ input data in Algorithm 1 determine if we are running on either a single or multi-modal setting. Also, if we are supported or not by virtual-world images or their virtual-to-real translated counterparts. Table 2.1, clarifies the different co-training settings depending on these datasets. In Algorithm 1, view-paired sets means that each image of one set has a counterpart in the other, i.e., following Table 2.1, its horizontal mirror or its estimated depth. Since the co-training is agnostic to the specific object detector in use, we explicitly consider its corresponding CNN architecture, Φ, and training hyper-parameter, $\mathcal{H}_\Phi$, as inputs. Finally, $\mathcal{H}_{ct}$ consists of the co-training hyper-parameters, which we will introduce while explaining the part of the algorithm in which each of them is required.
Output:

It consists in a set of images $(\mathcal{X}^{\hat{l}})$ from $\mathcal{X}^u_{v_1}$, for which co-training is providing pseudo-labels, i.e., object BBs in this work. In our experiments, according to Table 2.1, $\mathcal{X}^u_{v_1}$ always corresponds to the unlabeled set of original real-world images.

Table 2.1: The different configurations that we consider for Algorithm 1 in this work, according to the input datasets. In the single-modal cases, we work only with RGB images (appearance), either from a real-world dataset ($\mathscr{R}_{\text{RGB}}$), or a virtual-world one ($\mathscr{V}_{\text{RGB}}$), or a virtual-to-real domain-adapted one ($\mathscr{V}_{\mathscr{G}_{\mathscr{R}},\text{RGB}}$), i.e., using a GAN-based $\mathscr{V}_{\text{RGB}} \rightarrow \mathscr{R}_{\text{RGB}}$ image translation. One view of the data (v_1) corresponds to the original RGB images of each set, while the other view (v_2) corresponds to their horizontally mirrored counterparts, indicated with the symbol " ↰ ". In the multi-modal cases, view v_1 is the same as for the single-modal case (RGB), while view v_2 corresponds to the depth (D) estimated from the RGB images by using an off-the-shelf monocular depth estimation model.

Modality	**Domain Shift?**	$\mathscr{X}^l_{v_1}$	$\mathscr{X}^l_{v_2}$	$\mathscr{X}^u_{v_1}$	$\mathscr{X}^u_{v_2}$
Single-modal	No	$\mathscr{R}_{\text{RGB}}$	$\mathscr{R}_{\text{RGB}}^{↰}$	$\mathscr{R}_{\text{RGB}}$	
	Yes	$\mathscr{V}_{\text{RGB}}$	$\mathscr{V}_{\text{RGB}}^{↰}$		
	Adapted	$\mathscr{V}_{\mathscr{G}_{\mathscr{R}},\text{RGB}}$	$\mathscr{V}_{\mathscr{G}_{\mathscr{R}},\text{RGB}}^{↰}$		
Multi-modal	No	$\mathscr{R}_{\text{RGB}}$	$\mathscr{R}_{\text{D}}$	$\mathscr{R}_{\text{RGB}}$	$\mathscr{R}_{\text{D}}$
	Yes	$\mathscr{V}_{\text{RGB}}$	$\mathscr{V}_{\text{D}}$		
	Adapted	$\mathscr{V}_{\mathscr{G}_{\mathscr{R}},\text{RGB}}$	$\mathscr{V}_{\mathscr{G}_{\mathscr{R}},\text{D}}$		

Since we consider as output a set of pseudo-labeled images, which complement the input set of labeled images, they can be later used to train a model based on Φ or any other CNN architecture performing the same task (i.e., requiring the same type of BBs).

Initialize:

First, the initial object detection models (ϕ_1, ϕ_2) are trained using the respective views of the labeled data ($\mathscr{X}^l_{v_1}, \mathscr{X}^l_{v_2}$). After their training, these models are applied to the respective views of the unlabeled data ($\mathscr{X}^u_{v_1}, \mathscr{X}^u_{v_2}$). Detections (i.e., object BBs) with a confidence over a threshold are considered pseudo-labels. Since we address a multi-class problem, per-class thresholds are contained in the set T, a hyper-parameter in $\mathscr{H}_{ct}$. The temporary pseudo-labeled sets generated by ϕ_1 and ϕ_2 are $\mathscr{X}^{\hat{l}}_{1,new}$ and $\mathscr{X}^{\hat{l}}_{2,new}$, respectively. At this point no collaboration is produced between ϕ_1 and ϕ_2. In fact, while co-training loops (repeat body), the pseudo-labeled sets resulting from the collaboration are $\mathscr{X}^{\hat{l}}_1$ and $\mathscr{X}^{\hat{l}}_2$, which are initialized as empty. In the training function, $\text{Train}(\Phi, \mathscr{H}_\Phi, \mathscr{S}^l, \mathscr{S}^{\hat{l}}) : \phi$, we use BB labels (in $\mathscr{S}^l$) and BB pseudo-labels (in $\mathscr{S}^{\hat{l}}$) indistinctly. However, we only consider background samples from $\mathscr{S}^l$, since, as co-training progresses, $\mathscr{S}^{\hat{l}}$ may be instantiated with a set of pseudo-labeled images containing false negatives (i.e., undetected objects) which could be erroneously taken as hard negatives (i.e., background quite similar to objects) when training ϕ.

Collaboration:

The two object detection models collaborate by exchanging pseudo-labeled images (Figure 2.2-right). This exchange is inspired in disagreement-based SSL [143]. Our specific approach is controlled by the co-training hyper-parameters N, n, m, and, in case of working with image sequences instead of with sets of isolated images, also by $\mathcal{H}_{seq} = \{\Delta t_1, \Delta t_2\}, \Delta t_1, \Delta t_2$. This approach consists of the following three steps.

(First step)

Each model selects the set of its top-m most confident pseudo-labeled images $(\mathcal{X}^{\hat{l}}_{1,\uparrow}, \mathcal{X}^{\hat{l}}_{2,\uparrow})$; where, the confidence of an image is defined as the average over the confidences of the pseudo-labels of the image, i.e., in our case, over the object detections. Thus, $\mathcal{X}^{\hat{l}}_{i,\uparrow} \subseteq \mathcal{X}^{\hat{l}}_{i,new}, i \in \{1,2\}$. However, for creating $\mathcal{X}^{\hat{l}}_{i,\uparrow}$, we do not consider all the pseudo-labeled images in $\mathcal{X}^{\hat{l}}_{i,new}$. Instead, to minimize bias and favor speed, we only consider N randomly selected images from $\mathcal{X}^{\hat{l}}_{i,new}$. In the case of working with image sequences, to favor variability in the pseudo-labels, the random choice is constrained to avoid using consecutive frames. This is controlled by thresholds Δt_1 and Δt_2; where Δt_1 controls the minimum frame distance between frames selected at the current co-training cycle (k), and Δt_2 among frames at current cycle with respect to frames selected in previous cycles $(< k)$. We apply Δt_1 first, then Δt_2, and then the random selection among the frames passing these constraints.

(Second step) Model ϕ_i processes $\mathcal{X}^{\hat{l}}_{j,\uparrow}, i, j \in \{1,2\}, i \neq j$, keeping the set of the n less confident pseudo-labeled images for it. Thus, we obtain the new sets $\mathcal{X}^{\hat{l}}_{1,\downarrow}$ and $\mathcal{X}^{\hat{l}}_{2,\downarrow}$. Therefore, considering the first and second steps, we see that one model shares with the other those images that it has pseudo-labeled with more confidence, and, of these, each model retains for retraining those that it pseudo-labels with less confidence. Therefore, this step implements the actual collaboration between models ϕ_1 and ϕ_2.

(Third step) The pseudo-labeled sets obtained in previous step $(\mathcal{X}^{\hat{l}}_{1,\downarrow}, \mathcal{X}^{\hat{l}}_{2,\downarrow})$ are fused with those accumulated from previous co-training cycles $(\mathcal{X}^{\hat{l}}_{1}, \mathcal{X}^{\hat{l}}_{2})$. This is done by properly calling the function $\text{Fuse}(\mathcal{S}^{\hat{l}}_{old}, \mathcal{S}^{\hat{l}}_{new}) : \mathcal{S}^{\hat{l}}$ for each view. The returned set of pseudo-labeled images, $\mathcal{S}^{\hat{l}}$, contains $\mathcal{S}^{\hat{l}}_{old} \cup \mathcal{S}^{\hat{l}}_{new} - \mathcal{S}^{\hat{l}}_{old} \cap \mathcal{S}^{\hat{l}}_{new}$, and, from $\mathcal{S}^{\hat{l}}_{old} \cap \mathcal{S}^{\hat{l}}_{new}$, only those pseudo-labeled images in $\mathcal{S}^{\hat{l}}_{new}$ are added to $\mathcal{S}^{\hat{l}}$.

Retrain and update: At this point we have new sets of pseudo-labeled images

$(\mathscr{X}_1^{\hat{l}}, \mathscr{X}_2^{\hat{l}})$, which, together with the corresponding input labeled sets $(\mathscr{X}_{v_1}^{l}, \mathscr{X}_{v_2}^{l})$, are used to retrain the models ϕ_1 and ϕ_2. Afterwards, these new models are used to obtain new temporary pseudo-labeled set $(\mathscr{X}_{1,new}^{\hat{l}}, \mathscr{X}_{2,new}^{\hat{l}})$ through their application to the corresponding unlabeled sets $(\mathscr{X}_{v_1}^{u}, \mathscr{X}_{v_2}^{u})$. Then, co-training can start a new cycle.

Stop: The function $\text{Stop?}(\mathscr{H}_{stp}, \mathscr{S}_{old}^{\hat{l}}, \mathscr{S}_{new}^{\hat{l}}, k) : \text{Boolean}$ determines if a new co-training cycle is executed. This is controlled by the co-training hyper-parameters $\mathscr{H}_{stp} = \{K_{min}, K_{max}, T_{\Delta_{mAP}}, \Delta K\}$. Co-training will execute a minimum of K_{min} cycles and a maximum of K_{max}, being k the current number. The parameters $\mathscr{S}_{old}^{\hat{l}}$ and $\mathscr{S}_{new}^{\hat{l}}$ are supposed to be instantiated with the sets of pseudo-labeled images in previous and current co-training cycles, respectively. The similarity of these sets is monitored in each cycle, so that if its stable for more than ΔK consecutive cycles, convergence is assumed and co-trained stopped. This constrain could already be satisfied at $k = K_{min}$ provided $K_{min} \geq \Delta K$. The metric used to compute the similarity between these pseudo-labeled sets is mAP (mean average precision) [28], where $\mathscr{S}_{old}^{\hat{l}}$ plays the role of GT and $\mathscr{S}_{new}^{\hat{l}}$ the role of results under evaluation. Then, mAP is considered stable between two consecutive cycles if its magnitude variation is below the threshold $T_{\Delta_{mAP}}$.

```
Input  :View-paired sets of labeled images: $\mathcal{X}^l_{v_1}, \mathcal{X}^l_{v_2}$
        View-paired sets of unlabeled images: $\mathcal{X}^u_{v_1}, \mathcal{X}^u_{v_2}$
        Object detection architecture, and its training hyper-parameters: $\Phi, \mathcal{H}_\Phi$
        Co-training hyper-parameters: $\mathcal{H}_{ct} = \{T, N, n, m, \mathcal{H}_{stp}[, \mathcal{H}_{seq}]\}$
Output:New labeled images: $\mathcal{X}^{\hat{l}} \subseteq \mathcal{X}^u_{v_1}$
// Initialize models and working datasets.
  $<\mathcal{X}^{\hat{l}}_1, \mathcal{X}^{\hat{l}}_2, k> \leftarrow <\emptyset, \emptyset, 0>$
  $\phi_1, \phi_2 \leftarrow \mathrm{Train}(\Phi, \mathcal{H}_\Phi, \mathcal{X}^l_{v_1}, \mathcal{X}^{\hat{l}}_1), \mathrm{Train}(\Phi, \mathcal{H}_\Phi, \mathcal{X}^l_{v_2}, \mathcal{X}^{\hat{l}}_2)$
  $\mathcal{X}^{\hat{l}}_{1,new}, \mathcal{X}^{\hat{l}}_{2,new} \leftarrow \mathrm{Run}(\phi_1, \mathcal{X}^u_{v_1}, T), \mathrm{Run}(\phi_2, \mathcal{X}^u_{v_2}, T)$
repeat
    $\mathcal{X}^{\hat{l}}_{old} \leftarrow \mathcal{X}^{\hat{l}}_{1,new}$
    // Collaboration of models.
    $\mathcal{X}^{\hat{l}}_{1,\uparrow} \leftarrow \mathrm{Slct}(\uparrow, m, \mathrm{Rnd}(\mathcal{X}^{\hat{l}}_{1,new}, N[, \mathcal{H}_{seq}, k]))$
    $\mathcal{X}^{\hat{l}}_{2,\uparrow} \leftarrow \mathrm{Slct}(\uparrow, m, \mathrm{Rnd}(\mathcal{X}^{\hat{l}}_{2,new}, N[, \mathcal{H}_{seq}, k]))$
    $\mathcal{X}^{\hat{l}}_{1,\downarrow}, \mathcal{X}^{\hat{l}}_{2,\downarrow} \leftarrow \mathrm{Slct}(\downarrow, n, \mathrm{Run}(\phi_1, \mathcal{X}^{\hat{l}}_{2,\uparrow}, T)), \mathrm{Slct}(\downarrow, n, \mathrm{Run}(\phi_2, \mathcal{X}^{\hat{l}}_{1,\uparrow}, T))$
    $\mathcal{X}^{\hat{l}}_1, \mathcal{X}^{\hat{l}}_2 \leftarrow \mathrm{Fuse}(\mathcal{X}^{\hat{l}}_1, \mathcal{X}^{\hat{l}}_{1,\downarrow}), \mathrm{Fuse}(\mathcal{X}^{\hat{l}}_2, \mathcal{X}^{\hat{l}}_{2,\downarrow})$
    // Retrain models and Update datasets.
    $\phi_1, \phi_2 \leftarrow \mathrm{Train}(\Phi, \mathcal{H}_\Phi, \mathcal{X}^l_{v_1}, \mathcal{X}^{\hat{l}}_1), \mathrm{Train}(\Phi, \mathcal{H}_\Phi, \mathcal{X}^l_{v_2}, \mathcal{X}^{\hat{l}}_2)$
    $\mathcal{X}^{\hat{l}}_{1,new}, \mathcal{X}^{\hat{l}}_{2,new} \leftarrow \mathrm{Run}(\phi_1, \mathcal{X}^u_{v_1}, T), \mathrm{Run}(\phi_2, \mathcal{X}^u_{v_2}, T)$
until $Stop?(\mathcal{H}_{stp}, \mathcal{X}^{\hat{l}}_{old}, \mathcal{X}^{\hat{l}}_{1,new}, k{+}{+})$
  $\mathcal{X}^{\hat{l}} \leftarrow \mathcal{X}^{\hat{l}}_{1,new}$
return $\mathcal{X}^{\hat{l}}$
```

Algorithm 1: pseudo-labeling of object BBs by co-training.

2.4 Experimental results

2.4.1 Datasets and evaluation protocol

We follow the experimental setup of [110]. Therefore, we use KITTI [28] and Waymo [99] as real-world datasets, here denoted as $\mathcal{K}$ and $\mathcal{W}$, respectively. We use a variant of the SYNTHIA dataset [86] as virtual-world data, here denoted as $\mathcal{V}$. For $\mathcal{K}$ we use Xiang et al. [123] split, which reduces the correlation between training and testing data. While this implies that $\mathcal{K}$ is formed by isolated images, $\mathcal{W}$ is composed of image sequences. To align its acquisition conditions with $\mathcal{K}$, we consider daytime sequences without adverse weather. From them, as recommended in [99], we

Table 2.2: Datasets ($\mathcal{X}$): train ($\mathcal{X}^{tr}$) and test ($\mathcal{X}^{tt}$) statistics, $\mathcal{X} = \mathcal{X}^{tr} \cup \mathcal{X}^{tt}, \mathcal{X}^{tr} \cap \mathcal{X}^{tt} = \emptyset$.

Dataset ($\mathcal{X}$)	$\mathcal{X}^{tr}$			$\mathcal{X}^{tt}$		
	Images	**Vehicles**	**Pedestrians**	**Images**	**Vehicles**	**Pedestrians**
Virtual ($\mathcal{V}$)	19,791	43,326	44,863			
KITTI ($\mathcal{K}$)	3682	14,941	3154	3799	18,194	1333
Waymo ($\mathcal{W}$)	9873	64,446	9918	4161	24,600	3068

randomly select some sequences for training and the rest for testing. Furthermore, we adapt $\mathcal{W}$'s image size to match $\mathcal{K}$ (i.e., 1240 × 375 pixels) by first eliminating the top rows of each image so avoiding large sky areas, and then selecting a centered area of 1240 pixel width. The 2D BBs of $\mathcal{W}$ and $\mathcal{V}$, are obtained by projecting the available 3D BBs. On the other hand, $\mathcal{V}$ is generated by mimicking some acquisition conditions of $\mathcal{K}$, such as image resolution, non-adverse weather, daytime, and only considering isolated shots instead of image sequences. Besides, $\mathcal{V}$'s images include standard visual post-effects such as anti-aliasing, ambient occlusion, depth of field, eye adaptation, blooming, and chromatic aberration. In the following, we term as $\mathcal{K}^{tr}$ and $\mathcal{K}^{tt}$ the training and testing sets of $\mathcal{K}$, respectively. Analogously, $\mathcal{W}^{tr}$ and $\mathcal{W}^{tt}$ are the training and testing sets of $\mathcal{W}$. For each dataset, Table 2.2 summarizes the number of images and object BBs (vehicles and pedestrians) used for training and testing our object detectors. Note that $\mathcal{V}$ is only used for training purposes.

We apply the KITTI benchmark protocol for object detection [28]. Furthermore, following [110], we focus on the so-called moderate difficulty, which implies that the minimum BB height to detect objects is 25 pixels for $\mathcal{K}$ and 50 pixels for $\mathcal{W}$. Once co-training finishes, we use the labeled data ($\mathcal{X}^{l}$) and the data pseudo-labeled by co-training ($\mathcal{X}^{\hat{l}}$) to train the final object detector, namely, ϕ_F. Since this is the ultimate goal, we use the accuracy of such a detector as metric to evaluate the effectiveness of the co-training procedure. If it performs well at pseudo-labeling objects, the accuracy of ϕ_F should be close to the upper-bound (i.e., when the 100% of the real-world labeled data used to train ϕ_F is provided by humans), otherwise, the accuracy of ϕ_F is expected to be close to the lower-bound (i.e., when using either a small percentage of human-labeled data or only virtual-world data to train ϕ_F).

2.4.2 Implementation details

When using virtual-world images we not only experiment with the originals but also with their GAN-based virtual-to-real translated counterparts, i.e., aiming at closing the domain shift between virtual and real worlds. Since the translated images are the same for both co-training modalities, we take them from [110], where a CycleGAN [144] was used to learn the translations $\mathcal{G}_{\mathcal{K}} : \mathcal{V} \rightarrow \mathcal{K}$ and $\mathcal{G}_{\mathcal{W}} : \mathcal{V} \rightarrow \mathcal{W}$. To obtain these images, CycleGAN training was done for 40 epochs using a weight of 1.0 for the identity mapping loss, and a patch-wise strategy with patches of 300 × 300 pixels, while keeping the rest of the parameters as recommended in [144]. We denote as $\mathcal{V}_{\mathcal{G}_{\mathcal{K}}} = \mathcal{G}_{\mathcal{K}}(\mathcal{V})$ and $\mathcal{V}_{\mathcal{G}_{\mathcal{W}}} = \mathcal{G}_{\mathcal{W}}(\mathcal{V})$ the sets of virtual-world images transformed by $\mathcal{G}_{\mathcal{K}}$ and $\mathcal{G}_{\mathcal{W}}$, respectively. The 2D BBs in $\mathcal{V}$ are used for $\mathcal{V}_{\mathcal{G}_{\mathcal{K}}}$ and $\mathcal{V}_{\mathcal{G}_{\mathcal{W}}}$. Furthermore, note that analogously to $\mathcal{V}$, $\mathcal{V}_{\mathcal{G}_{\mathcal{K}}}$ and $\mathcal{V}_{\mathcal{G}_{\mathcal{W}}}$ are only used for training. For multi-modal co-training, depth estimation is applied indistinctly to the real-world datasets, the virtual-world one, and the GAN-based translated ones.

In the multi-modal setting, one of the co-training views is the appearance (RGB) and the other is the corresponding estimated depth (D). To keep co-training single-sensor, we use monocular depth estimation (MDE). In particular, we leverage a state-of-the-art MDE model publicly released by Yin et al. [130]. It has been trained on KITTI data, thus, being ideal to work with $\mathcal{K}$. However, since our aim is not to obtain accurate depth estimation, but to generate an alternative data view useful to detect the objects of interest, we have used the same MDE model for all the considered datasets. Despite this, Figure 2.3 shows how the estimated depth properly captures the depth structure for the images of all datasets, i.e., not only for $\mathcal{K}$, but also for $\mathcal{W}, \mathcal{V}, \mathcal{V}_{\mathcal{G}_{\mathcal{K}}}$ and $\mathcal{V}_{\mathcal{G}_{\mathcal{W}}}$. However, we observe that the depth structure for $\mathcal{V}_{\mathcal{G}_{\mathcal{K}}}$'s and $\mathcal{V}_{\mathcal{G}_{\mathcal{W}}}$'s images is more blurred at far distances than for $\mathcal{V}$, especially for $\mathcal{V}_{\mathcal{G}_{\mathcal{W}}}$.

Following [110], we use Faster R-CNN with a VGG16 feature extractor (backbone) as the CNN architecture for object detection, i.e., as Φ in Algorithm 1. In particular, we rely on the Detectron implementation [30]. For training, we always initialize VGG16 with ImageNet pre-trained weights, while the weights of the rest of the CNN (i.e., the candidates' generator and classifier stages) are randomly initialized. Faster R-CNN training is based on 40,000 iterations of the SGD optimizer. Note that these iterations refer to the function $\text{Train}(\Phi, \mathcal{H}_\Phi, \mathcal{S}^l, \mathcal{S}^{\hat{l}}) : \phi$ in Algorithm 1, not to co-training cycles. Each iteration uses a mini-batch of two images randomly sampled from $\mathcal{S}^l \cup \mathcal{S}^{\hat{l}}$. Thus, looking at how $\text{Train}(\Phi, \mathcal{H}_\Phi, \mathcal{S}^l, \mathcal{S}^{\hat{l}}) : \phi$ is called in Algorithm 1, we can see that, for each view, the parameter $\mathcal{S}^l$ receives the same input in all co-training cycles, while $\mathcal{S}^{\hat{l}}$ changes from cycle-to-cycle. The SGD learning rate starts at 0.001 and we set a decay of 0.1 at iterations 30,000 and 35,000. In the case of multi-modal co-training, we use horizontal mirroring as a data augmentation

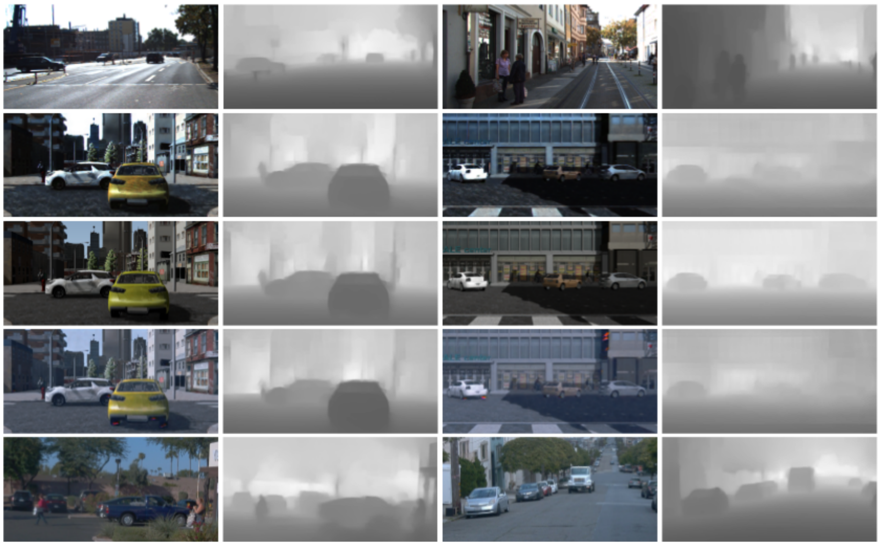


Figure 2.3: RGB images with their estimated depth. From top to bottom rows: samples from $\mathscr{K}$, $\mathscr{V}_{\mathscr{G}_{\mathscr{K}}}$, $\mathscr{V}$, $\mathscr{V}_{\mathscr{G}_{\mathscr{W}}}$, $\mathscr{W}$. The samples of $\mathscr{V}_{\mathscr{G}_{\mathscr{K}}}$ and $\mathscr{V}_{\mathscr{G}_{\mathscr{W}}}$ correspond to transforming the samples of $\mathscr{V}$ to $\mathscr{K}$ and $\mathscr{W}$ domains, respectively. The monocular depth estimation model [130] was trained on the $\mathscr{K}$ domain.

technique. However, we cannot do it in the case of single-modal co-training because both data views would highly correlate. Note that, as it was done in [110] and we can see in Table 2.1, horizontal mirroring is the technique used to generate one of the data views in single-modal co-training. In terms of Algorithm 1, all these settings are part of $\mathscr{H}_\Phi$ and they are the same to train both ϕ_1 and ϕ_2. The values set for the co-training hyper-parameters are shown in Table 2.3.

Finally, note that the final detection model used for evaluations, ϕ_F, could be based on any CNN architecture for object detection, provided the GT it expects consists of 2D BBs. However, for the sake of simplicity, we also rely on Faster R-CNN to obtain ϕ_F.

2.4.3 Results

To include multi-modality we improved and adapted the code used in [110]. For this reason, we not only execute the multi-modal co-training experiments but also redo the single-modal and baseline ones. The conclusions in [110] remain, but by

Table 2.3: Co-training hyper-parameters as defined in Algorithm 1. We use the same values for $\mathcal{K}$ and $\mathcal{W}$ datasets, but $\mathcal{H}_{seq}$ only applies to $\mathcal{W}$. N, n, m, Δt_1, and Δt_2 are set in number-of-images units, K_{min}, K_{max} and ΔK in number-of-cycles, $T_{\Delta_{mAP}}$ runs in [0..100]. T hyper-parameter contains the confidence detection thresholds for vehicles and pedestrians, which run in [0..1], and we have set the same value for both. The setting $m = \infty$ means that all the images pseudo-labeled at current co-training cycle are exchanged by the models ϕ_1 and ϕ_2 for collaboration, i.e., these will then select the n less confident for them.

					$\mathcal{H}_{stp}$			$\mathcal{H}_{seq}$	
T	N	n	m	K_{min}	K_{max}	ΔK	$T_{\Delta_{mAP}}$	Δt_1	Δt_2
{0.8,0.8}	500	100	∞	20	30	5	2.0	5	10

repeating these experiments, all the results presented in this chapter are based on the same code.

Standard SSL setting

We start the evaluation of co-training in a standard SSL setting, i.e., working only with either the $\mathcal{K}$ or $\mathcal{W}$ dataset to avoid domain shift. In this setting, the cardinality of the unlabeled dataset is supposed to be significantly higher than the cardinality of the labeled, we divide the corresponding training sets accordingly. In particular, for $\mathcal{X}^{tr} \in \{\mathcal{K}^{tr}, \mathcal{W}^{tr}\}$, we use the p% of $\mathcal{X}^{tr}$ as the labeled training set ($\mathcal{X}^{l}$) and the rest as the unlabeled training set ($\mathcal{X}^{u}$). We explore $p = 5$ and $p = 10$, where the corresponding $\mathcal{X}^{tr}$ is sampled randomly once and frozen for all the experiments. Table 2.4 shows the obtained results for both co-training modalities. We also report upper-bound (UB) and lower-bound (LB) results. The UB corresponds to the case $p = 100$, i.e., all the BBs are human-labeled. The LBs correspond to the $p = 5$ and $p = 10$ cases without using co-training, thus, not leveraging the unlabeled data. Although in this chapter we assume that ϕ_F will be based on RGB data alone, since we use depth estimation for multi-modal co-training, as a reference we also report the UB and LB results obtained by using the estimated depth alone to train the corresponding ϕ_F.

Analyzing Table 2.4, we confirm that the UB and LBs based only on the estimated depth (D) show a reasonable accuracy, although not at the level of appearance (RGB) alone. This is required for the co-training to have the chance to perform well. Aside from this, we see how, indeed, both co-training modalities clearly outperform LBs. In the $p = 5$ case, multi-modal co-training clearly outperforms single-modal in all classes (V and P) and datasets ($\mathcal{K}$ and $\mathcal{W}$). Moreover, the accuracy improvement over the LBs is significantly larger than the remaining distance to the UBs. In the $p = 10$ case, both co-training modalities perform similarly. On the other hand, for $\mathcal{K}$, the accuracy of multi-modal co-training with $p = 5$ is just ~2 points below the single-modal with $p = 10$, and less than 1 point for $\mathcal{W}$. Therefore, for 2D object detection, we recommend multi-modal co-training for a standard SSL setting with a low ratio of labeled vs. unlabeled images.

SSL under domain shift

Table 2.5 shows the LB results for a ϕ_F fully trained on virtual-world images (source domain); the results of training only on the real-world images (target domain), where these images are 100% human-labeled (i.e., 100% Labeled RGB in Table 2.4); and the combination of both, which turns out to be the UB. In the case of testing on $\mathcal{W}^{tt}$ and having $\mathcal{V}$ involved in the training, we need to accommodate the different labeling style (mainly the margin between BBs and objects) of $\mathcal{W}^{tt}$ and $\mathcal{V}$. This is only needed for a fair quantitative evaluation, thus, for performing such evaluation the detected BBs are resized by per-class constant factors. However, the qualitative results presented in the rest of the chapter are shown directly as they come by applying the corresponding ϕ_F, i.e., without applying any resizing. On the other hand, this resizing is not needed for $\mathcal{K}^{tt}$ since its labeling style is similar enough to $\mathcal{V}$.

According to Table 2.5, both co-training modalities significantly outperform the LB. Again, multi-modal co-training outperforms single-modal, especially on vehicles. Comparing multi-modal co-training with the LB, we see improvements of ~15 points for vehicles in $\mathcal{K}$, and ~25 in $\mathcal{W}$. Considering the joint improvement for vehicles and pedestrians we see ~8 points for $\mathcal{K}$, and ~15 for $\mathcal{W}$, while the distances to the UB are of ~5 points for $\mathcal{K}$, and ~2 for $\mathcal{W}$. Therefore, for 2D object detection, we recommend multi-modal co-training for an SSL scenario where the labeled data comes from a virtual world, i.e., when no human labeling is required at all, but there is a virtual-to-real domain shift.

SSL after GAN-based virtual-to-real image translation

Table 2.6 is analogous to Table 2.5, just changing the original virtual-world images ($\mathcal{V}$) by their GAN-based virtual-to-real translated counterparts ($\mathcal{V}_{\mathcal{G}_\mathcal{K}}/\mathcal{V}_{\mathcal{G}_\mathcal{W}}$). In the case of testing on $\mathcal{W}^{tt}$ and having $\mathcal{V}_{\mathcal{G}_\mathcal{W}}$ involved in the training, we apply the BB resizing mentioned in Section 2.4.3 for the quantitative evaluation. Focusing on the V&P results, we see that both the UB and LB of Table 2.6 show higher accuracy than in Table 2.5, which is due to the reduction of the virtual-to-real domain shift achieved thanks to the use of $\mathcal{V}_{\mathcal{G}_\mathcal{K}}/\mathcal{V}_{\mathcal{G}_\mathcal{W}}$. Still, co-training enables to improve the accuracy of the LBs, almost reaching the accuracy of the UBs. For instance, in the

Table 2.4: SSL (co-training) results on vehicle (V) and pedestrian (P) detection, reporting mAP. From a training set $\mathcal{X}^{tr} \in \{\mathcal{K}^{tr}, \mathcal{W}^{tr}\}$, we preserve the labeling information for a randomly chosen p% of its images, while it is ignored for the rest. We report results for p = 100 (all labels are used), p = 5 and p = 10. If $\mathcal{X}^{tt} = \mathcal{K}^{tt}$, then $\mathcal{X}^{tr} = \mathcal{K}^{tr}$; analogously, when $\mathcal{X}^{tt} = \mathcal{W}^{tt}$, then $\mathcal{X}^{tr} = \mathcal{W}^{tr}$, i.e., there is no domain shift in these experiments. Co-T (RGB) and Co-T (RGB/D) stand for single and multi modal co-training, respectively. UP and LB stand for upper bound and lower bound, respectively. Bold results indicate **best performing** within the block, where blocks are delimited by horizontal lines. $\underline{\text{Second best}}$ is underlined, but if the difference with the best is below 0.5 points, we use bold too. $\Delta\{\phi_{F_1}$ vs. $\phi_{F_2}\}$ stands for mAP of ϕ_{F_1} minus mAP of ϕ_{F_2}.

Training Set	$\mathcal{X}^{tt} = \mathcal{K}^{tt}$			$\mathcal{X}^{tt} = \mathcal{W}^{tt}$		
	V	**P**	**V&P**	**V**	**P**	**V&P**
100% Labeled (RGB)/UB	**83.43**	**67.77**	**75.60**	**61.71**	**57.74**	**59.73**
100% Labeled (D)/UB	80.80	53.43	67.12	55.14	37.67	46.41
5% Labeled (RGB)/LB	65.20	46.08	55.64	51.69	41.92	46.81
5% Labeled (D)/LB	64.45	26.70	45.58	45.21	29.98	36.70
5% Labeled + Co-T (RGB)	$\underline{74.26}$	$\underline{55.41}$	$\underline{64.84}$	$\underline{54.00}$	$\underline{56.34}$	$\underline{55.17}$
5% Labeled + Co-T (RGB/D)	**78.64**	**57.40**	**68.02**	**58.42**	**56.98**	**57.70**
10% Labeled (RGB)/LB	72.31	45.51	58.91	49.53	49.83	49.68
10% Labeled (D)/LB	69.54	46.31	57.93	47.93	33.98	40.96
10% Labeled + Co-T (RGB)	$\underline{78.63}$	**60.99**	**69.81**	$\underline{56.15}$	**60.20**	**58.18**
10% Labeled + Co-T (RGB/D)	**79.68**	**60.55**	**70.12**	**59.54**	$\underline{57.17}$	**58.36**
Δ{(5% L. + Co-T (RGB/D)) vs. (5% L. (RGB)/LB)}	+13.44	+11.32	+12.38	+6.73	+15.06	+10.89
Δ{(5% L. + Co-T (RGB/D)) vs. (100% L. (RGB)/UB)}	−4.79	−10.37	−7.58	−3.29	−0.76	−2.03
Δ{(10% L. + Co-T (RGB/D)) vs. (10% L. (RGB)/LB)}	+7.37	+15.04	+11.21	+10.01	+7.34	+8.68
Δ{(10% L. + Co-T (RGB/D)) vs. (100% L. (RGB)/UB)}	−3.75	−7.22	−5.48	−2.17	−0.57	−1.37

combined V&P detection accuracy, the single-modal co-training is 1.66 points behind the UB for $\mathcal{K}$, and 3.59 for $\mathcal{W}$. Multi-modal co-training is 2.63 points behind the UB for $\mathcal{K}$, and 4.01 for $\mathcal{W}$. Thus, in this case, single-modal co-training is performing better than multi-modal. Therefore, for 2D object detection, we can recommend even single-modal co-training for an SSL scenario where the labeled data comes from a virtual world but a properly trained GAN can perform virtual-to-real domain adaptation. On the other hand, in the case of $\mathcal{W}$, co-training from $\mathcal{V}_{\mathcal{G}_{\mathcal{W}}}$ gives rise to worse results than by using $\mathcal{V}$. We think this is due to a worse depth estimation (see Figure 2.3). In general, this suggests that whenever it is possible, training a specific monocular depth estimator for the unlabeled real-world data

Table 2.5: SSL (co-training) results on vehicle (V) and pedestrian (P) detection, under domain shift, reported as mAP. $\mathscr{X}^l$ refers to the human-labeled target-domain training set; thus, if $\mathscr{X}^{tt} = \mathscr{K}^{tt}$, then $\mathscr{X}^l = \mathscr{K}^{tr}$, and if $\mathscr{X}^{tt} = \mathscr{W}^{tt}$, then $\mathscr{X}^l = \mathscr{W}^{tr}$. $\mathscr{X}^{\hat{l}}$ consists of the same images as $\mathscr{X}^l$, but pseudo-labeled by co-training. Co-T (RGB), Co-T (RGB/D), UP, LB, $\Delta\{\phi_{F_1}$ vs. $\phi_{F_2}\}$, bold and underlined numbers are analogous to those in Table 2.4.

Training Set	$\mathscr{X}^{tt} = \mathscr{K}^{tt}$			$\mathscr{X}^{tt} = \mathscr{W}^{tt}$		
	V	**P**	**V&P**	**V**	**P**	**V&P**
Source ($\mathscr{V}$)/LB	67.46	65.18	66.32	38.88	53.37	46.13
Target ($\mathscr{X}^l$)	$\underline{83.43}$	$\underline{67.77}$	$\underline{75.60}$	**61.71**	$\underline{57.74}$	$\underline{59.73}$
Target + Source ($\mathscr{X}^l$&$\mathscr{V}$)/UB	**87.15**	**74.69**	**80.92**	$\underline{59.97}$	**62.86**	**61.42**
Co-T (RGB) + Source ($\mathscr{X}^{\hat{l}}$&$\mathscr{V}$)	77.97	**71.32**	74.65	48.56	56.33	52.45
Co-T (RGB/D) + Source ($\mathscr{X}^{\hat{l}}$&$\mathscr{V}$)	**82.90**	67.36	**75.13**	**64.40**	**59.17**	**61.79**
Δ\{(Co-T (RGB/D) + Source) vs. LB\}	+15.44	+2.18	+8.81	+25.52	+5.80	+15.66
Δ\{(Co-T (RGB/D) + ASource) vs. UB\}	−4.25	−7.33	−5.79	−0.16	-4.27	−2.21

may be beneficial for multi-modal co-training (recent advances on vision-based self-supervision for monocular depth estimation [31, 32] can be a good starting point). For this particular case, training the virtual-to-real domain adaptation GAN simultaneously to the monocular depth estimation CNN could be an interesting idea to explore in the future (we can leverage inspiration from [75, 140]).

Analyzing co-training cycles

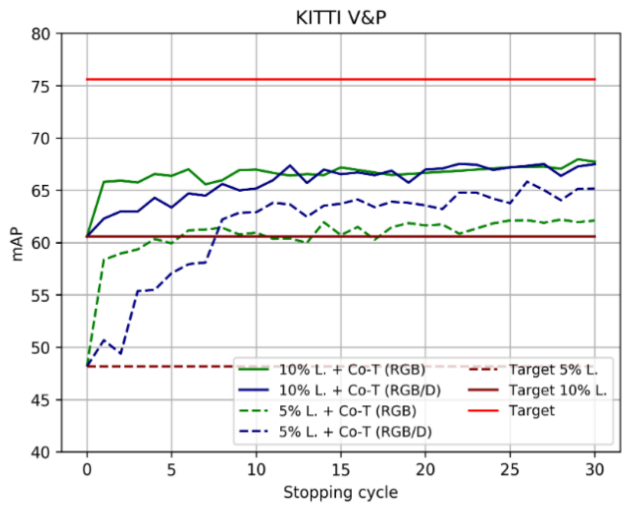
Figures 2.4 and 2.5 illustrate how co-training strategies would perform as a function of the stopping cycle, for a standard SSL setting (Figure 2.4), as well as under domain shift (Source) and when this is reduced (ASource) by using $\mathscr{V}_{\mathscr{G}_{\mathscr{K}}}/\mathscr{V}_{\mathscr{G}_{\mathscr{W}}}$ (Figure 2.5). We take the pseudo-labeled images at different co-training cycles (x-axis) as if these cycles were determined to be the stopping ones. The labeled images together with the pseudo-labeled by co-training up to the indicated cycle are used to train the corresponding ϕ_F. Then, we plot (y-axis) the accuracy (mAP) of each ϕ_F in the corresponding testing set, i.e., either $\mathscr{K}^{tt}$ or $\mathscr{W}^{tt}$. We can see how co-training strategies allow improving over the LBs from early iterations and, although slightly oscillating, keep improving until stabilization is reached. No drifting to erroneous pseudo-labeling is observed. At this point, the object samples which remain as unlabeled but are required to reach the maximum accuracy, probably are too different

Table 2.6: SSL (co-training) results on vehicle (V) and pedestrian (P) detection, after GAN-based virtual-to-real image translation, reported as mAP. ASource (adapted source) refers to $\mathcal{V}_{\mathcal{G}} \in \{\mathcal{V}_{\mathcal{G}_{\mathcal{K}}}, \mathcal{V}_{\mathcal{G}_{\mathcal{W}}}\}$. $\mathcal{X}^l$, $\mathcal{X}^{\hat{l}}$, Source, Co-T (RGB), Co-T (RGB/D), UP, LB, $\Delta\{\phi_{F_1}$ vs. $\phi_{F_2}\}$, bold and underlined numbers are analogous to those in Table 2.5.

Training Set	$\mathcal{X}^{tt} = \mathcal{K}^{tt}$			$\mathcal{X}^{tt} = \mathcal{W}^{tt}$		
	V	**P**	**V&P**	**V**	**P**	**V&P**
ASource ($\mathcal{V}_{\mathcal{G}}$)/LB	78.41	65.39	71.90	52.60	56.36	54.48
Target ($\mathcal{X}^l$)	<u>83.43</u>	<u>67.77</u>	<u>75.60</u>	<u>61.71</u>	<u>57.74</u>	<u>59.73</u>
Target + ASource ($\mathcal{X}^l$&$\mathcal{V}_{\mathcal{G}}$)/UB	**86.82**	**71.59**	**79.21**	**64.56**	**63.44**	**64.00**
Co-T (RGB) + ASource ($\mathcal{X}^{\hat{l}}$&$\mathcal{V}_{\mathcal{G}}$)	**85.17**	**69.93**	**77.55**	**61.49**	**59.33**	**60.41**
Co-T (RGB/D) + ASource ($\mathcal{X}^{\hat{l}}$&$\mathcal{V}_{\mathcal{G}}$)	83.68	**69.48**	76.58	**61.49**	58.49	**59.99**
Δ\{(Co-T (RGB) + ASource) vs. LB\}	+6.76	+4.54	+5.65	+8.89	+2.97	+5.93
Δ\{(Co-T (RGB/D) + ASource) vs. LB\}	+5.27	+4.09	+4.68	+8.89	+2.13	+5.51
Δ\{(Co-T (RGB) + ASource) vs. UB\}	−1.65	−1.66	−1.66	−3.07	−4.11	−3.59
Δ\{(Co-T (RGB/D) + ASource) vs. UB\}	−3.14	−2.11	−2.63	−3.07	-4.95	−4.01

in some aspect from the labeled and pseudo-labeled ones (e.g., they may be under a too-heavy occlusion) and would never be pseudo-labeled without additional information. Then, combining co-training with active learning (AL) cycles could be an interesting alternative, since occasional human loops could help co-training to progress more. We see also how when the starting point for co-training is at a lower accuracy, multi-modal co-training usually outperforms single-modal (e.g., in the 5% setting and under domain shift).

Figures 2.6 and 2.7 present qualitative results for ϕ_F's trained after stopping co-training at cycles 1, 10, 20 and when it stops automatically (i.e., the stopping condition of the loop in Algorithm 1 becomes true). The shown examples correspond to the most accurate setting for each dataset; i.e., for $\mathcal{K}$ (Figure 2.6) this is the co-training from $\mathcal{V}_{\mathcal{G}_{\mathcal{K}}}$ no matter the modality, while for $\mathcal{W}$ (Figure 2.7) this is the co-training from $\mathcal{V}$ in the multi-modal case and from $\mathcal{V}_{\mathcal{G}_{\mathcal{W}}}$ in the single-modal. Note that Tables 2.4–2.6, suggest to combine co-training with virtual-world data to obtain more accurate ϕ_F's.



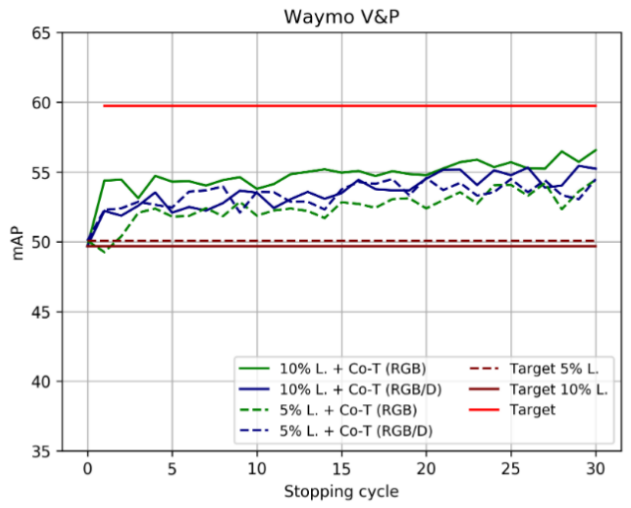


Figure 2.4: V&P detection accuracy of co-training approaches as a function of the stopping cycle. Co-T (RGB) and Co-T (RGB/D) refer to single and multi modal co-training, respectively. Target refers to the use of the 100% labeled training data, while Target p% L. indicates a lower percentage $p \in \{5, 10\}$ of labeled data available for training. Accordingly, p% L. + Co-T (view), view ∈{RGB, RGB/D}, are combinations of those. These plots complement the results shown in Table 2.4.

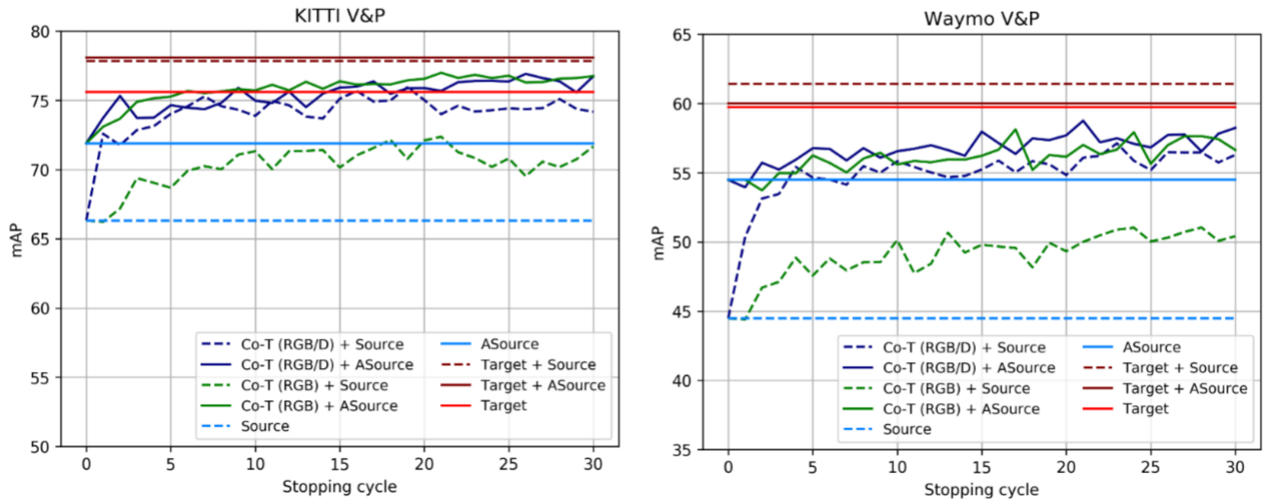


Figure 2.5: V&P detection accuracy of co-training approaches as a function of the stopping cycle. These plots are analogous to those in Figure 2.4 for the cases of using virtual-world data, i.e., both with domain shift (Source) and reducing it by the use of GANs (ASource). The Targets are the same as in Figure 2.4. These plots complement the results shown in Tables 2.5 and 2.6.

Qualitative results

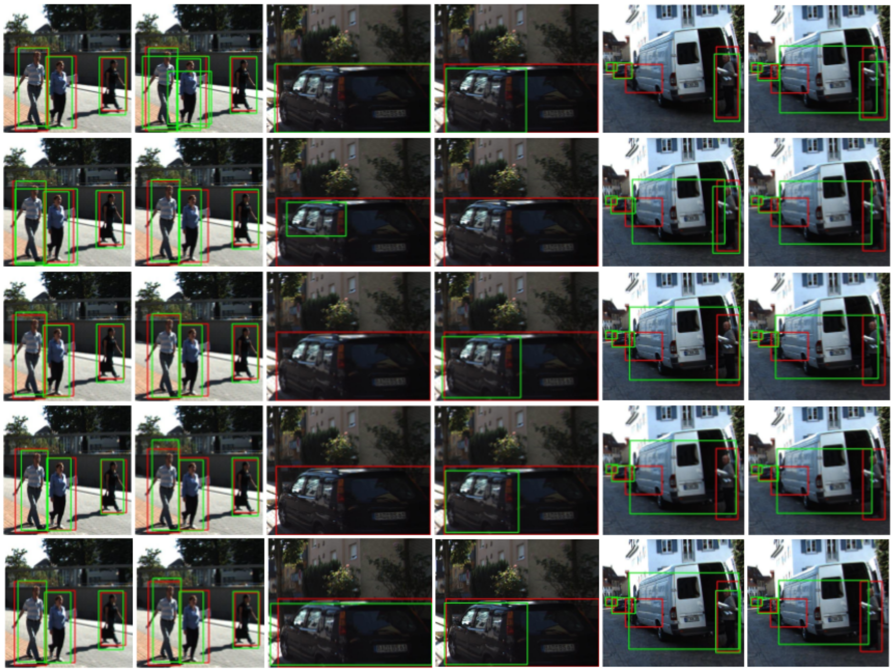


Figure 2.6: Qualitative results of how ϕ_F would perform on $\mathcal{K}^{tt}$ by stopping co-training at different cycles. We focus on co-training and object detection working from $\mathcal{V}_{\mathcal{G}_{\mathcal{K}}}$ (ASource). There are three blocks of results vertically arranged. At each block, the top-left image shows the results when using the 100% human-labeled training data plus $\mathcal{V}_{\mathcal{G}_{\mathcal{K}}}$ (Target + ASource), i.e., UB results. Detection results are shown as green BBs, and GT as red BBs. The top-right image of each block shows the results that we would obtain without leveraging the unlabeled data (ASource), i.e., LB results. The rest of the rows of the block, from top-second to bottom, correspond to stopping co-training at cycles 1, 10, 20, and automatically. In these rows, the images at the left column correspond to multi-modal co-training (i.e., Co-T (RGB/D)) and those at the right column to single-modal co-training (i.e., Co-T (RGB)).

In the left block of Figure 2.6, we show a case where both co-training modalities perform similarly on pedestrian detection, with final detections (green BBs) very close to the GT (red BBs), and clearly better than if we do not leverage the unlabeled data (top-right image of the block). We see also that the results are very similar to the case of using the 100% of human-labeled data (top-left image of the block). Moreover, even from the initial cycles of both co-training modalities the results are reasonably good, although, the best is expected when co-training finishes automatically (bottom row of the block), i.e., after the minimum number of cycles is exceeded ($K_{min} = 20$ in Table 3.1). In the mid-block, we see that only multi-modal co-training helps to properly detect a very close and partially occluded vehicle. In the right block, only multi-modal co-training helps to keep and improve the detection of a close pedestrian. Both co-training modalities help to keep an initially detected van, but multi-modal co-training induces a better BB adjustment. This is an interesting case. Since $\mathscr{V}$ only contains different types of cars but lacks a meaningful number of van samples, and $\mathscr{K}$ only has a very small percentage of those labeled, we have focused our study on the different types of cars. Therefore, vans are neither considered for training nor testing, i.e., their detection or misdetection does not affect the mAP metric either positively or negatively. However, co-training is an automatic pseudo-labeling procedure, thus it may capture or keep these samples and then force training with them. Moreover, in this setting, the hard-negatives are mined only from the virtual-world images (translated or not by a GAN) since they are fully labeled. Thus, if no sufficient vans are part of the virtual-world images, these objects cannot act as hard negatives, so that they may be detected or misdetected depending on their resemblance to the targeted objects (here types of cars). We think this is the case here. Thus, this is an interesting consideration for designing future co-training procedures supported by virtual-world data. Alternatively, by complementing co-training with occasional AL cycles, these special false positives could be reported by the human in the AL loop (provided we really want to treat them as false positives). On the other hand, in the same block of results, we see also a misdetection (isolated red BB), which does account for the quantitative evaluation. It corresponds to a rather occluded vehicle which is not detected even when relying on human labeling (top-left image of the block). Finally, note the large range of detection distances achieved for vehicles.

In the left block of Figure 2.7, we see even a larger detection range for the detected vehicles than in Figure 2.6. Faraway vehicles (small green BBs) are considered as false positives for the qualitative evaluation because these are not part of the $\mathscr{W}^{tt}$ GT (since they do not have labeled 3D BBs from which the 2D BBs are obtained). Thanks to the use of virtual-world data, these vehicles are detected (second row of the block) and both co-training modalities do not damage their detection. Note how the UBs based on virtual-world data and human-labeled real-world data are not able to detect such vehicles (first row of the block) because human labeling did not consider these faraway vehicles, while co-training does consider them as such. Besides, multi-modal co-training enables the detection of the closer vehicle

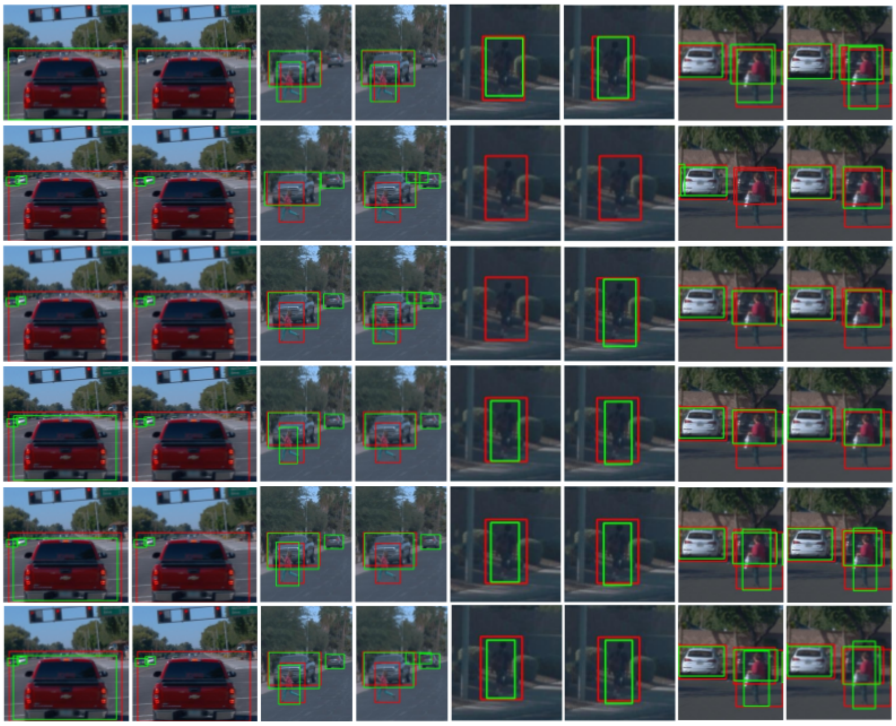


Figure 2.7: Qualitative results similar to those in Figure 2.6, but testing on $\mathcal{W}^{tt}$, co-training from $\mathcal{V}$ in the multi-modal case (left column of each block), and $\mathcal{V}_{\mathcal{G}_\mathcal{W}}$ in the single-modal case (right column of each block). Since, in these examples, the two co-training modalities are based on different (labeled) data, the first row of each block shows the respective UB results, i.e., those based on training with $\mathcal{W}^{tr}$ and either with $\mathcal{V}$ (left image: Target + Source) or $\mathcal{V}_{\mathcal{G}_\mathcal{W}}$ (right image: Target + ASource). The second row of each block shows the respective results we would obtain without leveraging the unlabeled data, i.e., the LBs based on training with $\mathcal{V}$ (left image: Source) or $\mathcal{V}_{\mathcal{G}_\mathcal{W}}$ (right image: ASource). As in Figure 2.6, the rest of the rows of each block correspond to stopping co-training at cycles 1, 10, 20, and automatically.

since cycle 10. In the next block to the right, multi-modal co-training enables to detect a close kid since cycle 10, while single-modal does not at the end. In addition, single-modal co-training also introduces a distant false positive. Similarly to the left block, in this block both co-training modalities keep an unlabeled vehicle detected thanks to the use of the virtual-world data (second row), not detected (first row) when these data are complemented with human-labeled data (since, again, this vehicle is not even labeled). What is happening in these cases, is that there is a lack of real-world human-labeled 3D BBs for distant vehicles, which is compensated by the use of virtual-world data and maintained by co-training. In the next block to the right, we see how a pedestrian is detected thanks to both co-training methods since only using virtual-world data was not possible (second row). In the right block, both co-training modalities allow for vehicle and pedestrian detections similar to the UBs (first row). Note that the vehicle partially hidden behind the pedestrian was not detected by only using virtual-world data (second row), and neither was detected the pedestrian when using $\mathscr{V}$ (second row, left) or was poorly detected when using $\mathscr{V}_{\mathscr{G}_{\mathscr{W}}}$ (second row, right).

Finally, Figure 2.8 shows additional qualitative results on $\mathscr{K}^{tt}$ and $\mathscr{W}^{tt}$ when using multi-modal co-training, in the case of $\mathscr{K}^{tt}$ based on $\mathscr{V}_{\mathscr{G}_{\mathscr{K}}}$ and $\mathscr{V}$ for $\mathscr{W}^{tr}$, i.e., we show the results of the respective best models. Overall, in the case of $\mathscr{K}^{tt}$, we see how multi-modal co-training (Co-T (RGB/D)) enables to better adjust detection BBs, and removing some false positives. In the case of $\mathscr{W}^{tt}$, multi-modal co-training enables to keep even small vehicles that are not part of the GT but are initially detected thanks to the use of virtual-world data. It also helps to detect vehicles and pedestrians not detected by only using the virtual-world data, although further improvements are needed since some pedestrians are still difficult to detect even with co-training.

Answering (Q1) and (Q2)

After presenting our multi-modal co-training and the extensive set of experiments carried out, we can answer the research questions driving this study. In particular, we base our answers in the quantitative results presented in Tables 2.4–2.6, the plots shown in Figures 2.4 and 2.5, as well as the qualitative examples shown in Figures 2.6–2.8, together with the associated comments we have drawn from them.

(Q1) Is multi-modal (RGB/D) co-training effective on the task of providing pseudo-labeled object BBs? Indeed, multi-modal co-training is effective for pseudo-labeling object BBs under different settings, namely, for standard SSL (no domain shift, a few human-labeled data) and when using virtual-world data (many virtual-world labeled data, but no human-labeled data) both under domain shift and after reducing it by GAN-based virtual-to-real image translation. The achieved improvement over the lower bound configurations is significant, allowing to be almost in pair with upper bound configurations. In the standard SSL setting, by only labeling the 5% of the training dataset, multi-modal co-training allows obtaining accuracy values relatively close to the upper bounds. When using virtual-world data, i.e., without human labeling at all, the same observations hold. Moreover, multi-modal co-training and GAN-based virtual-to-real image translation have been shown to complement each other.

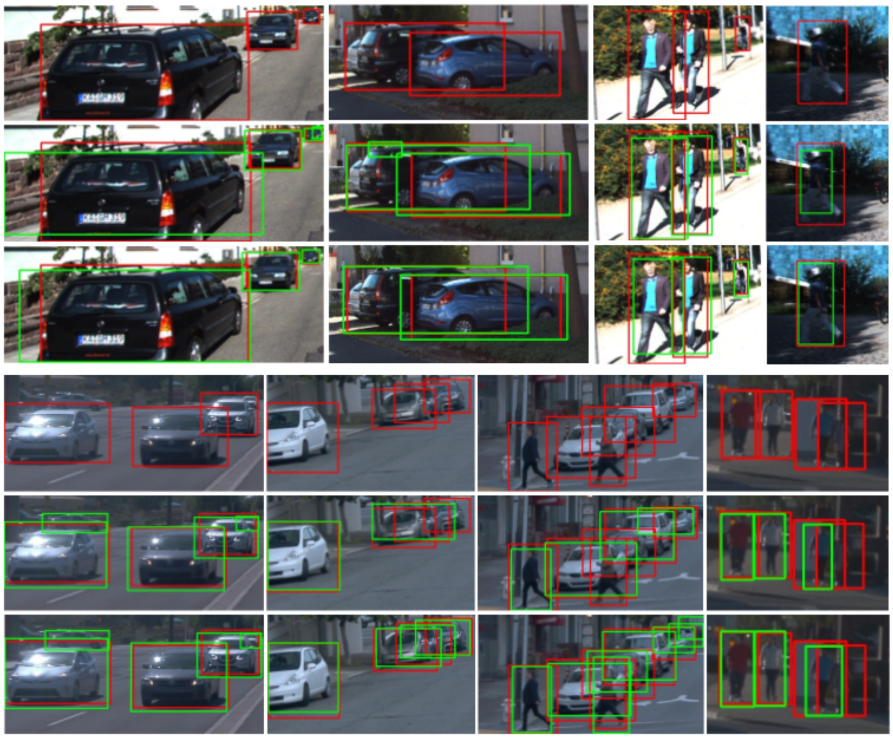


Figure 2.8: Qualitative results on $\mathcal{K}^{tt}$ (top block of rows) and $\mathcal{W}^{tt}$ (bottom block of rows). In each block, we show (top row) GT as red BBs, (mid row) detections, as green BBs, when training with $\mathcal{X}^{l}$, (bottom row) detections with $\mathcal{X}^{l} \cup \mathcal{X}^{\hat{l}}$. In this case, $\mathcal{X}^{\hat{l}}$ comes from applying C-T (RGB/D) on either $\mathcal{K}^{tr}$ or $\mathcal{W}^{tr}$, and $\mathcal{X}^{l}$ is $\mathcal{V}_{\mathcal{G}_{\mathcal{K}}}$ for $\mathcal{K}^{tr}$, while it is $\mathcal{V}$ for $\mathcal{W}^{tr}$.

(Q2) How does perform multi-modal (RGB/D) co-training compared to single-modal (RGB)? We conclude that in a standard SSL setting (no domain shift, a few human-labeled data) and under virtual-to-real domain shift (many virtual-world labeled data, no human-labeled data) multi-modal co-training outperforms single-modal. In the latter case, when GAN-based virtual-to-real image translation is performed both co-training modalities are on par; at least, by using an off-the-shelf monocular depth estimation model not specifically trained on the translated images.

To drive future research, we have performed additional experiments. These consist in correcting the pseudo-labels obtained by multi-modal co-training in three different ways, namely, removing false positives (FP), adjusting the BBs to the ones of the GT (BB) for correctly pseudo-labeled objects (true positives), and a combination of both (FP + BB). After changing the pseudo-labels in that way, we train the corresponding ϕ_F models and evaluate them. Table 2.7 presents the quantitative results. Focusing on the standard SSL setting (5%, 10%), we see that the main problem for vehicles in $\mathcal{K}$ is BB adjustment, while for pedestrians is the introduction of FPs. In the latter case, false negatives (FN; i.e., missing pseudo-labeled objects) seem to be also an issue to reach upper bound accuracy. When we have the support of virtual-world data, FNs do not seem to be a problem, and addressing BB correction for vehicles and removing FPs for pedestrians would allow reaching upper bounds. In the case of $\mathcal{W}$, we came to the same conclusions for vehicles, the main problem is BB adjustment, while in the case of pedestrians the main problem is not that clear. In other words, there is more balance between FP and BB. On the other hand, regarding these additional experiments, we trust more the conclusions derived from $\mathcal{K}$. The reason is that, as we have seen in Figures 2.7 and 2.8, co-training was correctly pseudo-labeling objects that are not part of the GT, so in this study, these are either considered FPs and so wrongly removed (FP, FP + BB settings), or would not have a GT BB to which adjust them (BB, FP + BB settings).

After this analysis, we think we can explore two main future lines of research. First, to improve BB adjustment, we could complement multi-modal co-training with instance segmentation, where using Mask R-CNN [37] would be a natural choice. Note that virtual-world data can also have instance segmentation as part of their GT suite. Second, to remove FPs, we could add an AL loop where humans could remove even several FP with a few clicks (note that this is much easier than delineating object BBs). On the other hand, additional CNN models could be explored to avoid FPs as a post-processing step to multi-modal co-training. Besides these ideas, we think that, whenever is possible, the monocular depth estimation model should be trained on the target domain data, rather than trying to use an off-the-shelf model. Since we think that not doing so was damaging the combination of multi-modal co-training and GAN-based virtual-to-real image translation, an interesting approach would be to perform both tasks simultaneously.

Table 2.7: Digging in the results throw three post-processing settings applied to co-training pseudo-labels: (FP) where we remove the false positive pseudo-labels; (BB) where we change the pseudo-labels by the corresponding GT (i.e., in terms of Figures 2.6–2.8, green BBs are replaced by red ones); (FP + BB) which combines both. This table follows the terminology of Tables 2.4–2.6. Δ_X, $X \in$ {FP,BB,FP+BB}, stands for difference of setting X minus the respective original (i.e., using the co-training pseudo-labels). Moreover, for each block of results, we add the #FP/FP% row, where #FP refers to the total number of false positives that are used to train the final object detector, ϕ_F, while FP% indicates what percentage they represent regarding the whole set (labeled and pseudo-labeled BBs) used to train ϕ_F.

Training Set	$\mathcal{X}^{tt} = \mathcal{K}^{tt}$			$\mathcal{X}^{tt} = \mathcal{W}^{tt}$		
	V	**P**	**V&P**	**V**	**P**	**V&P**
Target + ASource (UB)	86.82	71.59	79.21	64.56	63.44	64.00
5% Labeled + Co-T (RGB/D)	78.64	57.40	68.02	58.42	$\underline{56.98}$	57.70
5% Labeled + Co-T (RGB/D)/FP	79.29	**60.50**	69.90	59.28	55.89	57.59
5% Labeled + Co-T (RGB/D)/BB	**85.18**	$\underline{58.87}$	**72.03**	**63.25**	56.58	**59.92**
5% Labeled + Co-T (RGB/D)/FP + BB	**85.61**	58.75	**72.18**	$\underline{62.49}$	**57.63**	**60.06**
Δ_{FP}	+0.65	+3.10	+1.88	+0.86	−1.09	−0.11
Δ_{BB}	+6.54	+1.47	+4.01	+4.83	−0.40	+2.22
$\Delta_{\text{FP+BB}}$	+6.97	+1.35	+4.16	+4.07	+0.91	+2.36
#FP/FP%	1723/13.65%	275/16.33%		5952/10.39%	731/11.33%	
10% Labeled + Co-T (RGB/D)	79.68	60.55	70.12	59.54	**57.17**	58.36
10% Labeled + Co-T (RGB/D)/FP	79.81	**61.65**	70.73	60.24	**57.38**	58.81
10% Labeled + Co-T (RGB/D)/BB	**85.28**	59.21	**72.25**	**63.01**	56.07	$\underline{59.54}$
10% Labeled + Co-T (RGB/D)/FP + BB	$\underline{83.23}$	$\underline{61.03}$	**72.13**	**63.20**	**56.99**	**60.10**
Δ_{FP}	+0.13	+1.10	+0.61	+0.70	+0.21	+0.45
Δ_{BB}	+5.60	−1.34	+2.01	+3.47	−1.1	+1.18
$\Delta_{\text{FP+BB}}$	+3.55	+0.48	+2.01	+3.66	−0.18	+1.74
#FP/FP%	1998/14.06%	408/16.83%		4553/7.42%	547/7.30%	
Co-T (RGB/D) + Source	82.90	67.36	75.13	**64.40**	**59.17**	**61.79**
Co-T (RGB/D) + Source/FP	83.37	$\underline{70.95}$	$\underline{77.16}$	57.68	56.04	56.86
Co-T (RGB/D) + Source/BB	**88.94**	61.69	75.32	62.14	$\underline{57.22}$	$\underline{59.68}$
Co-T (RGB/D) + Source/FP + BB	**89.07**	**71.88**	**80.48**	$\underline{62.56}$	56.59	59.58
Δ_{FP}	+0.47	+3.59	+2.03	−6.72	−3.13	−4.93
Δ_{BB}	+6.04	−5.67	+0.19	−2.26	−1.95	−2.11
$\Delta_{\text{FP+BB}}$	+6.17	+4.52	+5.35	−1.84	−2.58	−2.21
#FP/FP%	3281/3.57%	883/0.87%		18293/21.06%	970/2.02%	
Co-T (RGB/D) + ASource	83.68	69.48	76.58	61.49	**59.33**	**60.41**
Co-T (RGB/D) + ASource/FP	83.07	$\underline{70.40}$	76.74	60.67	$\underline{57.72}$	59.20
Co-T (RGB/D) + ASource/BB	**89.27**	68.63	$\underline{78.95}$	$\underline{62.06}$	57.64	$\underline{59.85}$
Co-T (RGB/D) + ASource/FP + BB	**88.93**	**71.45**	**80.19**	**64.67**	55.27	**59.97**
Δ_{FP}	−0.61	+0.92	0.16	−0.82	−1.61	−1.21
Δ_{BB}	+5.59	−0.85	+3.61	+0.57	−1.69	−0.56
$\Delta_{\text{FP+BB}}$	+5.25	+1.97	+3.61	+3.18	−4.06	−0.44
#FP/FP%	3097/5.59%	479/1.03%		20949/23.15%	816/1.70%	

2.5 Conclusions

In this chapter, we have addressed the curse of data labeling for onboard deep object detection. In particular, following the SSL paradigm, we have proposed multi-modal co-training for object detection. This co-training relies on a data view based on appearance (RGB) and another based on estimated depth (D), the latter obtained by applying monocular depth estimation, so keeping co-training as a single-sensor method. We have performed an exhaustive set of experiments covering the standard SSL setting (no domain shift, a few human-labeled data) as well as the settings based on virtual-world data (many virtual-world labeled data, no human-labeled data) both with domain shift and without (using GAN-based virtual-to-real image translation). In these settings, we have compared multi-modal co-training and appearance-based single-modal co-training. We have shown that multi-modal co-training is effective in all settings.

In the standard SSL setting, from a 5% of human-labeled training data, co-training can already lead to a final object detection accuracy relatively close to upper bounds (i.e., with the 100% of human labeling). The same observation holds when using virtual-world data, i.e., without human labeling at all. Multi-modal co-training outperforms single-modal in standard SSL and under domain shift, while both co-training modalities are on par when GAN-based virtual-to-real image translation is performed; at least, by using an off-the-shelf depth estimation model not specifically trained on the translated images. Moreover, multi-modal co-training and GAN-based virtual-to-real image translation have been proved to be complementary.

Moreover, our results suggest that in the standard SSL setting and under virtual-to-real domain shift, multi-modal co-training outperforms single-modal. In the latter case, when GAN-based virtual-to-real image translation is performed both co-training modalities are on par; at least, by using an off-the-shelf depth estimation model not specifically trained on the translated images. Overall, multi-modal co-training significantly improves over lower bounds, actually not being too far from upper bounds when relying on virtual-world data too. GAN-based virtual-to-real image translation and co-training complement each other. For the future, we plan several lines of work, namely, improving the adjustment of object BBs by using instance segmentation upon detection, removing false-positive pseudo-labels by using a post-processing AL cycle, and coupling the training of monocular depth estimation and GAN-based virtual-to-real image translation to train on the target domain. Besides, we would like to extend co-training experiments to other classes of interest for onboard perception (traffic signs, motorbikes, bikes, etc.), as well as adapting the method to tackle other tasks such as pixel-wise semantic segmentation.

For the future, we plan several lines of work, namely, improving the adjustment of object BBs by using instance segmentation upon detection and removing false-positive pseudo-labels by using a post-processing AL cycle. Moreover, we believe that the monocular depth estimation model should be trained based on target domain data whenever possible. When GAN-based image translation is required, we could jointly train the monocular depth estimation model and the GAN on the target domain. Besides, we would like to extend co-training experiments to other classes of interest for onboard perception (traffic signs, motorbikes, bikes, etc.), as well as adapting the method to tackle other tasks such as pixel-wise semantic segmentation.

3 Co-training for unsupervised domain adaptation of semantic segmentation models

Semantic image segmentation is a central and challenging task in autonomous driving, addressed by training deep models. Since this training draws to a curse of human-based image labeling, using synthetic images with automatically generated labels together with unlabeled real-world images is a promising alternative. This implies to address an unsupervised domain adaptation (UDA) problem. In this chapter, we propose a new co-training procedure for synth-to-real UDA of semantic segmentation models. It consists of a self-training stage, which provides two domain-adapted models, and a model collaboration loop for the mutual improvement of these two models. These models are then used to provide the final semantic segmentation labels (pseudo-labels) for the real-world images. The overall procedure treats the deep models as black boxes and drives their collaboration at the level of pseudo-labeled target images, i.e., neither modifying loss functions is required, nor explicit feature alignment. We test our proposal on standard synthetic and real-world datasets for onboard semantic segmentation. Our procedure shows improvements ranging from ~13 to ~31 mIoU points over baselines.

3.1 Introduction

Semantic image segmentation is a central and challenging task in autonomous driving, as it involves predicting a class label (e.g., road, pedestrian, vehicle, etc) per pixel in outdoor images. Therefore, non surprisingly, the development of deep models for semantic segmentation has received a great deal of interest since deep learning is the core for solving computer vision tasks [3, 11, 12, 14, 65, 111, 125]. In this chapter, we do not aim at proposing a new deep model architecture for onboard semantic segmentation, but our focus is on the training process of semantic segmentation models. More specifically, we explore the setting where such models must perform in real-world images, while for training them we have access to automatically generated synthetic images with semantic labels together with unlabeled

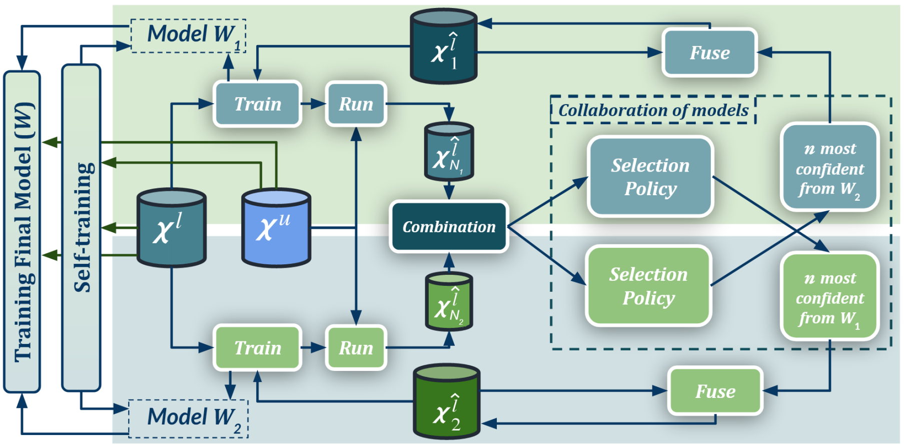


Figure 3.1: Co-training procedure for UDA. $\mathcal{X}^l$ is a set of labeled synthetic images, $\mathcal{X}^u$ a set of unlabeled real-world images, and $\mathcal{X}^{\hat{l}}_x$ is the set x of real-world pseudo-labeled images (automatically generated). Our self-training stage provides two initial domain-adapted models ($\mathcal{W}_1, \mathcal{W}_2$), which are further trained collaboratively by exchanging pseudo-labeled images. Thus, this procedure treats the deep models as black boxes and drives their collaboration at the level of pseudo-labeled target images, i.e., neither modifying loss functions is required, nor explicit feature alignment. See details in Sect. 3.3 and Algorithms 2–4.

real-world images. It is well-known that training deep models on synthetic images for performing on real-world ones requires domain adaptation [16, 112], which must be unsupervised if we have no labels from real-world images [116]. Thus, this chapter falls in the realm of unsupervised domain adaptation (UDA) for semantic segmentation [9,27,36,57,67,77,103,114,127,137,146,147], i.e., in contrast to assuming access to labels from the target domain [13,113]. Note that the great relevance of UDA in this context comes from the fact that, until now, pixel-level semantic image segmentation labels are obtained by a cumbersome and error-prone manual work. In fact, this is the reason why the use of synthetic datasets [85,86,119] arouses great interest.

In this chapter, we address synth-to-real UDA following a co-training pattern [5], which is a type of semi-supervised learning (SSL) [104, 107] approach. Essentially, canonical co-training consists in training two models in a collaborative manner when only few labeled data are available but we can access to a relatively large amount of unlabeled data. In the canonical co-training paradigm, domain shift between labeled and unlabeled data are not present. However, UDA can be instantiated in this paradigm.

In previous works, we successfully applied a co-training pattern under the synth-to-real UDA setting for deep object detection [34, 110]. This encourages us to address the challenging problem of semantic segmentation under the same setting by proposing a new co-training procedure, which is summarized in Figure 3.1. It consists of a self-training stage, which provides two domain-adapted models, and a model collaboration loop for the mutual improvement of these two models. These models are then used to provide the final semantic segmentation labels (pseudo-labels) for the real-world images. In contrast to previous related works, the overall procedure treats the deep models as black boxes and drives their collaboration only at the level of pseudo-labeled target images, i.e., neither modifying loss functions is required, nor explicit feature alignment. We test our proposal on synthetic (GTAV [85], Synscapes [119], SYNTHIA [86]) and real-world datasets (Cityscapes [15], BDD100K [131], Mapillary Vistas [70]) which have become standard for researching on onboard semantic segmentation. Our procedure shows improvements ranging from ~13 to ~31 mean intersection-over-union (mIoU) points over baselines, being less than 10 mIoU points below upper-bounds. Moreover, up to the best of our knowledge, we are the first reporting synth-to-real UDA results for semantic segmentation in BDD100K and Mapillary Vistas.

Section 3.2 contextualizes our work. Section 3.3 details the proposed procedure. Section 3.4 describes the experimental setup and discusses the obtained results. Finally, Section 3.6 summarizes this work.

3.2 Related works

Li et al. [60] and Wang et al. [114] rely on adversarial alignment to perform UDA. While training a deep model for semantic segmentation, it is performed adversarial image-to-image translation (synth-to-real) together with an adversarial alignment of the model features arising from the *source* (synthetic images) and *target* domains (real-world images). Both steps are alternated as part of an iterative training process. For feature alignment, pseudo-labeling of the target domain images is performed. This involves to apply an automatically computed per-class max probability threshold (MPT) to class predictions. Tranheden et al. [103] follow the idea of mixing source and target information (synthetic and real) as training samples [86]. However, target images are used after applying ClassMix [72], i.e., a class-based collage between source and target images. This requires the semantic segmentation ground truth, which for the synthetic images (source) is available while for the real-world ones (target) pseudo-labels are used. Such domain adaptation via cross-domain mixed sampling (DACS) is iterated so that the semantic segmentation model can improve its accuracy by eventually producing better pseudo-labels. Gao et al. [27] not only augment target images with source classes, but the way around too. Their dual soft-paste (DSP) is used withing a teacher-student framework, where the teacher model generates the pseudo-labels. Zou et al. [146] propose a self-training procedure where per-cycle pseudo-labels are considered by following a self-paced curriculum learning policy. An important step is class-balanced self-training (CBST), which is similar to MPT since a per-class confidence-based selection of pseudo-labels is performed. Spatial priors (SP) based on the source domain (synth) are also used. The authors improved their proposal in [147] by incorporating confidence regularization steps for avoiding error drift in the pseudo-labels.

Chao et al. [9] assume the existence of a set of semantic segmentation models independently pre-trained according to some UDA technique. Then, the pseudo-label confidences coming from such models are unified, fused, and finally distilled into a student model. Zhang et al. [137] propose a multiple fusion adaptation (MFA) procedure, which integrates online-offline masked pseudo-label fusion, single-model temporal fusion, and cross-model fusion. To obtain the offline pseudo-labels, existing UDA methods must be applied. In particular, so-called FDA [127] method is used to train two different models which produce two maps of offline pseudo-labels for each target image. Other two models, $m_1\&m_2$, are then iteratively trained. Corresponding temporal moving average models, $\hat{m}_1\&\hat{m}_2$, are kept and used to generate the online pseudo-labels. The training total loss seeks for consistence between class predictions of each m_i and both offline pseudo-labels and class predictions from the corresponding $\hat{m}_i$. Moreover, consistence between the online pseudo-labels from $\hat{m}_i$ and the predictions from m_j, $i \neq j$, is used as a collaboration mechanism between models. Offline and online pseudo-labels are separately masked out by corresponding CBST-inspired procedures. He et al. [36] assume the existence of different source domains. To reduce the visual gap between each source domain and the target domain there is a first step where their LAB color spaces are aligned. Then, there are as many semantic segmentation models to train as source domains. Model training relies on source labels and target pseudo-labels. The latter are obtained by applying the model to the target domain images and using a CBST-inspired procedure for thresholding the resulting class confidences. The training of each model is done iteratively so that the relevance of pseudo-labels follows a self-paced curriculum learning. Collaboration between models is also part of the training. In particular, it is encouraged the agreement of the confidences of the different models when applied to the same source domain, for all source domains. Qin et al. [77] proposed a procedure consisting of feature alignment based on cycleGAN [144], additional domain alignment via two models whose confidence discrepancies are considered, and a final stage where the confidences of these models are combined to obtain pseudo-labels which are later used to fine-tune the models. Luo et al. [67] focused on the lack of semantic consistency (some classes may not be well aligned between source and target domains, while others can be). Rather than a global adversarial alignment between domains, a per-class adversarial alignment is proposed. Using a common feature extractor, but two classification heads, per-class confidence discrepancies between the heads are used to evaluate class alignment. The classification heads are forced to be different by a cosine distance loss. Combining the confidences of the two classifiers yields the final semantic segmentation prediction. This approach does not benefit from pseudo-labels.

In contrast to these methods, our proposal is purely data-driven in the sense of neither requiring changing the loss function of the selected semantic segmentation model, nor explicit model features alignment of source and target domains via loss function, i.e., we treat the semantic segmentation model as a black box. Our UDA is inspired in co-training [5], so we share with some of the reviewed works the benefit of leveraging pseudo-labels. In our proposal two models collaborate at pseudo-label level for compensating labeling errors. These two models arise from our previous self-training stage, which share with previous literature self-paced adaptive thresholding inspired by MPT and CBST, as well as pixel-level domain mixes inspired by ClassMix. Our proposal is complementary to image pre-processing techniques such as color space adjustments and learnable image-to-image transformations. In case of having multiple synthetic domains, we assume they are treated as a single (heterogeneous) source domain, which has been effective in other visual tasks [79].

3.3 Method

In this section, we explain our data-driven co-training procedure, i.e., the self-training stage, and the model collaboration loop for the mutual improvement of these two models, which we call *co-training loop*. Overall, our proposal works at pseudo-labeling level, i.e., it does not change the loss function of the semantic segmentation model under training. Global transformations (e.g., color corrections, learnable image-to-image transformations) on either source or target domain images are seen as pre-processing steps. Moreover, in case of having access to multiple synthetic datasets, whether to use them one at a time or simultaneously is just a matter of the input parameters passed to our co-training procedure.

```
Input  : Set of labeled images: $\mathscr{X}^l$
         Set of unlabeled images: $\mathscr{X}^u$
         Net. init. weights & training hyp.-p.: $\mathscr{W}, \mathscr{H}_\Phi$
         Self-t. hyp.-p.: $\mathscr{H}_{st} = \{\mathscr{T}, N, n, K_m, K_M, \mathscr{M}_{df}\}$
Output : Two refined models: $\mathscr{W}_{K_m}, \mathscr{W}_{K_M}$
                        // Initialization
              $\mathscr{W}_0$   $\leftarrow$ BasicModelTraining($\mathscr{W}, \mathscr{H}_\Phi, \mathscr{X}^l$)
 $< \mathscr{X}^{\hat{l}}, k, \mathscr{V}_{C_T}, \mathscr{W} >$   $\leftarrow < \emptyset, 0, \mathbf{0}, \mathscr{W}_0 >$
repeat
                        // Self-training Loop
    $< \mathscr{X}^{\hat{l}}_N, \mathscr{V}_{C_T} >$   $\leftarrow$ Run($\mathscr{W}$, Rnd($\mathscr{X}^u, N$), $k, \mathscr{T}$)
              $\mathscr{X}^{\hat{l}}$   $\leftarrow$ Fuse($\mathscr{X}^{\hat{l}}$, Select($n, \mathscr{X}^{\hat{l}}_N$))
              $\mathscr{W}$   $\leftarrow$ Train($\mathscr{W}_0, \mathscr{H}_\mathscr{W}, \mathscr{X}^l, \mathscr{X}^{\hat{l}}, \mathscr{M}_{df}, \mathscr{V}_{C_T}$)
    if ($k == K_m$) then $\mathscr{W}_{K_m} \leftarrow \mathscr{W}$
    elseif ($k == K_M$) then $\mathscr{W}_{K_M} \leftarrow \mathscr{W}$ endif
until ($k == K_M$; $k$++)
return $\mathscr{W}_{K_m}, \mathscr{W}_{K_M}$
```

Algorithm 2: Self-training Stage

3.3.1 Self-training stage

Algorithm 2 summarizes the self-training stage, which we detail in the following.

Input & output parameters. The input $\mathscr{X}^l$ refers to the set of fully labeled source images; while $\mathscr{X}^u$ refers to the set of unlabeled target images. In our UDA setting, the source images are synthetic and have automatically generated per-pixel semantic segmentation ground truth (labels), while the target images are acquired with real-world cameras. $\mathscr{W}$ refers to the weights of the semantic segmentation model (a CNN) already initialized (randomly or by pre-training on a previous task); while $\mathscr{H}_\Phi$ are the usual hyper-parameters required for training the model in a supervised manner (e.g., optimization policy, number of iterations, etc). $\mathscr{H}_{st} = \{\mathscr{T}, N, n, K_m, K_M, \mathscr{M}_{df}\}$ consists of parameters specifically required by the proposed self-training. K_M is the number of self-training cycles, where we output the model, $\mathscr{W}_{K_M}$, at the final cycle. K_m, $K_m < K_M$, indicates an intermediate cycle from where we also output the corresponding model, $\mathscr{W}_{K_m}$. N is the number of target images used to generate pseudo-labels at each cycle, while n, $n < N$, is the number of pseudo-labeled images to be kept for next model re-training. $\mathscr{H}_{st}$ also contains $\mathscr{T} = \{p_m, p_M, \Delta p, C_m, C_M\}$, a set of parameters to implement a self-paced curriculum learning policy for obtaining pseudo-labels from model confidences, which is inspired in MPT [57] and CBST [146]. Finally, $\mathcal{M}_{df} = \{p_{MB}, p_{CM}\}$ consists of parameters to control how source and target images are combined.

Initialization. We start by training a model, $\mathcal{W}_0$, on the (labeled) source images, $\mathcal{X}^l$, according to $\mathcal{W}$ and $\mathcal{H}_\Phi$. At each self-training cycle, $\mathcal{W}_0$ is used as pre-trained model.

Self-training cycles (loop). Each cycle starts by obtaining a set of pseudo-labeled images, $\mathcal{X}_N^{\hat{l}}$. For the shake of speed, we do not consider all the images in $\mathcal{X}^u$ as candidates to obtain pseudo-labels. Instead, N images are *selected* from $\mathcal{X}^u$ and, then, the current model $\mathcal{W}$ is applied to them (*run*). Thus, we obtain N semantic maps. Each map can be seen as a set of confidence channels, one per class. Thus, for each class, we have N confidence maps. Let's term as $\mathcal{V}_c$ the vector of confidence values > 0 gathered from the N confidence maps of class c. For each class c, a confidence threshold, C_{T_c}, is set as the value required for having $p\%$ values of vector $\mathcal{V}_c$ over it, where $p = \min(p_m + k\Delta p, p_M)$. Let's term as $\mathcal{V}_{C_T}$ the vector of confidence thresholds from all classes. Now, $\mathcal{V}_{C_T}$ is used to perform per-class thresholding on the N semantic segmentation maps, so obtaining the N pseudo-labeled images forming $\mathcal{X}_N^{\hat{l}}$. Note how the use of $p_m + k\Delta p$, where k is the self-training cycle, acts as a mechanism of *self-paced curriculum learning* on the thresholding process. The maximum percentage, p_M, allows to prevent noise due to accepting too much per-class pseudo-labels eventually with low confidence. Moreover, for any class c, we apply the rule $C_{T_c} \leftarrow \max(C_m, \min(C_{T_c}, C_M))$; where, irrespective of $p\%$, C_m prevents from considering not sufficiently confident pseudo-labels, while C_M ensures to consider pseudo-labels with a sufficiently high confidence. Then, in order to set the final set of pseudo-labels during each cycle, only n of the N pseudo-labeled images are *selected*. In this case, an image-level confidence ranking is established by simply averaging the confidences associated to the pseudo-labels of each image. The top-n most confident images are considered and *fused* with images labeled in previous cycles. If one of the selected n images was already incorporated in previous cycles, we kept the pseudo-labels corresponding to the highest image-level confidence average. The resulting set of pseudo-labeled images is termed as $\mathcal{X}^{\hat{l}}$.

Finally, we use the (labeled) source images, $\mathcal{X}^l$, and the pseudo-labeled target images, $\mathcal{X}^{\hat{l}}$, to *train* a new model, $\mathcal{W}$, by fine-tuning $\mathcal{W}_0$ according to the hyper-parameters $\mathcal{H}_\Phi$ and $\mathcal{M}_{df}$. A parameter we can find in any $\mathcal{H}_\Phi$ is the number of images per mini-batch, N_{MB}. Then, given $\mathcal{M}_{df} = \{p_{MB}, p_{CM}\}$, for training $\mathcal{W}$ we use $p_{MB}N_{MB}$ images from $\mathcal{X}^{\hat{l}}$ and the rest from $\mathcal{X}^l$. In fact, the former undergo a ClassMix-inspired collage transform [72]. In particular, we select $p_{MB}N_{MB}$ images from $\mathcal{X}^l$ and, for each one individually, we gather all the class information (appearance and labels) until considering a p_{CM}% of classes, going from less to more confident ones, which is possible thanks to $\mathcal{V}_{C_T}$. This information is pasted at appearance level (class regions from source on top of target images) and at label level (class labels from source on top of the pseudo-label maps of the target images).

```
Input  : Sets of pseudo-labeled images: $\mathcal{X}^{\hat{l}}_{N_1}, \mathcal{X}^{\hat{l}}_{N_2}$
         Vectors of per-class conf. thr.: $\mathcal{V}_{C_T 1}, \mathcal{V}_{C_T 2}$
         Amount of images to exchange: $n$
         Image-level confidence threshold control: $\lambda$
Output : New sets of ps.-lab. images: $\mathcal{X}^{\hat{l}}_{N_1,new}, \mathcal{X}^{\hat{l}}_{N_2,new}$
  // ClassSort($v$) returns the vector of sorted $v$ indices after
  // sorting by $v$ values, so that $\Delta_i[k]$ is a class index.
                  $\Delta_1$ ← ClassSort($\mathcal{V}_{C_T 2} - \mathcal{V}_{C_T 1}$)
                  $\Delta_2$ ← ClassSort($\mathcal{V}_{C_T 1} - \mathcal{V}_{C_T 2}$)
  // ClassImageList($\mathcal{X}$) returns a vector so that $\mathcal{S}_i[k]$ is the
  // list of images in $\mathcal{X}$ containing pseudo-labels of class $k$.
                  $\mathcal{S}_1$ ← ClassImageList($\mathcal{X}^{\hat{l}}_{N_1}$),
                  $\mathcal{S}_2$ ← ClassImageList($\mathcal{X}^{\hat{l}}_{N_2}$)
  $\mathcal{X}^{\hat{l}}_{N_1,new}, \mathcal{X}^{\hat{l}}_{N_2,new}$ ← $\emptyset, \emptyset$
                  $k, N_c$ ← 0, Num. Classes
repeat
    // $\mathcal{S}_j[\Delta_i[k]] > t$ applies element-wise.
    // Images from $\mathcal{X}^{\hat{l}}_{N_2}$ move to $\mathcal{X}^{\hat{l}}_{N_1,new}$.
    $t_c$ ← $\lambda \max(\mathcal{S}_1[\Delta_2[k]]) + (1-\lambda) \min(\mathcal{S}_1[\Delta_2[k]])$
    $\mathcal{X}^{\hat{l}}_{N_1,new}$ ← Append($\mathcal{X}^{\hat{l}}_{N_1,new}, \mathcal{S}_1[\Delta_2[k]] > t_c$)
    // Images from $\mathcal{X}^{\hat{l}}_{N_1}$ move to $\mathcal{X}^{\hat{l}}_{N_2,new}$.
    $t_c$ ← $\lambda \max(\mathcal{S}_2[\Delta_1[k]]) + (1-\lambda) \min(\mathcal{S}_2[\Delta_1[k]])$
    $\mathcal{X}^{\hat{l}}_{N_2,new}$ ← Append($\mathcal{X}^{\hat{l}}_{N_2,new}, \mathcal{S}_2[\Delta_1[k]] > t_c$)
until $((|\mathcal{X}^{\hat{l}}_{N_1,new}| == |\mathcal{X}^{\hat{l}}_{N_2,new}| == n) \| (k == N_c); k{+}{+})$
return $\mathcal{X}^{\hat{l}}_{N_1,new}, \mathcal{X}^{\hat{l}}_{N_2,new}$
```

Algorithm 3: Collaboration of models

```
Input  : Set of labeled images: $\mathscr{X}^l$
         Set of unlabeled images: $\mathscr{X}^u$
         Net. init. weights & training hyp.-p.: $\mathscr{W}, \mathscr{H}_\Phi$
         Self-t. hyp.-p.: $\mathscr{H}_{st} = \{\mathscr{T}, N, n, K_m, K_M, \mathscr{M}_{df}\}$
         Co-t. hyp.-p.: $\mathscr{H}_{ct} = \{K, w, \lambda\}$
Output : Refined model: $\mathscr{W}$
                                // Initialization
$\mathscr{W}_{0,1}, \mathscr{W}_{0,2} \leftarrow \mathrm{SelfTraining}(\mathscr{X}^l, \mathscr{X}^u, \mathscr{W}, \mathscr{H}_\Phi, \mathscr{H}_{st})$
$\mathscr{X}_1^{\hat{l}}, \mathscr{X}_2^{\hat{l}}, k, \mathscr{V}_{CT1}, \mathscr{V}_{CT2}, \mathscr{W}_1, \mathscr{W}_2 \leftarrow \emptyset, \emptyset, 0, \mathbf{0}, \mathbf{0}, \mathscr{W}_{0,1}, \mathscr{W}_{0,2}$
repeat
    // Co-training Loop
    $\mathscr{X}_N^u \leftarrow \mathrm{Rnd}(\mathscr{X}^u, N)$
    $< \mathscr{X}_{N_1}^{\hat{l}}, \mathscr{V}_{CT1} > \leftarrow \mathrm{Run}(\mathscr{W}_1, \mathscr{X}_N^u, k, \mathscr{T})$
    $< \mathscr{X}_{N_2}^{\hat{l}}, \mathscr{V}_{CT2} > \leftarrow \mathrm{Run}(\mathscr{W}_2, \mathscr{X}_N^u, k, \mathscr{T})$
    $\mathscr{X}_{N_1}^{\hat{l}}, \mathscr{X}_{N_2}^{\hat{l}} \leftarrow \mathrm{Combination}(\mathscr{X}_{N_1}^{\hat{l}}, \mathscr{X}_{N_2}^{\hat{l}})$
    $\mathscr{X}_{N_1}^{\hat{l}}, \mathscr{X}_{N_2}^{\hat{l}} \leftarrow \mathrm{Collaboration}(\mathscr{X}_{N_1}^{\hat{l}}, \mathscr{V}_{CT1}, \mathscr{X}_{N_2}^{\hat{l}}, \mathscr{V}_{CT2}, n, \lambda)$
    $\mathscr{X}_1^{\hat{l}}, \mathscr{X}_2^{\hat{l}} \leftarrow \mathrm{Fuse}(\mathscr{X}_1^{\hat{l}}, \mathscr{X}_{N_1}^{\hat{l}}), \mathrm{Fuse}(\mathscr{X}_2^{\hat{l}}, \mathscr{X}_{N_2}^{\hat{l}})$
    $\mathscr{W}_1 \leftarrow \mathrm{Train}(\mathscr{W}_{0,1}, \mathscr{H}_\mathscr{W}, \mathscr{X}^l, \mathscr{X}_1^{\hat{l}}, \mathscr{M}_{df}, \mathscr{V}_{CT1})$
    $\mathscr{W}_2 \leftarrow \mathrm{Train}(\mathscr{W}_{0,2}, \mathscr{H}_\mathscr{W}, \mathscr{X}^l, \mathscr{X}_2^{\hat{l}}, \mathscr{M}_{df}, \mathscr{V}_{CT2})$
until $(k == K; k{+}{+})$
$\mathscr{W} \leftarrow \mathrm{LastTrain}(w, \mathscr{W}_1, \mathscr{W}_2, \mathscr{H}_\mathscr{W}, \mathscr{X}^l, \mathscr{X}^u, \mathscr{M}_{df})$
return $\mathscr{W}$
```

Algorithm 4: Co-training procedure **Uses:** Algorithms 2 & 3.

3.3.2 Co-training procedure

Algorithm 4 summarizes the co-training procedure supporting the scheme shown in Figure 3.1, which is based on the previous self-training stage (Algorithm 2), on combining pseudo-labels, as well as on a model collaboration stage (Algorithm 3). We detail Algorithm 4 in the following.

Input & output parameters, and Initialization. Since the co-training procedure includes the self-training stage, we have the input parameters required for Algorithm 2. As additional parameters we have $\mathscr{H}_{ct} = \{K, w, \lambda\}$, where K is the maximum number of iterations for mutual model improvement, which we term as *co-training loop*, w is just a selector to be used in the last training step (after the co-training loop), and λ is used during pseudo-label exchange between models. The output parameter, $\mathscr{W}$, is the final model. The co-training procedure starts by running the self-training stage.

Co-training cycles (loop). Similarly to self-training, a co-training cycle starts by obtaining pseudo-labeled images. In this case two sets, $\mathcal{X}^{\hat{l}}_{N_1}$&$\mathcal{X}^{\hat{l}}_{N_2}$, are obtained since we *run* two different models, $\mathcal{W}_1$&$\mathcal{W}_2$. These are applied to the same subset, $\mathcal{X}^{u}_{N}$, of N unlabeled images *randomly selected* from $\mathcal{X}^{u}$. As for self-training, we not only obtain $\mathcal{X}^{\hat{l}}_{N_1}$&$\mathcal{X}^{\hat{l}}_{N_2}$, but also corresponding vectors of per-class confidence thresholds, $\mathcal{V}_{CT\,1}$&$\mathcal{V}_{CT\,2}$. Since, $\mathcal{X}^{\hat{l}}_{N_1}$&$\mathcal{X}^{\hat{l}}_{N_2}$ come from the same $\mathcal{X}^{u}_{N}$ but result from different models, we can perform a simple step of pseudo-label *combination*. In particular, for each image in $\mathcal{X}^{\hat{l}}_{N_i}$, if a pixel has the *void* class as pseudo-label, then, if the pseudo-label for the same pixel of the corresponding image in $\mathcal{X}^{\hat{l}}_{N_j}$ is not *void*, we adopt such pseudo-label, $i \in \{1,2\}, j \in \{1,2\}, i \neq j$. This step reduces the amount of non-labeled pixels, while keeping pseudo-labeling differences between $\mathcal{X}^{\hat{l}}_{N_1}$&$\mathcal{X}^{\hat{l}}_{N_2}$ at non-void pseudo-labels.

Note that co-training strategies assume that the models under collaboration perform in a complementary manner. Therefore, after this basic combination of pseudo-labels, a more elaborated *collaboration* stage is applied, which is described in Algorithm 3. Essentially, n pseudo-labeled images from $\mathcal{X}^{\hat{l}}_{N_i}$ will form the new $\mathcal{X}^{\hat{l}}_{N_j}$ after such collaboration, $i \in \{1,2\}, j \in \{1,2\}, i \neq j$. Thus, along the co-training cycle, pseudo-labeled images arising from $\mathcal{W}_i$ will be used to retain $\mathcal{W}_j$. In particular, visiting first those images containing classes for which $\mathcal{X}^{\hat{l}}_{N_i}$ is more confident than $\mathcal{X}^{\hat{l}}_{N_j}$, sufficiently high confident images in $\mathcal{X}^{\hat{l}}_{N_i}$ are selected for the new $\mathcal{X}^{\hat{l}}_{N_j}$ set, until reaching n. The class confidences of $\mathcal{X}^{\hat{l}}_{N_i}$&$\mathcal{X}^{\hat{l}}_{N_j}$ are given by the respective $\mathcal{V}_{CT\,1}$&$\mathcal{V}_{CT\,2}$, while the confidence of a pseudo-labeled image is determined as the average of the confidences of its pseudo-labels. Being sufficiently high confident means that the average is over a dynamic threshold controlled by the λ parameter.

Once this process is finished, we have two new sets of pseudo-labels, $\mathcal{X}^{\hat{l}}_{N_1}$&$\mathcal{X}^{\hat{l}}_{N_2}$, which are used separately for finishing the co-training cycle. In particular, each new $\mathcal{X}^{\hat{l}}_{i}$ is used as its self-training counterpart (see $\mathcal{X}^{\hat{l}}$ in the loop of Algorithm 2), i.e., performing the fusion with the corresponding set of pseudo-labels from previous cycles and fine-tuning of $\mathcal{W}_{0,i}$. Finally, once the co-training loops finishes, a last train is performed. In this case, the full $\mathcal{X}^{u}$ is used to produce pseudo-labels. For this task, we can use an ensemble of $\mathcal{W}_1$ and $\mathcal{W}_2$ (e.g., averaging confidences), or any of these two models individually. This option is selected according to the parameter w. In this last training, the ClassMix-inspired procedure is not applied, but mixing source and target images at mini-batch level still is done according to the value $p_{MB} \in \mathcal{M}_{df}$. It is also worth noting that, inside co-training loop, the two

Run() operations can be parallelized, and the two Train() too.

3.4 Experimental results

3.4.1 Datasets and evaluation

Our experiments rely on three well-known synthetic datasets used for UDA semantic segmentation as source data, namely, GTAV [85], SYNTHIA [86] and Synscapes [119]. GTAV is composed by 24,904 images with a resolution of 1914 × 1052 pixels directly obtained from the render engine of the videogame GTA V. Synscapes is composed by 25,000 images with a resolution of 1440 × 720 pixels of urban scenes, obtained by using a physic-based rendering pipeline. SYNTHIA is composed by 9,000 images of urban scenes highly populated, with a resolution of 1280 × 760 pixels, generated by a videogame-style rendering pipeline based on the Unity3D framework. As real-world datasets (target domain) we rely on Cityscapes [15], BDD100K [131] and Mapillary Vistas [70]. Cityscapes is a popular dataset composed of onboard images acquired at different cities in Germany under clean conditions (e.g., no heavy occlusions or bad weather), it is common practice to use 2,975 images for training semantic segmentation models, and 500 images for reporting quantitative results. The latter are known as validation set. Cityscapes images have a resolution of 2048 × 1024 pixels. Another dataset is BDD100K, which contains challenging onboard images taken from different vehicles, in different US cities, and under diverse weather conditions. The dataset is divided into 7,000 images for training purposes and 1,000 for validation. However, a high amount of training images are heavily occluded by the ego vehicle, thus, for our experiments we rely on a occlusion-free training subset of 1,777 images. Image resolution is 1280 × 720 pixels. Finally, Mapillary Vistas is composed by high resolution images of street views around the world. These images have a high variation in resolutions and aspect ratios due to the fact that are taken from diverse devices like smartphones, tablets, professional cameras, etc. For simplicity, we only consider those images with aspect ratio 4:3, which, in practice, are more than the 75%. Then, we have 14,716 images for training and 1,617 for validation.

As is common practice, we evaluate the performance of our system on the validation set of each real-world (target) dataset using the 19 official classes defined for Cityscapes. These 19 classes are common in all the datasets except in SYNTHIA that only contains 16 of these 19 classes, additional dataset-specific classes are ignored for training and evaluation. Note that, although there are semantic labels available for the target datasets, for performing UDA we ignore them at training time, and we use them at validation time. In other words, we only use the semantic

Table 3.1: Hyper-parameters of our method (Sect. 3.3 and Alg. 2–4). Datasets: GTAV (G), Synscapes (S), STNTHIA (SIA), Cityscapes (C), BDD (B), Mapillary Vistas (M).

								$\mathcal{M}_{df}$		$\mathcal{H}_{st}$				$\mathcal{H}_{ct}$				
Source	***Target***	N	n	Δp	C_m	C_M	N_{MB}	p_{MB}	p_{CM}	P_m	P_M	K_m	K_M	P_m	P_M	K	w	λ
SIA	C	500	100	0.05	0.5	0.9	4	0.75	0.5	0.5	0.6	1	10	0.5	0.6	5	1	0.8
S, G+S	C, M	500	100	0.05	0.5	0.9	4	0.75	0.5	0.5	0.6	1	10	0.5	0.6	5	1	0.8
G	C	500	100	0.05	0.5	0.9	4	0.75	0.5	0.3	0.5	1	10	0.5	0.6	5	1	0.8
G+S	B	500	100	0.05	0.5	0.9	2	0.5	0.5	0.3	0.5	1	10	0.5	0.6	5	1	0.8

labels of the validation sets, and with the only purpose of reporting quantitative results. All the synthetic datasets provide semantic labels, since they act as source domain, we use them. In addition, we note that for our experiments we do not perform any learnable image-to-image transform to align synthetic and real-world domains (like GAN-based ones). However, following [36], we perform synth-to-real LAB space alignment as pre-processing step.

As is standard, quantitative evaluation relies on PASCAL VOC intersection-over-union metric $IoU = TP/(TP + FP + FN)$ [22], where TP, FP, and FN refer to true positives, false positives, and false negatives, respectively. IoU can be computed per class, while using a mean IoU (mIoU) to consider all the classes at once.

3.4.2 Implementation details

We use the Detectron2 [121] framework and leverage their implementation of DeepLabV3+ for semantic segmentation, with ImageNet weight initialization. We chose V3+ version of DeepLab instead the V2 because it provides a configuration which fits well in our 12GB-memory GPUs, turning out in a ×2 training speed over the V2 configuration and allowing a higher batch size. Other than this, V3+ does not provide accuracy advantages over V2. We will see it when discussing Table 3.2, where the baselines of V3+ and V2 performs similarly (SYNTHIA case) or V3+ may perform worse (GTAV case). The hyper-parameters used by our co-training procedure are set according to Table 3.1. Since their meaning is intuitive, we just tested some reasonable values, but did not perform hyper-parameter search. As we can see in Table 3.1 they are pretty similar across datasets. This table does not include the hyper-parameter related to the training of DeepLabV3+, termed as $\mathcal{H}_{\Phi}$ in Algorithms 2–4, since they are not specific of our proposal. Thus, we summarize them in the following.

For training the semantic segmentation models, we use SGD optimizer with a starting learning rate of 0.002 and momentum 0.9. We crop the training images to 1024 × 512 pixels, 816 × 608, and 1280 × 720, when we work with Cityscapes, Mapillary Vistas, and BDD100K, respectively. Considering this cropping and our available hardware, we set batch sizes (N_{MB}) of 4 images, 4, and 2, for these datasets, respectively. Moreover, we perform data augmentation consisting of random zooms and horizontal flips. For computing each source-only baseline model ($\mathscr{W}_0$ in Algorithm 2) and the final model (returned $\mathscr{W}$ in Algorithm 4) we set the number of iterations to 60K when we work with Cityscapes and Mapillary Vistas, and 120K for BDD100K to maintain consistency given the mentioned batch sizes. The number of iterations for the self-training stage and the co-training loop are equally set to 8K for Cityscapes and Mapillary Vistas, and 16K for BDD100K.

In addition, on training only using GTAV, a class balancing sample policy (CB) is applied. Due to the scarcity of samples from several classes (e.g., bicycle, train, rider and motorcycle), these are under-represented during training. A simple, yet efficient, method to balance the frequency of samples from these classes is computing individual class frequency in the whole training dataset and apply a higher selection probability for the under-represented classes. The other synthetic datasets in isolation and the combination of GTAV + Synscapes are already well balanced and we do not need to apply this technique.

Table 3.2: UDA results. mIoU considers the 19 classes. mIoU* considers 13 classes, which only applies to SYNTHIA, where classes with '*' are not considered for global averaging and those with '-' scores do not have available samples. **Bold** stands for best, and underline for second best. In this table, the target domain is always Cityscapes.

Methods	Road	Sidewalk	Building	Wall*	Fence*	Pole*	Traffic light	Traffic Sign	Vegetation	Terrain	Sky	Person	Rider	Car	Truck	Bus	Train	Motorbike	Bike	mIoU*	mIoU	Baseline	Δ(Diff.)
											SYNTHIA (Source) → Cityscapes												
AdSegNet [105]	81.7	39.1	78.4	11.1	0.3	25.8	6.8	9.0	79.1	-	80.8	54.8	21.0	66.8	-	34.7	-	13.8	29.9	45.8	39.6	33.5	+6.1
IntraDA [73]	84.3	37.7	79.5	5.3	0.4	24.9	9.2	8.4	80.0	-	84.1	57.2	23.0	78.0	-	38.1	-	20.3	36.5	48.9	41.7	33.5	+8.2
CBST [146]	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	48.9	42.6	29.2	+13.4
CRST [147]	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	50.1	43.8	34.9	+8.9
DACS [103]	80.5	25.1	81.9	21.4	2.8	37.2	22.6	23.9	83.6	-	90.7	67.6	38.3	82.9	-	38.9	-	28.4	47.5	54.8	48.3	29.4	+18.9
DSP [27]	86.4	42.0	82.0	2.1	1.8	34.0	31.6	33.2	87.2	-	88.5	64.1	31.9	83.8	-	65.4	-	28.8	54.0	59.9	51.0	33.5	+17.5
MFA [137]	81.8	40.2	85.3	-	-	-	38.0	33.9	82.3	-	82.0	73.7	41.1	87.8	-	56.6	-	46.3	63.8	62.5	-	-	-
RED [9]	88.6	46.6	83.7	22.6	4.1	35.0	35.9	36.1	82.8	-	81.3	61.6	32.1	87.9	-	52.7	-	31.9	57.6	59.9	52.5	35.3	+17.2
ProDA [138]	87.8	45.7	84.6	37.1	0.6	44.0	54.6	37.0	88.1	-	84.4	74.2	24.3	88.2	-	51.1	-	40.5	45.6	62.0	55.5	34.9	**+20.6**
Co-T (ours)	78.1	36.9	84.0	9.3	0.2	47.4	49.2	19.3	89.0	-	89.6	77.9	52.3	91.5	-	60.3	-	47.1	64.7	**64.6**	**56.0**	**35.4**	**+20.6**
											GTAV (Source) → Cityscapes												
AdSegNet [105]	86.5	36.0	79.9	23.4	23.3	23.9	35.2	14.8	83.4	33.3	75.6	58.5	27.6	73.7	32.5	35.4	3.9	30.1	28.1	-	42.4	36.6	+5.8
IntraDA [73]	90.6	36.1	82.6	29.5	21.3	27.6	31.4	23.1	85.2	39.3	80.2	59.3	29.4	86.4	33.6	53.9	0.0	32.7	37.6	-	46.3	36.6	+9.7
CBST [146]	89.6	58.9	78.5	33.0	22.3	41.4	48.2	39.2	83.6	24.3	65.4	49.3	20.2	83.3	39.0	48.6	12.5	20.3	35.3	-	47.0	35.4	+11.6
CRST [147]	91.7	45.1	80.9	29.0	23.4	43.8	47.1	40.9	84.0	20.0	60.6	64.0	31.9	85.8	39.5	48.7	25.0	38.0	47.0	-	49.8	35.4	+14.4
DACS [103]	89.9	39.6	87.8	30.7	39.5	38.5	46.4	52.7	87.9	43.9	88.7	67.2	35.7	84.4	45.7	50.1	0.0	27.2	33.9	-	52.1	32.8	+19.3
DSP [27]	92.4	48.0	87.4	33.4	35.1	36.4	41.6	46.0	87.7	43.2	89.8	66.6	32.1	89.9	57.0	56.1	0.0	44.1	57.8	-	55.0	36.6	+18.4
MFA [137]	94.5	61.1	87.6	41.4	35.4	41.2	47.1	45.7	86.6	36.6	87.0	70.1	38.3	87.2	39.5	54.7	0.3	45.4	57.7	-	55.7	**45.6**	+10.1
RED [9]	94.4	60.9	88.0	39.4	41.8	43.2	49.0	56.0	88.0	45.8	87.7	67.5	38.0	90.0	57.6	51.9	0.0	46.5	55.2	-	57.9	34.8	+23.1
ProDA [138]	87.8	56.0	79.7	46.3	44.8	45.6	53.5	53.5	88.6	45.2	82.1	70.7	39.2	88.8	45.5	59.4	1.0	48.9	56.4	-	57.5	36.6	+20.9
Co-T (ours)	89.9	51.0	89.0	40.0	34.2	51.6	56.5	51.3	89.5	50.1	89.8	71.8	46.5	90.9	55.7	56.7	0.0	52.6	64.2	-	**59.5**	28.5	**+31.0**
											Synscapes (Source) → Cityscapes												
AdSegNet [105]	94.2	60.9	85.1	29.1	25.2	38.6	43.9	40.8	85.2	29.7	88.2	64.4	40.6	85.8	31.5	43.0	28.3	30.5	56.7	-	52.7	**45.3**	+7.4
IntraDA [73]	94.0	60.0	84.9	29.5	26.2	38.5	41.6	43.7	85.3	31.7	88.2	66.3	44.7	85.7	30.7	53.0	29.5	36.5	60.2	-	54.2	**45.3**	+8.9
Co-T (ours)	91.4	55.7	81.6	34.5	38.9	53.6	64.7	67.4	91.0	48.7	93.4	77.5	42.4	93.1	18.3	20.8	1.2	60.0	74.2	-	**58.3**	45.0	**+13.3**
											GTAV + Synscapes (Source) → Cityscapes												
MADAN [141]	94.1	61.0	86.4	43.3	32.1	40.6	49.0	44.4	87.3	47.7	89.4	61.7	36.3	87.5	35.5	45.8	31.0	33.5	52.1	-	55.7	**51.6**	+4.1
MsDACL [36]	93.6	59.6	87.1	44.9	36.7	42.1	49.9	42.5	87.7	47.6	89.9	63.5	40.3	88.2	41.0	58.3	53.1	37.9	57.7	-	59.0	**51.6**	+7.4
Co-T (ours)	96.3	74.7	90.4	48.8	49.1	58.3	61.5	67.0	90.7	54.7	93.5	79.4	57.7	90.4	45.6	85.1	59.9	60.4	70.9	-	**70.2**	50.0	**+20.2**

3.4.3 Comparison with the state of the art

In Table 3.2 we compare our co-training procedure with state-of-the-art methods when using Cityscapes as target domain. We divide the results into four blocks according to the source images we use: SYNTHA, GTAV, Synscapes, or GTAV+Synscapes. Most works in the literature present their results only using GTAV or SYNTHIA as source data. We obtain the best results in the SYNTHIA case, with 56 mIoU (19 classes), and for GTAV with 59.5 mIoU. On the other hand, each proposal from the literature uses their own CNN architecture and pre-trained models. Thus, we have added the mIoU score of the baseline that each work uses as starting point to improve according to the corresponding proposed method. Then, we show the difference between the final achieved mIoU score and the baseline one. In Table 3.2 this corresponds to column Δ(Diff.). Note how our method reaches 20.6 and 31.0 points of mIoU increment on SYNTHIA and GTAV, respectively. The highest for GTAV, and the highest for SYNTHIA on pair with ProDA proposal. Additionally, for the shake of completeness, we have added the mIoU scores for the 13 classes setting of SYNTHIA since it is also a common practice in the literature. We can see that co-training obtains the best mIoU too. On the other hand, we are mostly interested in the 19 classes setting. Using Synscapes as source data we achieved state-of-the-art results in both Δ(Diff.) (13.3 points) and final mIoU score (58.3). Note that, in this case, our baseline score is similar to the ones reported in previous literature.

By performing a different LAB transform for each synthetic dataset individually, our co-training procedure allows to joint them as if they were one single domain. Thus, we have considered this setting too. Preliminary baseline experiments (i.e., without performing co-training) showed that the combinations GTAV + Synscapes and GTAV + Synscapes + SYNTHIA are the best performing, with a very scarce mIoU difference between them (0.62). Thus, for the shake of bounding the number of experiments, we have chosen GTAV + Synscapes as the only case combining datasets, so also avoiding the problem of the 19 vs. 16 classes discrepancy when SYNTHIA is combined with them. In fact, using GTAV + Synscapes, we reach a Δ(Diff.) of 20.2 points, with a final mIoU of 70.2, which outperforms the second best in 11.2 points, and it clearly improves the mIoU with respect to the use of these synthetic datasets separately (15.6 points comparing to GTAV, 11.9 for Synscapes). Again, in this case, our baseline score is similar to the ones reported in previous literature.

Table 3.3: Co-training results compared to baseline (Source), LAB adjustment pre-processing (SrcLAB), self-training stage, and upper-bound (SrcLAB + Target). Note that Target refers to using the 100% of the target domain images labeled for training by a human oracle. We apply target LAB adjustment on the synthetic datasets before self-training stage and co-training. In this table, the source data are SYNTHIA and the target domain is Cityscapes.

Methods	Road	Sidewalk	Building	Wall	Fence	Pole	Traffic light	Traffic Sign	Vegetation	Terrain	Sky	Person	Rider	Car	Truck	Bus	Train	Motorbike	Bike	mIoU
SYNTHIA (Source) → Cityscapes																				
Source	38.47	17.42	75.39	4.92	0.22	30.58	17.79	15.48	73.86	-	80.55	63.28	22.57	62.47	-	27.12	-	13.22	23.92	35.46
SrcLAB	58.5	22.23	78.33	6.03	0.28	37.8	17.0	15.64	78.01	-	79.36	63.05	22.48	78.91	-	32.1	-	13.31	28.8	39.48
Self-training stage (ours)	72.14	30.37	82.97	2.97	0.11	43.72	34.49	14.00	87.09	-	86.87	73.82	43.45	87.57	-	42.44	-	21.69	56.36	48.74
Co-training proce. (ours)	78.14	36.98	84.07	9.34	0.28	47.49	49.2	19.35	89.07	-	89.62	77.92	52.32	91.50	-	60.37	-	47.10	64.76	56.09
SrcLAB + Target	97.92	84.42	92.60	53.87	61.70	65.93	70.67	78.00	92.71	(65.68)	94.98	83.29	66.49	95.32	(77.37)	88.20	(71.84)	67.45	78.13	79.48
Δ(SrcLAB vs. Source)	20.03	4.81	2.94	1.11	0.06	7.22	-0.79	0.16	4.15	-	-1.19	-0.23	-0.09	16.44	-	4.98	-	0.09	4.88	4.02
Δ(Co-t vs. Source)	39.62	18.38	8.24	4.33	0.06	16.84	31.41	3.87	13.81	-	9.04	14.6	29.66	26.29	-	19.71	-	33.86	40.74	19.40
Δ(Co-t vs. SrcLAB)	19.64	14.75	5.74	3.31	0	9.69	32.2	3.71	11.06	-	10.26	14.87	29.84	12.59	-	28.27	-	33.79	35.96	16.6
Δ(Co-t vs. Self-t)	6.00	6.61	1.10	6.37	0.17	3.77	14.71	5.35	1.98	-	2.75	4.10	8.87	3.93	-	17.93	-	25.41	8.40	7.34
Δ(Co-t vs. SrcLAB + Tgt)	-19.83	-48.62	-8.97	-44.62	-61.42	-18.51	-21.47	-58.65	-5.04	-	-5.39	-5.41	-14.26	-6.56	-	-41.37	-	-20.37	-13.47	-24.62

Table 3.4: Analogous to Table 3.3 using GTAV as source.

Methods	Road	Sidewalk	Building	Wall	Fence	Pole	Traffic light	Traffic Sign	Vegetation	Terrain	Sky	Person	Rider	Car	Truck	Bus	Train	Motorbike	Bike	mIoU
GTAV (Source) → Cityscapes																				
Source	51.85	13.57	64.71	8.19	15.86	14.39	31.66	10.86	71.95	6.91	38.21	55.65	22.21	72.40	32.62	9.76	0.0	9.93	11.14	28.52
SrcLAB	75.25	23.38	76.59	19.72	16.86	32.28	28.37	13.73	81.75	25.47	46.71	64.0	31.76	84.14	32.29	16.23	0.08	23.41	27.27	37.86
SrcLAB + CB	73.34	26.30	73.50	29.57	21.16	35.04	42.78	20.09	84.64	26.48	53.20	63.02	40.77	81.90	34.16	31.56	4.74	36.05	34.07	42.76
Self-training stage (ours)	85.31	36.82	85.11	41.09	25.62	46.39	45.19	33.44	88.98	45.55	72.99	69.54	42.43	89.36	44.42	57.5	1.28	45.51	59.78	53.49
Co-training proce. (ours)	89.92	51.03	89.09	40.05	34.23	51.61	56.54	51.36	89.50	50.12	89.83	71.88	46.50	90.91	55.72	56.77	0.0	52.61	64.21	59,57
SrcLAB + Target	98.20	85.43	92.74	59.07	63.05	65.26	69.43	77.10	92.63	65.26	94.70	82.11	63.22	95.22	85.05	86.07	67.27	64.84	77.21	78.10
Δ(SrcLAB + CB vs. Source)	21.49	12.73	8.79	21.38	5.3	20.65	11.12	9.23	12.69	19.57	14.99	7.37	18.56	9.5	1.54	21.8	4.74	26.12	22.93	14.24
Δ(Co-t vs. Source)	38.07	37.46	24.38	31.86	18.37	37.22	24.88	40.5	17.55	43.21	51.62	16.23	24.29	18.51	23.1	47.01	0.0	42.68	53.07	31.05
Δ(Co-t vs. SrcLAB + CB)	16.58	24.73	15.59	10.48	13.07	16.57	13.76	31.27	4.86	23.64	36.63	8.86	5.73	9.01	21.56	25.21	-4.74	16.56	30.14	16.81
Δ(Co-t vs. Self-t)	4.61	14.21	3.98	-1.04	8.61	5.22	11.35	17.92	0.52	4.57	16.84	2.34	4.07	1.55	11.3	-0.73	-1.28	7.1	4.43	6.08
Δ(Co-t vs. SrcLAB + Tgt)	-8.28	-34.4	-3.65	-19.02	-28.82	-13.65	-12.89	-25.74	-3.13	-15.14	-4.87	-10.23	-16.72	-4.31	-29.33	-29.3	-67.27	-12.23	-13.0	-18.53

Table 3.5: Analogous to Table 3.3 using Synscapes as source.

Methods	Road	Sidewalk	Building	Wall	Fence	Pole	Traffic light	Traffic Sign	Vegetation	Terrain	Sky	Person	Rider	Car	Truck	Bus	Train	Motorbike	Bike	mIoU
Synscapes (Source) → Cityscapes																				
Source	83.81	42.15	61.87	26.10	21.69	44.65	47.12	53.86	81.30	33.57	53.53	67.79	29.68	85.66	14.81	6.66	2.36	34.94	63.53	45.01
SrcLAB	78.39	37.47	67.39	16.45	19.09	48.5	51.79	58.54	83.18	29.89	64.79	70.17	29.27	85.39	18.42	10.42	3.32	36.48	64.61	45.98
Self-training stage (ours)	89.55	50.19	84.26	33.61	37.67	57.29	60.11	64.00	90.61	47.13	91.22	72.15	21.17	91.99	15.38	20.09	9.35	44.94	70.78	55.34
Co-training proce. (ours)	91.46	55.76	81.63	34.58	38.92	53.66	64.74	67.43	91.02	48.72	93.45	77.54	42.40	93.14	18.35	20.84	1.29	60.03	74.22	58.38
SrcLAB + Target	98.03	84.49	92.90	59.10	63.70	67.18	71.67	79.50	92.74	65.51	94.81	83.93	68.07	95.45	82.89	91.83	83.79	70.91	79.24	80.30
Δ(SrcLAB vs. Source)	0.97	-5.42	-4.68	5.52	-9.65	-2.6	3.85	4.67	4.68	1.88	-3.68	11.26	2.38	-0.41	-0.27	3.61	3.76	0.96	1.54	1.08
Δ(Co-t vs. Source)	7.65	13.61	19.76	8.48	17.23	9.01	17.62	13.57	9.72	15.15	39.92	9.75	12.72	7.48	3.54	14.18	-1.07	25.09	10.69	13.37
Δ(Co-t vs. SrcLAB)	13.07	18.29	14.24	18.13	19.83	5.16	12.95	8.89	7.84	18.83	28.66	7.37	13.13	7.75	-0.07	10.42	-2.03	23.55	9.61	12.40
Δ(Co-t vs. Self-t)	1.91	5.57	-2.63	0.97	1.25	-3.63	4.63	3.43	0.41	1.59	2.23	5.39	21.23	1.15	2.97	0.75	-8.06	15.09	3.44	3.04
Δ(Co-t vs. SrcLAB + Tgt)	-6.57	-28.73	-11.27	-24.52	-24.78	-13.52	-6.93	-12.07	-1.72	-16.79	-1.36	-6.39	-25.67	-2.31	-64.54	-70.99	-82.5	-10.88	-5.02	-21.92

Table 3.6: Analogous to Table 3.3 using GTAV + Synscapes as source.

Methods	Road	Sidewalk	Building	Wall	Fence	Pole	Traffic light	Traffic Sign	Vegetation	Terrain	Sky	Person	Rider	Car	Truck	Bus	Train	Motorbike	Bike	mIoU
GTAV + Synscapes (Source) → Cityscapes																				
Source	66.39	33.54	79.58	29.43	40.24	49.73	56.12	46.51	81.22	18.40	79.06	73.18	29.67	85.25	43.00	6.46	23.02	47.71	61.63	50.01
SrcLAB	87.97	47.45	85.14	34.31	43.16	49.82	57.16	47.85	88.88	45.00	82.53	72.58	38.22	89.16	51.91	61.31	40.06	43.64	60.85	59.32
Self-training stage (ours)	93.93	66.08	89.95	46.40	48.13	56.30	59.65	65.16	90.25	52.22	93.33	75.97	41.15	90.40	44.98	75.08	65.52	55.52	71.98	67.47
Co-training proce. (ours)	96.30	74.72	90.44	48.89	49.15	58.36	61.52	67.05	90.75	54.75	93.52	79.48	57.71	90.48	45.61	85.11	59.95	60.41	70.96	70.23
SrcLAB + Target	97.88	83.43	92.33	64.15	61.77	63.45	68.04	75.17	92.31	62.74	94.08	82.00	64.16	95.01	84.25	89.66	75.56	63.28	76.07	78.18
Δ(SrcLAB vs. Source)	21.58	13.91	5.56	4.88	2.92	0.09	1.04	1.34	7.66	26.6	3.47	-0.6	8.55	3.91	8.91	54.85	17.04	-4.07	-0.78	9.31
Δ(Co-t vs. Source)	29.91	41.18	10.86	19.46	8.91	8.63	5.40	20.54	9.53	36.35	14.46	6.3	28.04	5.23	2.61	78.65	36.93	12.70	9.33	20.22
Δ(Co-t vs. SrcLAB)	8.33	27.27	5.30	14.58	5.99	8.54	4.36	19.20	1.87	9.75	10.99	6.9	19.49	1.32	-6.3	23.8	19.89	16.77	9.28	10.91
Δ(Co-t vs. Self-t)	2.37	8.64	0.49	2.49	1.02	2.06	1.87	1.89	0.5	2.53	0.19	3.51	16.56	0.08	0.63	10.03	-5.57	4.89	-1.85	2.76
Δ(Co-t vs. SrcLAB + Tgt)	-1.58	-8.71	-1.89	-15.26	-12.62	-5.09	-6.52	-8.12	-1.56	-7.99	-0.56	-2.52	-6.45	-4.53	-38.64	-4.55	-15.61	-2.87	-5.94	-7.95

Table 3.7: Analogous to Table 3.6 with BDD100K and Mapillary as target domains.

Methods	Road	Sidewalk	Building	Wall	Fence	Pole	Traffic light	Traffic Sign	Vegetation	Terrain	Sky	Person	Rider	Car	Truck	Bus	Train	Motorbike	Bike	mIoU
GTAV + Synscapes (Source) → BDD100K																				
Source	67.83	20.84	54.86	9.00	27.57	30.24	31.74	20.75	62.69	15.39	63.75	54.53	24.08	65.92	12.82	9.10	0.07	39.58	39.04	34.20
SrcLAB	74.22	26.07	68.48	7.94	15.51	31.09	38.69	22.90	69.33	25.92	74.27	59.35	18.81	72.79	23.66	19.75	0.02	54.72	35.48	38.68
Self-training stage (ours)	88.52	26.21	78.77	14.48	35.41	41.40	49.27	31.74	75.86	35.89	88.85	60.39	35.22	85.48	35.04	42.29	0.00	51.28	47.40	48.60
Co-training proce. (ours)	88.43	31.63	80.05	13.05	39.89	41.81	46.12	29.67	76.06	37.79	89.50	63.08	39.94	85.78	40.52	42.71	0.07	53.60	50.40	50.11
SrcLAB + Target	93.33	60.89	84.41	31.45	47.74	49.62	55.33	47.77	85.00	42.77	92.60	66.10	38.91	88.11	40.63	71.09	0.00	57.71	54.90	58.33
Δ(SrcLAB vs. Source)	6.39	5.23	9.62	-1.06	-12.06	0.85	6.95	2.15	6.64	10.53	10.52	4.82	-5.27	6.87	10.84	10.65	-0.05	15.14	-3.56	4.48
Δ(Co-t vs. Source)	25.5	40.05	29.55	22.45	20.17	19.38	23.59	27.02	22.31	27.38	28.85	11.57	14.83	22.19	27.81	61.99	-0.07	18.13	15.86	24.13
Δ(Co-t vs. SrcLAB)	19.11	34.82	19.93	23.51	32.23	18.53	16.64	24.87	15.67	16.85	18.33	6.75	20.1	15.32	16.97	51.34	-0.02	2.99	19.42	19.65
Δ(Co-t vs. Self-t)	-0.09	5.42	1.28	-1.43	4.48	0.41	-3.15	-2.07	0.2	1.9	0.65	2.69	4.72	0.3	5.48	0.42	0.07	2.32	3.0	1.51
Δ(Co-t vs. SrcLAB + Tgt)	-4.9	-29.26	-4.36	-18.4	-7.85	-7.81	-9.21	-18.1	-8.94	-4.98	-3.1	-3.02	1.03	-2.33	-0.11	-28.38	0.07	-4.11	-4.5	-8.22
GTAV + Synscapes (Source) → Mapillary Vistas																				
Source	68.81	31.73	68.88	25.20	37.94	38.79	49.79	20.57	73.27	29.66	80.62	63.81	42.75	80.65	35.74	16.86	1.85	44.56	47.09	45.19
SrcLAB	72.62	43.18	70.89	17.21	25.18	35.05	57.74	55.73	76.78	27.09	88.72	71.34	24.34	77.89	46.29	47.37	0.00	34.27	46.77	48.34
Self-training stage (ours)	89.44	53.30	85.28	36.57	44.89	47.10	59.18	65.94	84.58	48.25	97.44	74.23	55.71	89.37	58.34	59.45	1.47	49.44	51.50	60.60
Co-training proce. (ours)	90.44	57.83	85.59	36.38	45.56	49.64	59.73	67.62	84.27	47.08	96.79	74.80	56.05	90.42	56.34	49.86	10.71	49.62	55.94	61.30
SrcLAB + Target	94.02	69.46	88.70	51.38	60.17	57.59	64.21	75.16	90.70	69.35	98.27	76.02	56.70	91.42	60.49	73.35	33.81	60.87	66.63	70.44
Δ(SrcLAB vs. Source)	-2.93	-0.24	0.73	-8.10	-8.64	5.25	5.94	21.32	3.34	2.73	1.74	3.47	4.75	1.86	5.84	-3.15	1.09	7.51	6.69	2.59
Δ(Co-t vs. Source)	21.63	26.10	16.71	11.18	7.62	10.85	9.94	47.05	11.0	17.42	16.17	10.99	13.30	9.77	20.60	33.0	8.86	5.06	8.85	16.11
Δ(Co-t vs. SrcLAB)	24.56	26.34	15.98	19.28	16.26	5.6	4.0	25.73	7.66	14.69	14.43	7.52	8.55	7.91	14.76	36.15	7.77	-2.45	2.16	13.52
Δ(Co-t vs. Self-t)	1.0	4.53	0.31	-0.19	0.67	2.54	0.55	1.68	-0.31	-1.17	-0.65	0.57	0.34	1.05	-2.0	-9.59	9.24	0.18	4.44	0.7
Δ(Co-t vs. SrcLAB + Tgt)	-3.58	-11.63	-3.11	-15.0	-14.61	-7.95	-4.48	-7.54	-6.43	-22.27	-1.48	-1.22	-0.65	-1.0	-4.15	-23.49	-23.1	-11.25	-10.69	-9.14

Table 3.8: Contribution of the main components of our proposal. Case study: GTAV + Synscapes → Cityscapes.

			Co-training Procedure			
			Self-training Stage			
	Baseline	+ LAB	+ MixBatch	+ ClassMix	+ Co-training loop	Upper-bound
mIoU	50.01	59.32	66.18	67.47	70.23	78.18
Gain	-	+9.31	+6.86	+1.29	+2.76	-

3.4.4 Ablative study and Qualitative results

In Tables 3.3, 3.4,3.5 and 3.6 we compare co-training results with corresponding baselines and upper-bounds. We also report the results of applying LAB adjustment as only UDA step, as well as the results from one of the models obtained after our self-training stage (we chose the model from the last cycle). Overall, in all cases the co-training loop (which completes the co-training procedure) improves the self-training stage, and this stage, in turn, improves over LAB adjustment. Moreover, when combining GTAV + Synscapes we are only 7.95 mIoU points below the upper-bound, after improving ~ 20.22 mIoU points the baseline.

To complement our experimental analysis, we summarize in Table 3.8 the contribution of the main components of our proposal for the case GTAV + Synscapes → Cityscapes. First, we can see how a proper pre-processing of the data is relevant. In particular, performing synth-to-real LAB space alignment already allows to improve 9.31 points of mIoU. This contribution can also be seen in Tables 3.3-3.6 and 3.7, where improvements range from 2.59 mIoU points (GTAV+Synscapes→Mapillary Vistas) to 9.34 (GTAV→Cityscapes). This LAB adjustment is a step hardly seen in synth-to-real UDA literature which should not be ignored. Then, back to Table 3.8, we see that properly combining labeled source images and pseudo-labeled target images (MixBatch) is also relevant since it provides an additional gain of 6.86 points. Note that this MixBatch is basically the *cool world* idea which we can trace back to work of our own lab done before the deep learning era in computer vision [109]. In addition, performing our ClassMix-inspired collage also contributes with 1.29 points of mIoU, and the final collaboration of models returns 2.76 additional points of mIoU. Overall, the main components of our synth-to-real UDA procedure contribute with 10.91 points of mIoU and LAB alignment 9.31 points. We conclude that all the components of the proposed procedure are relevant.

In order to confirm these positive results, we applied our method to two additional target domains which are relatively challenging, namely, Mapillary Vistas and BDD100K. In fact, up to the best of our knowledge, in the current literature

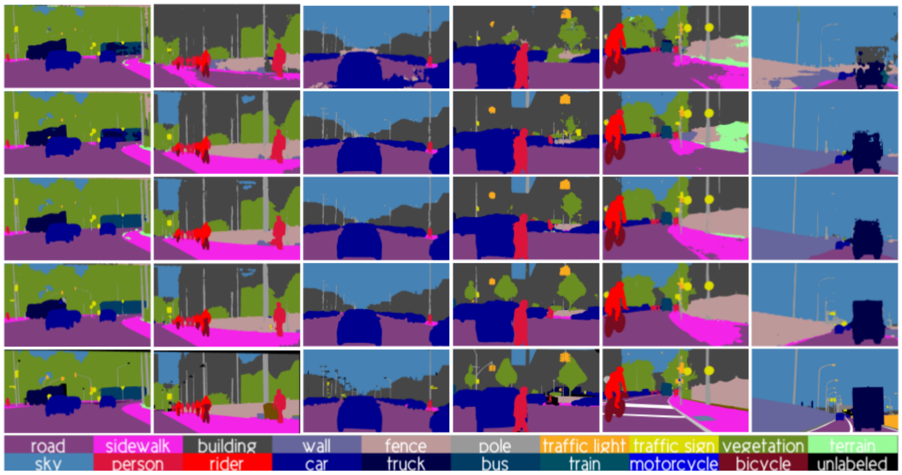


Figure 3.2: Qualitative results using GTAV + Synscapes as source domain. From left to right, the two first columns correspond to Cityscapes in the role of target domain, next two columns to BDD100K, and last two to Mapillary Vistas. Top to bottom rows correspond to SrcLAB, self-training stage, full co-training procedure, upper-bound, and ground truth, respectively.

there are not synth-to-real UDA semantic segmentation results reported for them. Our results can be seen in Table 3.7, directly focusing on the combination of GTAV + Synscapes as source domain. In this case, the co-training loop improves less over the intermediate self-training stage. Still, for BDD100K the final mIoU is only 8.22 mIoU points below the upper-bound, after improving 24.13 mIoU points the baseline. For Mapillary Vistas our methods remains only 9.14 mIoU points below the upper-bound and improves 16.11 mIoU points the baseline. Up to the best of our knowledge, this are state-of-the-art results for BDD100K and Mapillary Vistas when addressing synth-to-real UDA semantic segmentation.

Figure 3.2 presents qualitative results of semantic segmentation for the different real-world (target) datasets, when using GTAV + Synscapes as source domain. We observe how the baselines have problems with dynamic objects (e.g., cars, trucks) and some infrastructure classes like sidewalk are noisy. The self-training stage mitigates the problems observed in the only-source (with LAB adjustment) results to a large extent. However, we can still observe instabilities in classes like truck or bus, which the co-training loop (full co-training procedure) achieves to address properly. Nevertheless, the co-training procedure is not perfect and several errors

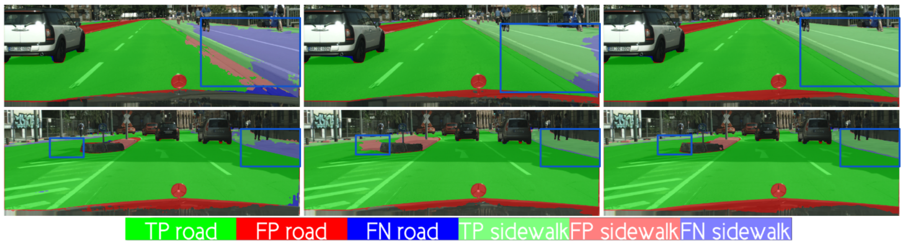


Figure 3.3: Qualitative results (GTAV + Synscapes → Cityscapes) focusing on TP/F-P/FN for road and sidewalk classes. Columns, left to right: SrcLAB, self-training stage, co-training loop (full co-training procedure). Blue boxes highlight areas of interest.

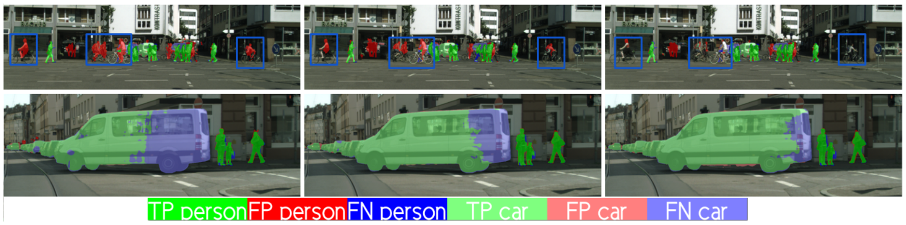


Figure 3.4: Analogous to Figure 3.3 for the classes Person and Car.

are observed in some classes preventing to reach upper-bound mIoU. In fact, upper-bounds are neither perfect, which is due to the difficulty of performing semantic segmentation in onboard images.

Figures 3.3 and 3.4 exemplify these comments by showing the pseudo-labeling evolution for several classes of special interests such as Road, Sidewalk, Pedestrian, and Car. In Figure 3.3, we see how the SrcLAB model has particular problems to segment well the sidewalk, however, the self-training stage resolves most errors although it may introduce new ones (mid-bottom image), while the co-training loop is able to recover from such errors. In Figure 3.4, we can see (bottom row) how the self-training stage improves the pseudo-labeling of a van, while the co-training loop improves it even more. Analogously, we can see (top row) how self-training helps to alleviate the confusion between pedestrian and riders, while the co-training loop almost removes all the confusion errors between these two classes.

Table 3.9: Content statistics for the considered datasets. For each class, we indicate the percentage (%) of: (1) Images containing samples of the class, and (2) Pixels in the dataset with the class label.

		Road	Sidewalk	Building	Wall	Fence	Pole	Traffic light	Traffic Sign	Vegetation	Terrain	Sky	Person	Rider	Car	Truck	Bus	Train	Motorbike	Bicycle
SYNTHIA	Images	99.9	99.9	99.9	41.7	79.9	99.9	68.1	94.3	97.3	0.0	93.9	99.9	99.5	95.8	0.0	76.4	0.0	84.1	99.5
	Pixels	18.54	19.31	29.43	0.27	0.26	1.04	0.04	0.10	10.15	0.0	6.80	4.25	0.47	4.04	0.0	1.53	0.0	0.21	0.22
GTAV	Images	99.3	97.3	99.8	95.7	84.2	99.2	74.3	60.5	99.3	97.8	99.2	93.9	13.1	91.7	67.0	20.4	4.5	16.7	1.8
	Pixels	32.1	8.29	16.91	1.85	0.63	1.06	0.13	0.08	7.6	2.14	13.53	0.36	0.03	2.51	1.13	0.37	0.06	0.03	0.1
Synscapes	Images	99.9	98.7	99.1	32.8	98.2	99.6	96.7	98.9	97.6	48.2	98.0	99.7	82.3	98.5	86.9	80.8	65.8	86.5	89.5
	Pixels	28.29	6.82	22.57	1.13	1.37	2.07	0.37	0.53	13.93	0.88	8.31	3.18	0.79	5.59	0.79	1.08	1.17	0.6	0.52
Cityscapes	Images	98.6	94.5	98.6	32.6	43.6	99.1	55.7	94.4	70.3	55.6	90.3	78.8	34.4	95.2	12.1	9.2	4.8	17.2	55.3
	Pixels	32.63	5.39	20.19	0.58	0.78	1.09	0.18	0.49	14.08	1.03	3.55	1.08	0.12	6.19	0.24	0.21	0.21	0.09	0.37
BDD100K	Images	96.5	66.7	88.4	15.4	30.6	95.0	47.1	75.3	91.7	36.7	94.8	34.7	5.2	97.3	30.5	15.0	0.7	3.8	6.4
	Pixels	21.26	2.03	13.24	0.48	1.03	0.94	0.18	0.34	13.2	1.03	17.26	0.25	0.02	8.13	0.97	0.56	0.01	0.02	0.05
Mapillary	Images	98.8	72.2	91.3	46.2	62.7	98.6	51.0	91.0	96.8	46.9	98.9	50.5	19.0	93.4	24.0	15.8	1.3	15.1	16.7
	Pixels	19.42	2.99	12.47	0.75	1.26	0.9	0.18	0.45	14.94	1.05	29.33	0.31	0.06	3.36	0.37	0.26	0.02	0.06	0.07

3.5 Supplementary work

3.5.1 Additional datasets information

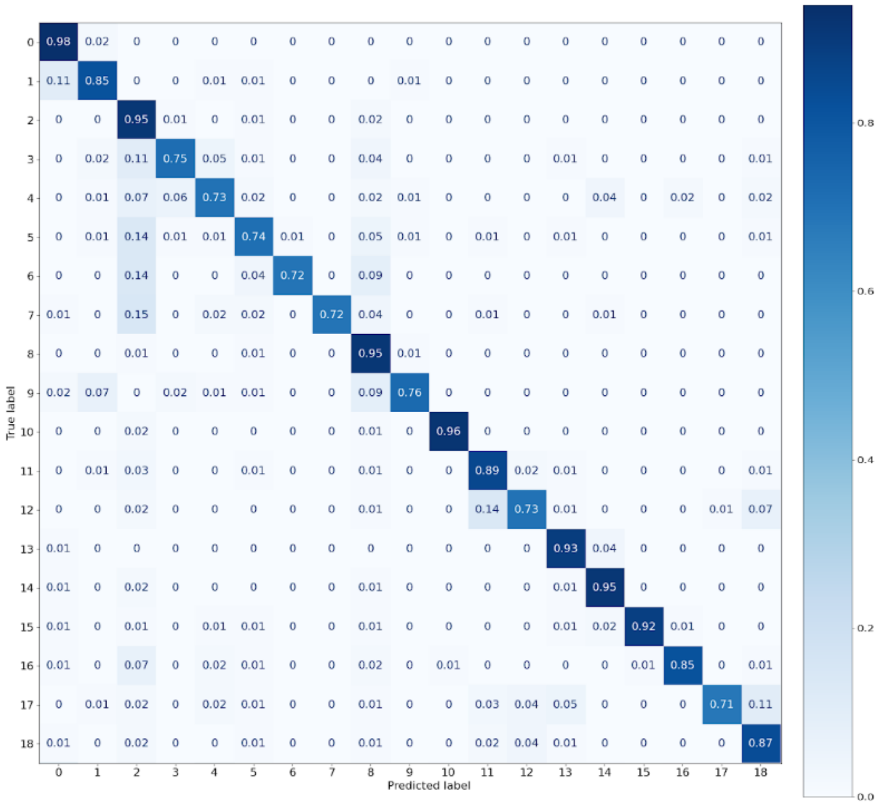
Table 3.9 presents statistics about the content of the different datasets. Focusing on the synthetic datasets, we observe that GTAV have few cases of Bicycle and Train, so explaining why the semantic segmentation baseline performs poorly on these classes. On other hand, SYNTHIA and Synscapes are overall well balanced, however, in Synscapes, the examples of Bus, Train, and Truck are very similar in terms of global shape. On other hand, SYNTHIA is less photo-realistic than Synscapes.

3.5.2 Additional co-training information

Figure 3.5 shows the confusion matrix of the co-training model trained with GTAV + Synscapes as source data and Cityscapes as target data. We can see how background classes like Road, Building, Vegetation and Sky reach a ~ 95% accuracy. All classes corresponding to dynamic objects show a ~ 90% accuracy, except for Motorbike and Rider with ~ 70%. Riders may be confused with pedestrians (Person class) and motorbikes with bicycles. This problem could be addressed injecting more rider samples in the source data, including corner cases where they appear with pedestrians around. Analogously, having more synthetic samples showing motorbikes and bicycles may help to better differentiate such classes.

Figure 3.6 shows the confusion matrix of the co-training model trained with GTAV + Synscapes as source data and BDD100K as target data. Again, most classes corresponding to environment (Road, Building, Vegetation and Sky) have a high accuracy. However, there are some environment classes with a large magin for improvement. For instance, the Sidewalk class tends to be labeled as Road, which we think is due to lacking real-world images with sidewalks; in Table 3.9, we can see that only the ~ 66% of training images show sidewalks, while this statistics reaches the ~ 98% in the case of Cityscapes. Wall, Fence, and Pole classes form a sort of local cluster of confusion. Sometimes, pixels of these classes are also labeled as Vegetation because instances of this class use to occlude instances of Wall/Fence/Pole or vice versa. Traffic lights and signs are frequently labeled as Building/Pole/Vegetation. In here, a different labeling policy may also be introducing confusion on the trained model. While the rear part of traffic lights and signs is not labeled in Synscapes and Cityscapes, they are in BDD100K. Other classes such as Truck, Motorbike and Bike, tend to be labeled as Car. The Truck class is also under a discrepancy in labeling policy, since pick-up cars are labeled as Truck in BDD100K but as Car in the others datasets.

Figure 3.7 shows the confusion matrix of the co-training model trained with GTAV + Synscapes as source data and Mapillary Vistas as target data. The diagonal scores and confusion cases are similar to those of Cityscapes. Focusing on difficult specifics, we can observe cases of high accuracy but mid IoU. For instance, the class Terrain shows ~ 93% and ~ 47%, respectively. Other classes showing a similar pattern are Truck, Bus, and Motorbike. We think this can be at least partially due to differences in labeling policies. Mapillary Vistas accounts for around one hundred different classes, which cannot be easily mapped to the 19 classes of Cityscapes. Then, using only 19 classes while mapping as unlabeled the rest, leaves a ground truth with less information per training image than in cases like Cityscapes and BDD100K.



True \ Predicted	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
0	0.98	0.02	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0.11	0.85	0	0	0.01	0.01	0	0	0	0.01	0	0	0	0	0	0	0	0	0
2	0	0	0.95	0.01	0	0.01	0	0	0.02	0	0	0	0	0	0	0	0	0	0
3	0	0.02	0.11	0.75	0.05	0.01	0	0	0.04	0	0	0	0	0.01	0	0	0	0	0.01
4	0	0.01	0.07	0.06	0.73	0.02	0	0	0.02	0.01	0	0	0	0	0.04	0	0.02	0	0.02
5	0	0.01	0.14	0.01	0.01	0.74	0.01	0	0.05	0.01	0	0.01	0	0.01	0	0	0	0	0.01
6	0	0	0.14	0	0	0.04	0.72	0	0.09	0	0	0	0	0	0	0	0	0	0
7	0.01	0	0.15	0	0.02	0.02	0	0.72	0.04	0	0	0.01	0	0	0.01	0	0	0	0
8	0	0	0.01	0	0	0.01	0	0	0.95	0.01	0	0	0	0	0	0	0	0	0
9	0.02	0.07	0	0.02	0.01	0.01	0	0	0.09	0.76	0	0	0	0	0	0	0	0	0
10	0	0	0.02	0	0	0	0	0	0.01	0	0.96	0	0	0	0	0	0	0	0
11	0	0.01	0.03	0	0	0.01	0	0	0.01	0	0	0.89	0.02	0.01	0	0	0	0	0.01
12	0	0	0.02	0	0	0	0	0	0.01	0	0	0.14	0.73	0.01	0	0	0	0.01	0.07
13	0.01	0	0	0	0	0	0	0	0	0	0	0	0	0.93	0.04	0	0	0	0
14	0.01	0	0.02	0	0	0	0	0	0.01	0	0	0	0	0.01	0.95	0	0	0	0
15	0.01	0	0.01	0	0.01	0.01	0	0	0.01	0	0	0	0	0.01	0.02	0.92	0.01	0	0
16	0.01	0	0.07	0	0.02	0.01	0	0	0.02	0	0.01	0	0	0	0	0.01	0.85	0	0.01
17	0	0.01	0.02	0	0.02	0.01	0	0	0.01	0	0	0.03	0.04	0.05	0	0	0	0.71	0.11
18	0.01	0	0.02	0	0	0.01	0	0	0.01	0	0	0.02	0.04	0.01	0	0	0	0	0.87

0	1	2	3	4	5	6	7	8	9
Road	Sidewalk	Building	Wall	Fence	Pole	Traffic light	Traffic sign	Vegetation	Terrain

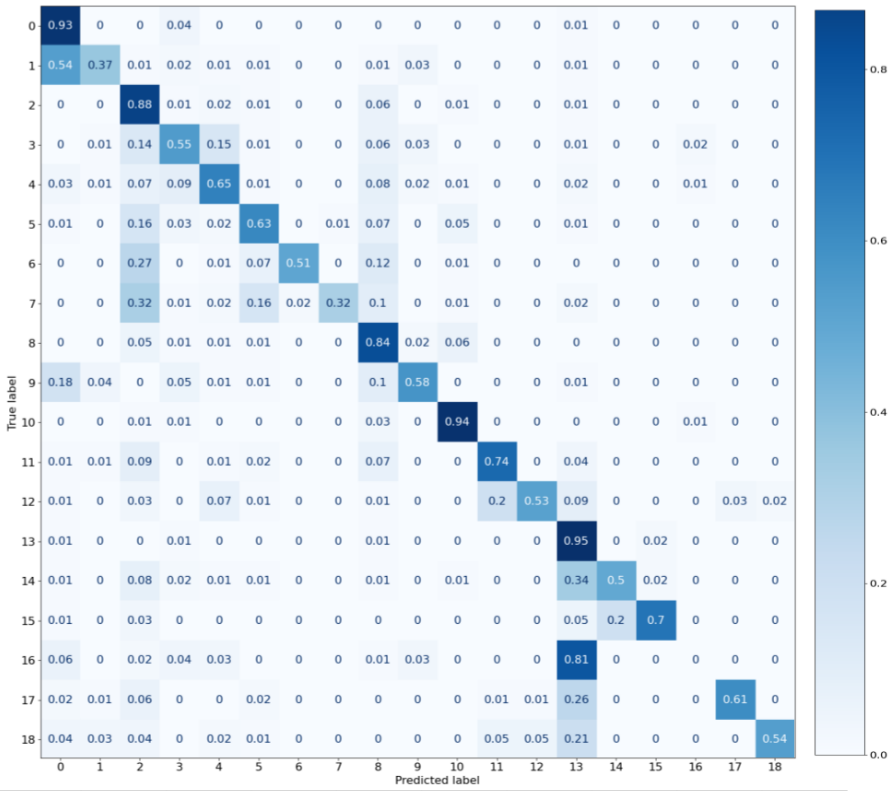
10	11	12	13	14	15	16	17	18
Sky	Person	Rider	Car	Truck	Bus	Train	Motorcycle	Bicycle

Figure 3.5: Confusion matrix of the co-training model trained with GTAV + Synscapes as source data and Cityscapes as target data.

3.5.3 Additional qualitative analysis

GTAV + Synscapes → Cityscapes: In Figures 3.8 and 3.9 we show additional qualitative results obtained on Cityscapes, when using GTAV+Synscapes as source domain.

Figure 3.8 remarks where several improvements from the co-training model vs. self-training one appear. In the example at the left column, we see that the baseline and self-training models have problems labeling a bus while co-training one labels



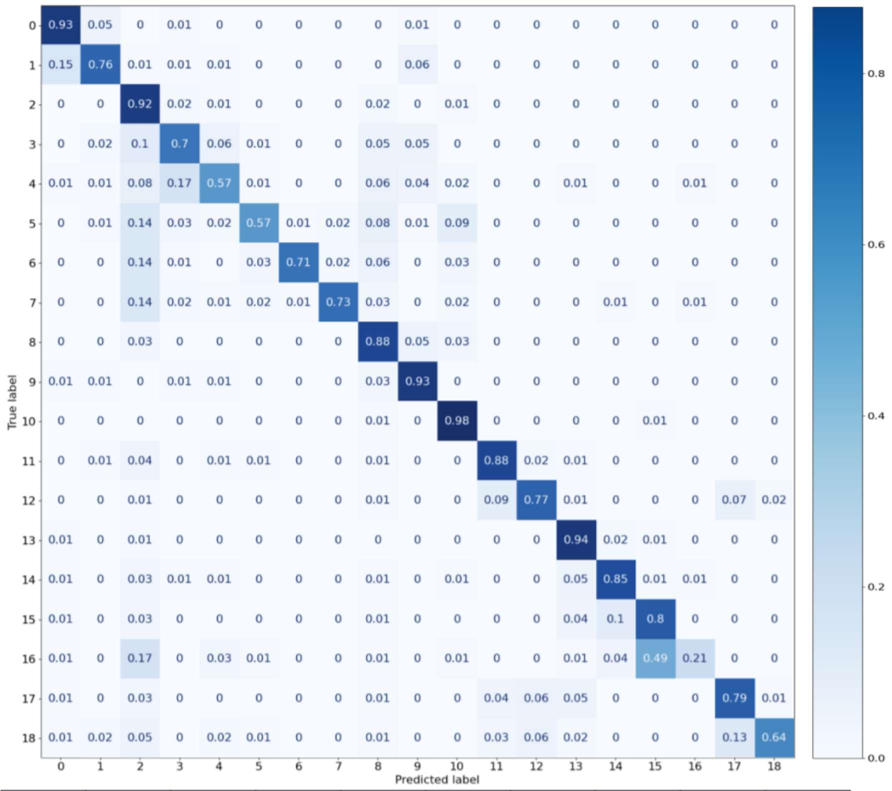
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
0	0.93	0	0	0.04	0	0	0	0	0	0	0	0	0	0.01	0	0	0	0	0
1	0.54	0.37	0.01	0.02	0.01	0.01	0	0	0.01	0.03	0	0	0	0.01	0	0	0	0	0
2	0	0	0.88	0.01	0.02	0.01	0	0	0.06	0	0.01	0	0	0.01	0	0	0	0	0
3	0	0.01	0.14	0.55	0.15	0.01	0	0	0.06	0.03	0	0	0	0.01	0	0	0.02	0	0
4	0.03	0.01	0.07	0.09	0.65	0.01	0	0	0.08	0.02	0.01	0	0	0.02	0	0	0.01	0	0
5	0.01	0	0.16	0.03	0.02	0.63	0	0.01	0.07	0	0.05	0	0	0.01	0	0	0	0	0
6	0	0	0.27	0	0.01	0.07	0.51	0	0.12	0	0.01	0	0	0	0	0	0	0	0
7	0	0	0.32	0.01	0.02	0.16	0.02	0.32	0.1	0	0.01	0	0	0.02	0	0	0	0	0
8	0	0	0.05	0.01	0.01	0.01	0	0	0.84	0.02	0.06	0	0	0	0	0	0	0	0
9	0.18	0.04	0	0.05	0.01	0.01	0	0	0.1	0.58	0	0	0	0.01	0	0	0	0	0
10	0	0	0.01	0.01	0	0	0	0	0.03	0	0.94	0	0	0	0	0	0.01	0	0
11	0.01	0.01	0.09	0	0.01	0.02	0	0	0.07	0	0	0.74	0	0.04	0	0	0	0	0
12	0.01	0	0.03	0	0.07	0.01	0	0	0.01	0	0	0.2	0.53	0.09	0	0	0	0.03	0.02
13	0.01	0	0	0.01	0	0	0	0	0.01	0	0	0	0	0.95	0	0.02	0	0	0
14	0.01	0	0.08	0.02	0.01	0.01	0	0	0.01	0	0.01	0	0	0.34	0.5	0.02	0	0	0
15	0.01	0	0.03	0	0	0	0	0	0	0	0	0	0	0.05	0.2	0.7	0	0	0
16	0.06	0	0.02	0.04	0.03	0	0	0	0.01	0.03	0	0	0	0.81	0	0	0	0	0
17	0.02	0.01	0.06	0	0	0.02	0	0	0	0	0	0.01	0.01	0.26	0	0	0	0.61	0
18	0.04	0.03	0.04	0	0.02	0.01	0	0	0	0	0	0.05	0.05	0.21	0	0	0	0	0.54

0	1	2	3	4	5	6	7	8	9
Road	Sidewalk	Building	Wall	Fence	Pole	Traffic light	Traffic sign	Vegetation	Terrain

10	11	12	13	14	15	16	17	18
Sky	Person	Rider	Car	Truck	Bus	Train	Motorcycle	Bicycle

Figure 3.6: Confusion matrix of the co-training model trained with GTAV + Synscapes as source data and BDD100K as target data.

all the dynamic objects accurately as the upper-bound model does. In the mid column, the co-training model improves the labeling of the sidewalk, the closest person and rider. Last column shows a challenging case where several pedestrians mixed with cyclists are crossing the road. The baseline and self-training models are poor at distinguishing riders from pedestrians. The co-training model is able to improve on classifying riders over the baseline and self-training models, but not reaching the performance of the upper-bound model in this case.



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
0	0.93	0.05	0	0.01	0	0	0	0	0	0.01	0	0	0	0	0	0	0	0	0
1	0.15	0.76	0.01	0.01	0.01	0	0	0	0	0.06	0	0	0	0	0	0	0	0	0
2	0	0	0.92	0.02	0.01	0	0	0	0.02	0	0.01	0	0	0	0	0	0	0	0
3	0	0.02	0.1	0.7	0.06	0.01	0	0	0.05	0.05	0	0	0	0	0	0	0	0	0
4	0.01	0.01	0.08	0.17	0.57	0.01	0	0	0.06	0.04	0.02	0	0	0.01	0	0	0.01	0	0
5	0	0.01	0.14	0.03	0.02	0.57	0.01	0.02	0.08	0.01	0.09	0	0	0	0	0	0	0	0
6	0	0	0.14	0.01	0	0.03	0.71	0.02	0.06	0	0.03	0	0	0	0	0	0	0	0
7	0	0	0.14	0.02	0.01	0.02	0.01	0.73	0.03	0	0.02	0	0	0	0.01	0	0.01	0	0
8	0	0	0.03	0	0	0	0	0	0.88	0.05	0.03	0	0	0	0	0	0	0	0
9	0.01	0.01	0	0.01	0.01	0	0	0	0.03	0.93	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0.01	0	0.98	0	0	0	0	0.01	0	0	0
11	0	0.01	0.04	0	0.01	0.01	0	0	0.01	0	0	0.88	0.02	0.01	0	0	0	0	0
12	0	0	0.01	0	0	0	0	0	0.01	0	0	0.09	0.77	0.01	0	0	0	0.07	0.02
13	0.01	0	0.01	0	0	0	0	0	0	0	0	0	0	0.94	0.02	0.01	0	0	0
14	0.01	0	0.03	0.01	0.01	0	0	0	0.01	0	0.01	0	0	0.05	0.85	0.01	0.01	0	0
15	0.01	0	0.03	0	0	0	0	0	0.01	0	0	0	0	0.04	0.1	0.8	0	0	0
16	0.01	0	0.17	0	0.03	0.01	0	0	0.01	0	0.01	0	0	0.01	0.04	0.49	0.21	0	0
17	0.01	0	0.03	0	0	0	0	0	0.01	0	0	0.04	0.06	0.05	0	0	0	0.79	0.01
18	0.01	0.02	0.05	0	0.02	0.01	0	0	0.01	0	0	0.03	0.06	0.02	0	0	0	0.13	0.64

0	1	2	3	4	5	6	7	8	9
Road	Sidewalk	Building	Wall	Fence	Pole	Traffic light	Traffic sign	Vegetation	Terrain

10	11	12	13	14	15	16	17	18
Sky	Person	Rider	Car	Truck	Bus	Train	Motorcycle	Bicycle

Figure 3.7: Confusion matrix of the co-training model trained with GTAV + Synscapes as source data and Mapillary Vistas as target data.

Figure 3.9 shows examples of wrong labeling even from the co-training model. The left column shows a building erroneously labeled by all models except by the upper-bound one. A large area of the building is labeled as fence. Which we believe is due to the reflections seen in the facade windows. Furthermore, the variability of buildings in the synthetic data are not enough to cover these variants seen in the real-world scenarios. The upper-bound model properly labels most of the building, but still labels part of its bottom as fence. The mid column shows an usual troublesome case in Cityscapes, where a stone-based road is labeled as sidewalk. Note that stones are also used to build sidewalks. The right column shows a bus with some kind of advertising in the back, which induces all the models (including the upper-bound one) to label the bus as a mixture of Traffic Sign and Building. We note that, overall, there are not sufficient training samples of this type.

GTAV + Synscapes → BDD100K: In Figures 3.10 and 3.11 we present qualitative results changing the target to BDD100K.

Figure 3.10 shows how noisy is the baseline model in this case. This is due to variability regarding weather conditions, lighting, and onboard cameras. Note that, contrarily to the case of Cityscapes, BDD100K cameras are not even installed in the same position from car to car, not even in the same car model in all the cases (see Figure 3.12). In the left column of Figure 3.10, we see how both self-training and co-training models clearly perform much better than the baseline, in fact, similarly to the upper-bound one. In the mid column, co-training model is the one performing most similarly to the upper-bound. In the last column, self-taining, co-training, and upper-bound models have problems labeling the bus cabin, which is confused with a truck cabin (upper-bound), a car cabin (co-training), and a bit of both (self-training as coming from the baseline).

Figure 3.11 shows examples of wrong labeling even from the co-training model. The left column shows an example where a far bus is confused with a Truck (baseline) or a Car (self-training and co-training), and the sidewalk is largely confused with Terrain/Road (baseline and co-training), Road (self-training), even the upper-bound confuses part of the sidewalk with Terrain. Traffic lights are also miss-classified. The mid column also shows failures on sidewalk classification for self-training and co-training models, although both label the road better than the baseline. These models label the closest car even better than the upper-bound model. The right column shows a extreme case where all models have difficulties labeling a case of rider-with-bicycle. The baseline model provides some insufficient cues, self-training one improves them, but co-training model labels the rider-with-bicycle partially as Car and partially as Road. Even the upper-bound model misses the rider, only properly labeling the bicycle.

GTAV + Synscapes → Mapillary: In Figures 3.13 and 3.14 we introduce qualitative results changing the target to Mapillary Vistas.

Figure 3.13 aims to show cases where the co-training model performs similarly or even better than the upper-bound one. In the left column, the co-training model performs better than the rest of models by avoiding parts of the wall being labeled as Fence. In the mid column, only the co-training and the upper-bound models perform relatively well labeling the sidewalk, being the co-training model even better. In the right column, only the co-training model is properly labeling the wall and relatively well the close truck. Note how the upper-bound model labels the wall as Fence.

Figure 3.14 shows examples of wrong labeling from the co-training model. In the left column, some buildings are labeled as Train, in fact, these building have a shape and are arranged in a way that resemble train wagons. In the mid column, we see that the co-training model is labeling some vegetation as Terrain, performing a bit worse than the self-training model. In the right column, the co-training model partially labels two buses as Car, while the self-training model performs a better labeling in these cases.

3.6 Conclusions

In this chapter, we have addressed the training of semantic segmentation models under the challenging setting of synth-to-real unsupervised domain adaptation (UDA), i.e., assuming access to a set of synthetic images (source) with automatically generated ground truth together with a set of unlabeled real-world images (target). We have proposed a new co-training procedure combining a self-training stage and a co-training loop where two models arising from the self-training stage collaborate for mutual improvement. The overall procedure treats the deep models as black boxes and drives their collaboration at the level of pseudo-labeled target images, i.e., neither modifying loss functions is required, nor explicit feature alignment. We have tested our proposal on standard synthetic (GTAV, Synscapes, SYNTHIA) and real-world datasets (Cityscapes, BDD100K, Mapillary Vistas). Our co-training shows improvements ranging from ~13 to ~31 mIoU points over baselines, remaining very closely (less than 10 points) to the upper-bounds. In fact, up to the best of our knowledge, we are the first reporting such kind of results for challenging target domains such as BDD100K and Mapillary Vistas. Moreover, we have shown how the different components of our co-training procedure contribute to improve final mIoU. Future work, will explore collaboration from additional perception models at the co-training loop, i.e., not necessarily based on semantic segmentation but such collaborations may arise from object detection or monocular depth estimation.

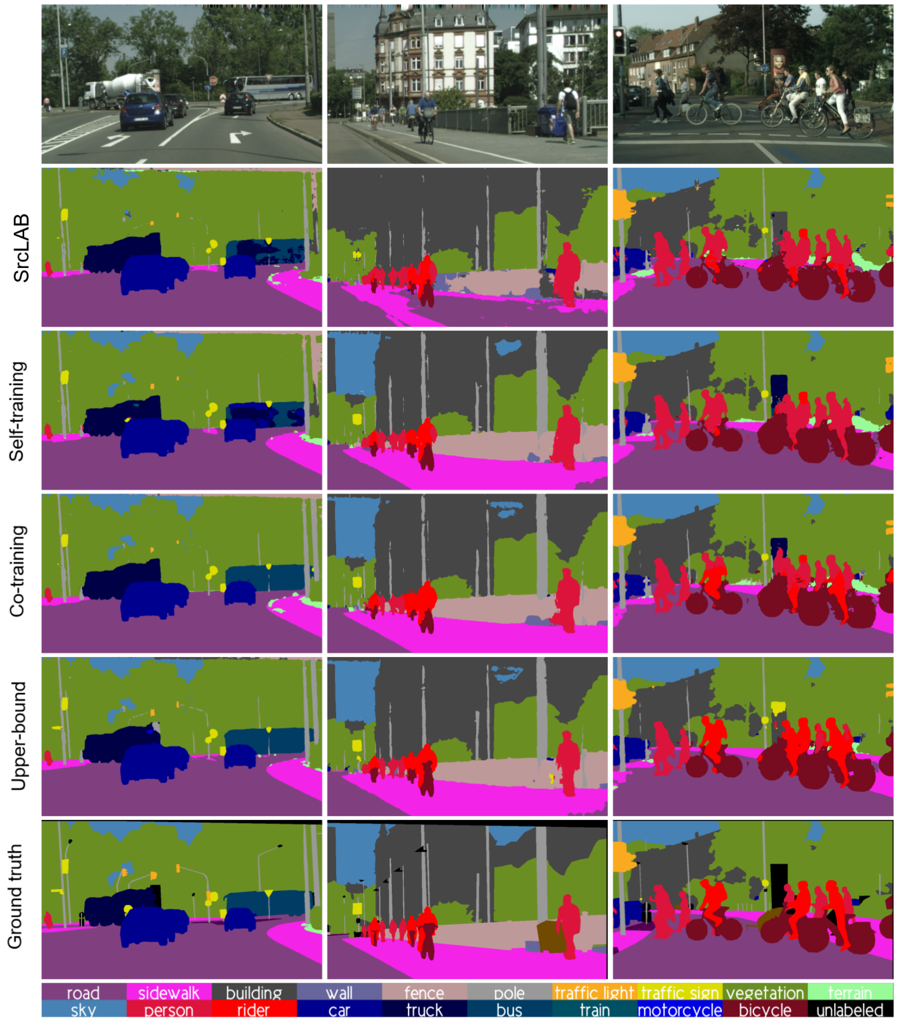


Figure 3.8: Qualitative results on the validation set of Cityscapes, when relying on GTAV + Synscapes as source data. The baseline model is trained using only these source data, the upper-bound model uses these source data and all the labeled training data of Cityscapes. Self-training and co-training models rely on the source data and the same training data from Cityscapes but without the labeling information.

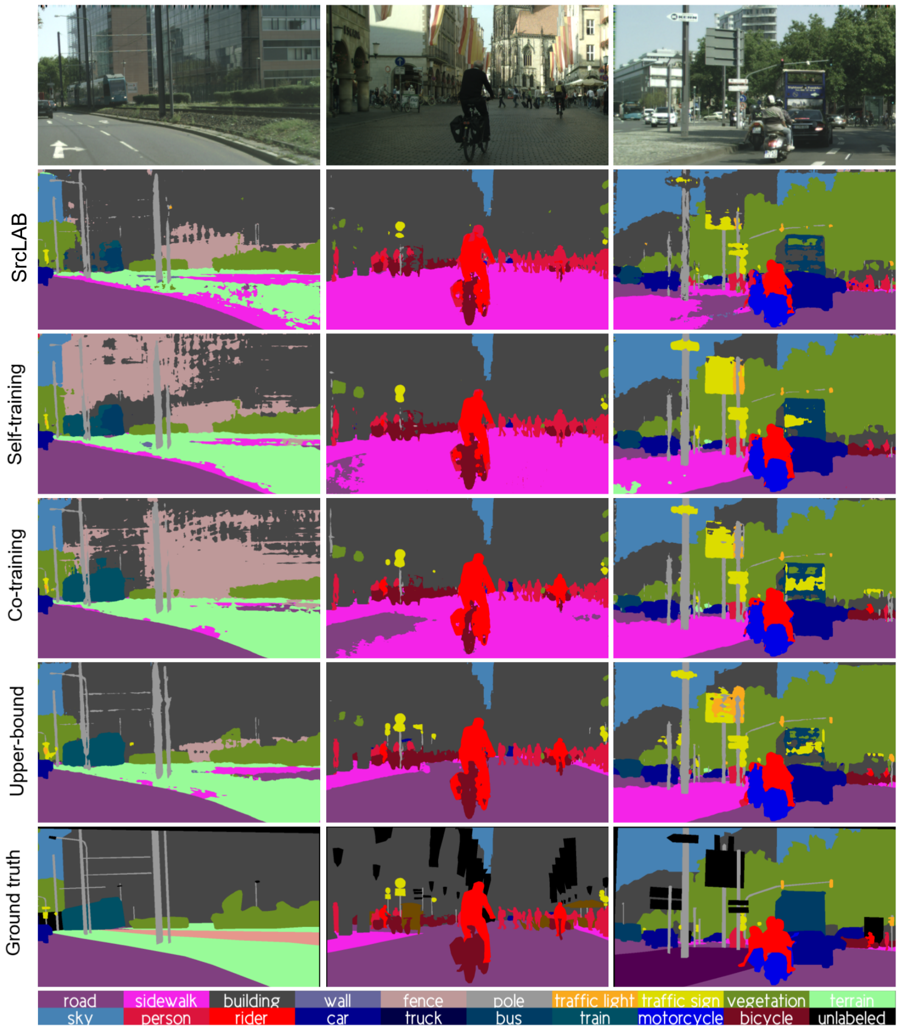


Figure 3.9: Analogous to Figure 3.8, focusing on problematic examples.

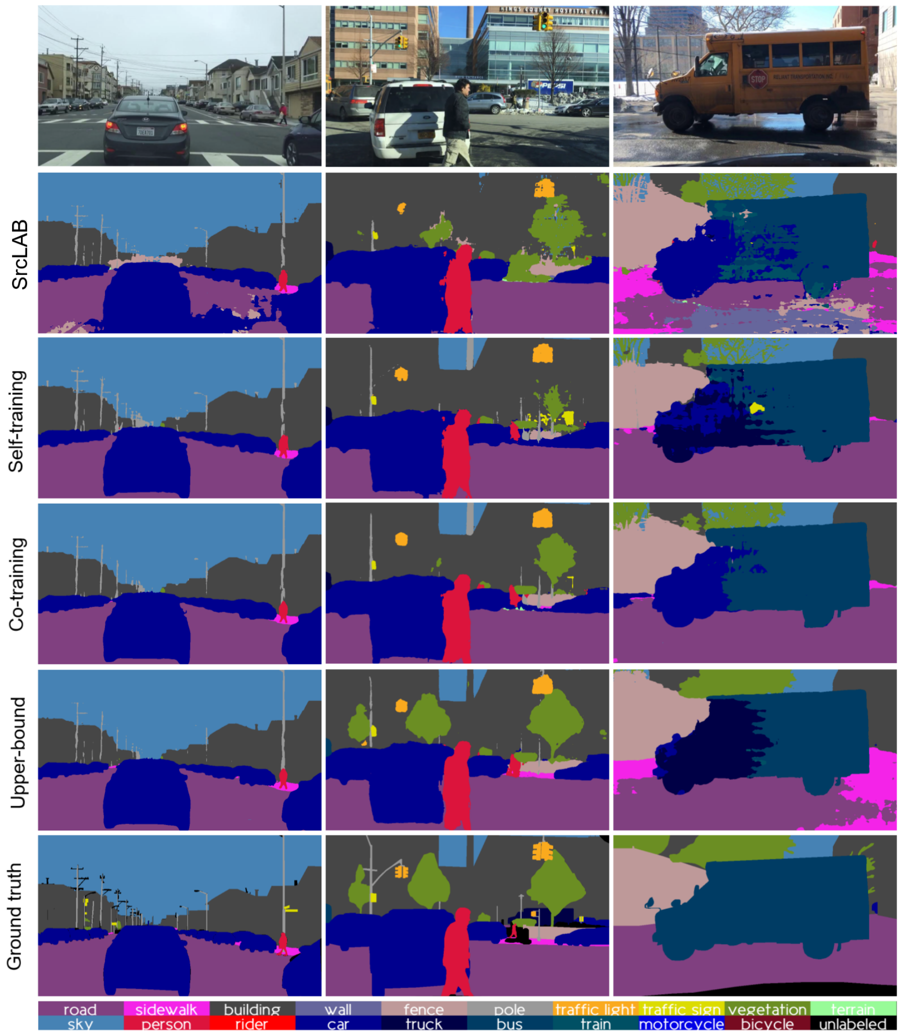


Figure 3.10: Qualitative results similar to Figure 3.8 when using BDD100K as target.

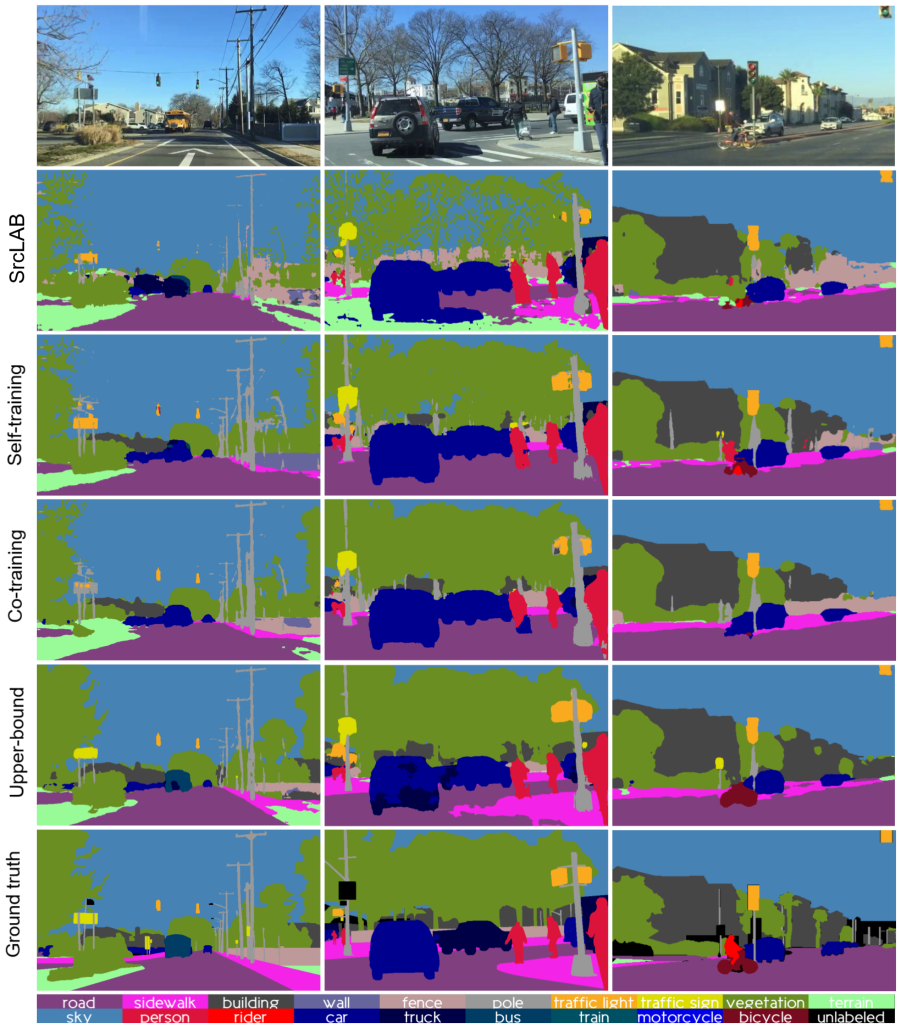


Figure 3.11: Analogous to Figure 3.10, focusing on problematic examples.

Figure 3.12: Image samples from BDD100K dataset.

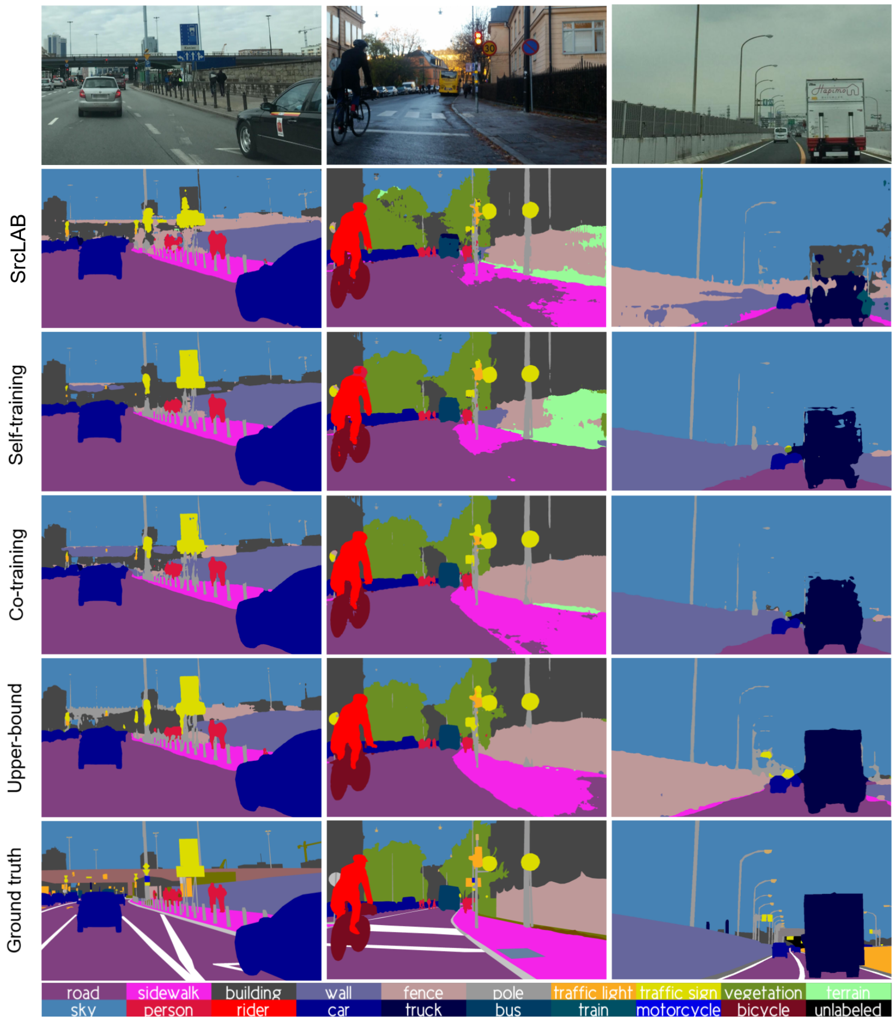


Figure 3.13: Qualitative results similar to Figure 3.8 when using Mapillary Vistas as target.

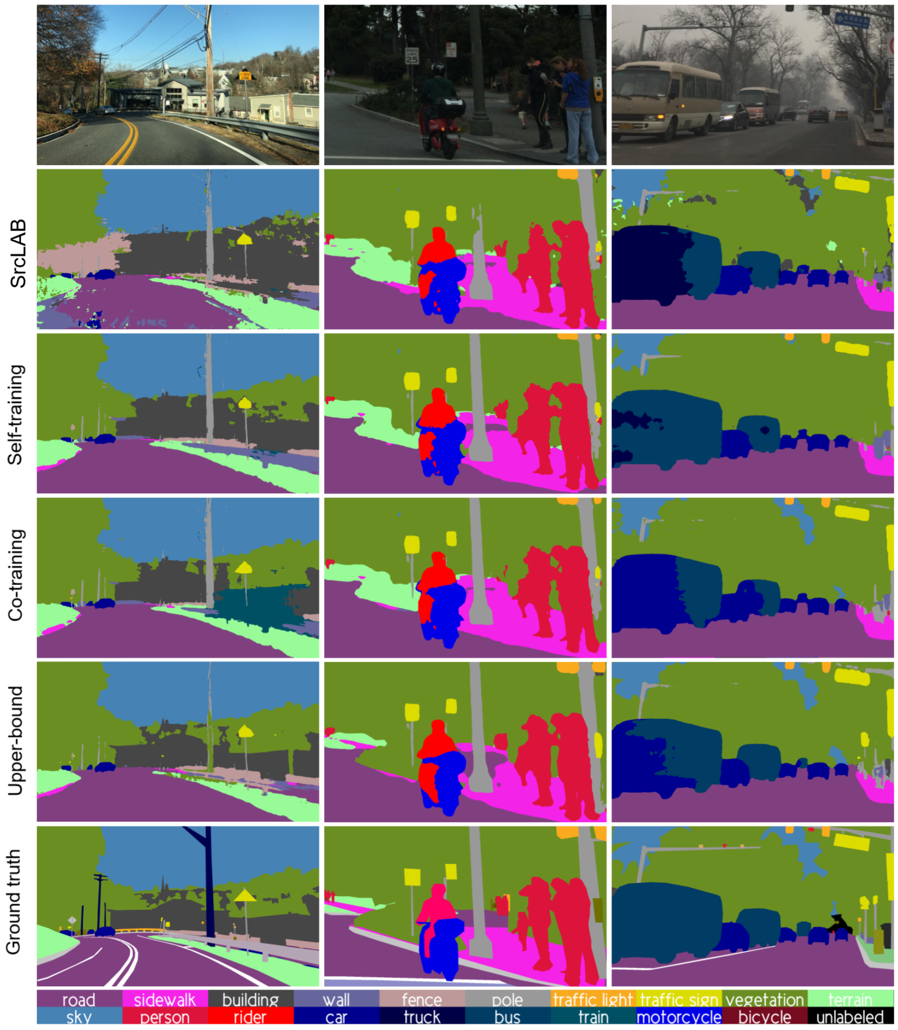


Figure 3.14: Analogous to Figure 3.13, focusing on problematic examples.

4 Human-inspired semantic pseudo-labeling with synthetic images generated by path-tracing

We face an unsupervised domain adaptation (UDA) setting for semantic segmentation. Our contribution is two-fold. On the one hand, we propose a new SSL framework inspired by human labelers. In particular, semantic classes are categorized. Then, self-training is applied per category. In this way, each image has different per-category layers of pseudo-labels produced by corresponding self-trained models. Then, we fuse the pseudo-labels by an ordered composition. We denote this SSL technique as *ordered composition of specialized models* (OCSM). As human labelers usually do, we order (stack) the layers of pseudo-labels according to the prior area size of the semantic categories, so that pseudo-labels from higher layers prevail in case of labeling conflict in a pixel. After applying this fusion of layers, we obtain fully labeled images. On the other hand, we have also contributed to the generation of a new synthetic dataset for supporting the automatic semantic labeling of onboard images. It is photo-realistic since it is based on path-tracing rendering, proper 3D assets in terms of geometry and materials, and realistic illumination environments. OCSM seamlessly leverages publicly available synthetic datasets as well as this new one to obtain state-of-the-art results on synth-to-real UDA for semantic segmentation. We show that the new dataset allows us to reach better labeling accuracy than previously existing datasets, while it complements them well when combined. Moreover, we also show that OCSM outperforms co-training. In the Cityscapes validation set, we reach mIoU=73.8, only 6 points behind the upper bound based on human labeling.

4.1 Introduction

Real-world data are difficult to label, thus, having real-world data supervised for particular tasks is time-consuming and expensive. Synthetic data are presented as an alternative to simulate scenarios related to these particular tasks. However, recreating the same properties of the real data with synthetic ones is a challenging procedure difficult to accomplish, propitiating a critical domain shift between synthetic and real domains. On the one hand, the advances in computer graphics allow us to create more photo-realistic scenarios [85, 108, 119]. On the other hand, domain adaptation (DA) methods aim at reducing the domain shift [67, 90, 113] and make affordable systems to properly perform across domains. Furthermore, unsupervised domain adaptation (UDA) methods rely on the knowledge obtained from labeled source data (synthetic data in our case) to reduce the domain shift using the real-world data without knowledge of it (unlabeled target) [66, 93, 102]. Thus, UDA methods are of great interest for leveraging synthetic data. More in detail, UDA self-training frameworks for semantic segmentation [27, 60, 103, 146, 147] rely on iteratively generating good pseudo-labels of the target data to include them on the training procedure. Obtaining robust initial pseudo-labels from models trained on synthetic data is challenging due to the domain shift, hence good policies to select and filter pseudo-labels are needed.

In this chapter, we address synth-to-real UDA for onboard semantic segmentation focusing on two different aspects to improve the generated pseudo-labels. On the one hand, we aim at improving our framework for automatic labeling. More specifically, we propose a human-inspired labeling procedure where we use several specialized models to compose a fully pseudo-labeled image. Each of these models is trained on categorized semantic classes, similarly to how humans label semantically an RGB image using AI tools [1, 89] (first background, then large objects, and finally small ones). On the other hand, in collaboration with a team of experts, we are developing a new photo-realistic synthetic dataset for urban scenarios similar to [85, 86, 119]. Our aim with this new dataset is at improving our automatically generate semantic labels.

This work leverages the knowledge and framework from co-training (see section 3.3) for semantic segmentation. The results obtained with the co-training method were satisfactory, achieving state-of-the-art in each of the synthetic datasets used as a source. However, the clear difference between single-source and multi-source scenarios opened new doors to explore. In fact, combining two free and available datasets such as GTAV [85] and Synscapes [119] to train a baseline model, outperformed a considerable amount of related UDA techniques. Thus, our new synthetic dataset proposal uses path-tracing rendering, high-quality 3D assets in terms of geometry and materials, and realistic illumination environments, so that we obtain images of a large visual fidelity. We defined two urban scenes by adding setups uncommon in publicly available synthetic datasets, which are populated by a rich set of 3D assets for each class involved in Cityscapes dataset. A model trained with our new dataset alone surpasses equivalent models trained on GTAV and Synscapes individually, in particular, by a margin of ~7 mean intersection-over-union (mIoU) on Cityscapes validation set. Furthermore, in combination with them pushes the multi-source baseline score to ~66 mIoU points.

Our next natural step was to apply the co-training method to see how it performed with the addition of our new synthetic data. While the self-training step with GTAV + Synscapes + Our data, reached ~71 mIoU in Cityscapes validation, the co-training step could not improve to the same degree (only ~2 mIoU points over self-training). We believe that we reached the maximum capacity of co-training. Hence, in this work, we propose a new methodology to obtain pseudo-labels, which we call *ordered composition of specialized models* (OCSM). It leverages the self-training stage from co-training. Specialized models are trained on a predefined set of categorized semantic classes, using synthetic data. Thus, we divide the 19 official classes defined in Cityscapes into several categories, e.g., category Vehicles: Car, Truck, Bus, and Train. Each of these specialized models is refined by self-training. Finally, we obtain fully pseudo-labeled images by an ensemble-like method inspired by human labeling. In particular, we stack the pseudo-labels from each specialized model ordered by the usual area size of the categories (from larger categories to smaller ones). With this simple yet effective combination of category-specialized pseudo-labels and the available synthetic data, we train a final model which achieves a mIoU of ~74 on Cityscapes validation set, only ~6 mIoU points below the upper bound. Up to the best of our knowledge, we obtain the best result reported until now on synth-to-real UDA. In addition, we improve the co-training results for the other real-world datasets evaluated, BDD100K and Mapillary Vistas.

Section 4.2 contextualizes our work. Section 4.3 explains the proposed pseudo-labeling method. Section 4.4 details the dataset generation process. Section 4.5 describes the experimental setup and discusses the obtained results. Finally, Section 4.6 summarizes this work.

4.2 Related works

As we have mentioned previously, our work proposes a new synthetic dataset and a novel UDA procedure. Thus, we divide this section accordingly to review the works related to each topic.

4.2.1 Synth-to-real UDA

Section 3.2 summarizes works on synth-to-real UDA at the time we designed our co-training method. Here, we review more recent related works. Hoyer et al. [41] successfully use the novel vision-transformers [20] to solve UDA semantic segmentation in a teacher-student self-training framework with additional training strategies as computing ImageNet feature distances among others. Furthermore, several new works propose techniques on top of the already existing frameworks (teacher-student, self-training, etc.) to improve further performance. Zhang et al. [136] introduce a spectrum transformer that mitigates inter-domain discrepancies and multi-view spectral learning to learn useful representations. Huang et al. [44] propose a Category Contrast technique (CaCo) that uses semantics-aware dictionaries to apply category contrastive learning for domain-invariant class representations. Hoyer et al. [42] define a multi-resolution training approach combining feature information from small high-resolution image crops and large low-resolution ones to capture context dependencies. We consider our method complementary to other techniques too, where we apply it on top of a self-training stage, but it could be applied to the aforementioned frameworks.

Ensembling models is a powerful tool where we combine multiple model outputs in a single prediction. Classic approaches include bagging [7], boosting [91] and AdaBoost [24] are applied to today's CNN methods. One of the first works on CNNs for semantic segmentation using ensembles was Marmanis et al. [68] modifying an FCN network to improve the deconvolution step and then train multiples instances to obtain an ensemble averaging the predictions. However, the high complexity and computational of ensembling models were drawbacks to be addressed. Several recent works, try to address computational issues sharing part of the networks between models. Bousselham et al. [6] propose a self-ensemble approach using the multi-scale features produced by a spatial pyramid network to feed different decoders and compose an ensemble by different strategies (averaging, majority vote, and hierarchical attention). In a similar fashion, Cao et al. [8] use different semantic heads sharing the same backbone to compute an ensemble using cooperative learning. Khirodkar et al. [53] propose a sequential ensembling method where models are trained in cascade and each model output is fed to the next one. These approaches have less impact on memory and complexity. Our new proposal falls inside the ensemble-of-models techniques due to the aforementioned pseudo-label composition strategy using specialized models by categories. In our case, we combine the semantic predictions computed off-line to generate the full pseudo-labeled image, thus we reduce the computational complexity to only train several semantic segmentation models (one per category) simultaneously.

In addition, ensemble models are applied to solve UDA tasks like our method. Extensive works were addressed on image classification [2, 49, 71, 78, 120, 142]. However, our interest falls in synth-to-real UDA for semantic segmentation where the available works are scarcer. Piva et al. [74] propose a framework similar to [60] where an image translation module feeds 3 different semantic networks and an ensemble layer aggregate the information to generate pseudo-labels, then the process continues with a self-training step. Yi et al. [129] train several GANs in parallel with different discrepancy and segmentation loss functions under different upsampling strategies. Finally, our pseudo-labeling procedure is motivated by the work by Chao et al. [9], where they propose an end-to-end ensemble-distillation framework for UDA. They compare several ensemble approaches and propose a pixel-wise fusion and channel-wise fusion policy to generate the final pseudo-labels from different CNN models. In our case, our ensemble fusion is simplified to a stacked combination ordered by category, and in addition to our synthetic dataset, we generate better pseudo-labels than using other ensemble methodologies.

4.2.2 Synthetic datasets

We summarize the work related to synthetic dataset generation on outdoor environments and autonomous driving. One of the first synthetic environments popularized to train AI agents was TORCS [122], an open racing car simulator game released in the late 90s that became later an important tool to train AI models for autonomous driving. A more modern approach to training AI agents was introduced by Dosovitskiy et al. [19]. They present one of the most relevant synthetic environments for end-to-end driving named CARLA, an open-source simulator for urban driving on Unreal Engine 4. CARLA provides an API with all the tools needed to record video scenes with different kinds of vehicle sensors for autonomous driving. [4, 46, 52, 92] are examples of works that generate their own data from CARLA.

Our dataset is inspired by real-world cities, similar works are, Gaidon et al. [25] presented Virtual KITTI, a small photo-realistic synthetic video dataset generated using the Unity3D platform. This dataset was created simulating some scenarios of the real dataset KITTI [28], with the objective to perform object detection and object tracking evaluation. In the same line, Wrenninge et al. [119] proposed an even better photo-realistic dataset called Synscapes, composed by 25K images designed procedurally to match the structure and contents of the Cityscapes dataset [15]. Ros et al. [86] created SYNTHIA using Unity3D, composed of 213K images for semantic segmentation with a wide variety of scenes and environmental conditions simulating New York City. In addition, Hernandez et al. [39] expanded SYNTHIA recreating scenarios from San Francisco.

Other works leverage video game engines to create new datasets. Johnson et al. [48] were the first to use Grand Theft Auto V (GTA V) video game to collect high amounts of data (200K) to train object detection models. In addition, Richter et al. [84, 85] proposed GTAV and VIPER datasets obtained also from this game, with 25K and 250K images respectively, with fully labeled semantic segmentation. Moreover, VIPER had optical flow, instance segmentation, 3D scene layout, and visual odometry information. Hurl et al. [45] also used GTA V and added a precise LIDAR simulator to obtain synthetic point cloud representations in their PreSIL dataset for 3D object detection. Saleh et al. [88] proposed VEIS created using Unity3D, a virtual environment for instance segmentation with additional foreground classes common in Cityscapes. Li et al. [58] created a synthetic dataset, using also Unity3D, of foggy urban scenes focused on roads.

In addition, our work uses OpenStreetMaps to obtain real-world layouts. A similar dataset created this way is from Tian et al. [101], who developed a multipurpose framework to generate synthetic data known as ParallelEye. ParallelEye leverages the 3D city batches generation capabilities from Esri CityEngine [43] to generate data from the OpenStreetMaps platform. Furthermore, Khan et al. [52] proposed a pipeline that models real-world urban environments and in combination with CARLA adds photo-realism and other elements to solve semantic segmentation. Finally, Li et al. [59] proposed the AADS framework that augments real-world pictures with simulated traffic flow to solve diverse AD tasks. They gather real-world data with LIDAR and cameras of streets to recreate the traffic flows.

Our dataset aims to contribute to onboard semantic segmentation. The most popular datasets in this task, SYNTHIA, GTAV, and Synscapes, have important deficiencies that we address. First, SYNTHIA dataset lacks the realism of the other two and do not have all 19 classes of Cityscapes. GTAV dataset lacks balance among classes and images are not enough populated due to game engine limitations. Finally, Synscapes lacks scenario variability where the layouts are a straight road and a crossroad. Thus, our dataset is characterized by better photo-realistic images than these datasets, properly balanced classes (we use the 19 Classes from Cityscapes), populated images, several urban layouts not seen in Synscapes or SYNTHIA, and a set of different illumination profiles.

4.3 Ordered composition of specialized models

Our new pseudo-labeling proposal is motivated by our previous work on co-training for semantic segmentation (Section 3.3). The qualitative results obtained during the self-training and co-training steps revealed an improvement margin on several classes represented in the pseudo-labels. The principal issue regarding the pseudo-labeling method is the per-class thresholding technique that we used, inspired by [60, 146, 147], to decide which pixels compose the pseudo-label and which pixels are filtered. Usually, these methods are conservative (low precision, high recall) and need the model to be overconfident to show rich pseudo-labels, making the model missing important information. Our co-training method, took these issues into account relaxing the decision boundaries and implementing a self-paced curriculum learning policy. However, relaxing the thresholds could increase the noise (lower recall) of several classes in the pseudo-label, becoming complex to find the parameters trade-off in the unsupervised scenario.

We address the aforementioned issue by proposing our ordered composition of

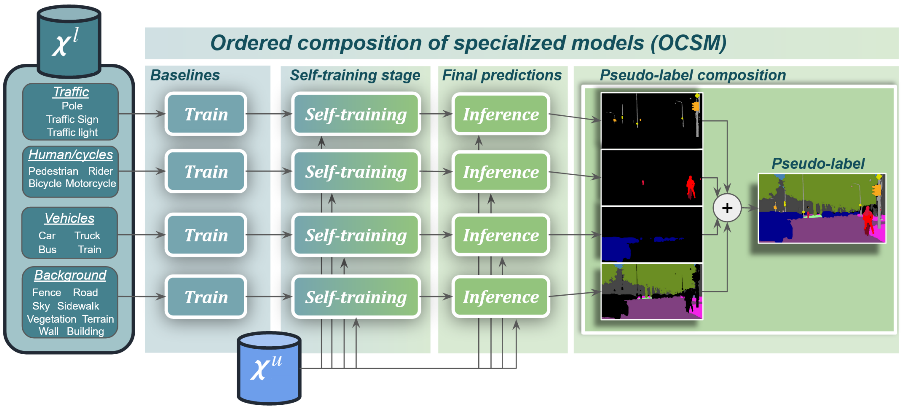


Figure 4.1: Framework of our ordered composition of specialized models (OCSM) method to generate pseudo-labels. $\mathscr{X}^l$ is a set of labeled synthetic images grouped by categories (related classes) and $\mathscr{X}^u$ is a set of unlabeled real-world images. We define four related categories: Vehicles (Car, Truck, Bus, and Train), Traffic (Poles, Traffic lights, and Traffic signs), humans/cycles (Pedestrian, Rider, Bike, and Motorbike), and Background (Road, Sidewalk, Building, Fence, Wall, Vegetation, Terrain, and Sky); so comprising the 19 classes defined in Cityscapes. We train one model for each category and apply a self-training step on them to generate pseudo-labels. Finally, we compose a fully pseudo-labeled image stacking each of these pseudo-labels ordered by area size (similar to a human labeler), background, vehicles, pedestrian/cycles, and traffic, respectively. These fully pseudo-labeled images are used to train a final model in combination with the source data.

specialized models framework, Figure 4.1, which we detail in the following.

Class categorization. We reduce the problem difficulty creating categories of classes, $\mathscr{X}^l_K$, from the source data $\mathscr{X}^l$. This step consists of generating new pixel-wise semantic segmentation labels with the categories specified and assigning a priority order similar to how a human labels (K). We define four coherent categories based on the 19 classes used in Cityscapes evaluation grouped by how related are the classes among them. We establish the priority order by the average area size of the classes that compose a category (from larger to smaller): Background ($K = 0$) groups static classes as Road, Sidewalk, Building, Wall, Fence, Sky, Vegetation, and Terrain; Vehicles ($K = 1$) groups Car, Truck, Bus, and Train; Humans/Cycles ($K = 2$) groups Pedestrian, Rider, Bicycle and Motorcycle; Traffic ($K = 3$) groups Pole, Traffic sign and Traffic lights. Additionally, for each category, we generate a new source ground truth with its respective classes where classes that do not belong to this category are relabeled as Unlabeled.

Baseline. For each category, we train a model, $\mathscr{W}_0^K$, using the aforementioned new source ground truths, $\mathscr{X}_K^l$, to obtain a baseline. We guarantee that each specialized model trained with these new ground truths also learns to separate category classes from the rest through the Unlabeled class. Learning the Unlabeled class is essential to remove undesired noise and avoid the necessity to apply a complex thresholding technique.

Self-training. We apply the self-training step, Algorithm 2 from chapter 3.3.1, in each $\mathscr{W}_0^K$ obtained from the baseline step with its respective hyper-parameters, $\mathscr{H}_{st} = \{\mathscr{T}, N, n, K_m, K_M, \mathscr{M}_{df}\}$, and we keep the weights from the last cycle, $\mathscr{W}^K{}_{K_M}$.

Pseudo-label composition. We compute the pseudo-labels, $\mathscr{X}^{\hat{l}}$, of the target images, $\mathscr{X}^u$, composing the pseudo-labels, $\mathscr{P}^K$, of each $\mathscr{W}^K{}_{K_M}$. In Equation 4.1 we formulate the composition process, where I denotes an image from $\mathscr{X}^u$, p are pixel coordinates, and C is the composition operation. The pixel value of the composed pseudo-label, $\mathscr{X}_I^{\hat{l}}(p)$ is obtained following the formula:

$$\begin{aligned}\forall I \in \mathscr{X}^u, \quad \forall p \in I, \quad \mathscr{X}_I^{\hat{l}}(p) &= C(\mathscr{P}^{K_M}(I(p)), C(\mathscr{P}^{K_M-1}(I(p)), ...)) \\ C(l_1, l_2) &= \begin{cases} l_1, & if \quad l_1 \neq U \\ l_2, & \text{otherwise} \end{cases}\end{aligned} \tag{4.1}$$

where $K_M = 3$. In plain words, when there is a labeling conflict for a given pixel, the assigned label corresponds to the one indicated by the larger category ID, provided the label is different than U. In our case, this means that classes with instances of smaller area size have higher priority.

The $\mathscr{X}^{\hat{l}}$ generated from $\mathscr{X}^u$ are used to train a final model analogous to the co-training final step. In other words, we train a model combining at batch time $\mathscr{X}^l$ data and $\mathscr{X}^u$ with its respective $\mathscr{X}^{\hat{l}}$.

4.4 Our dataset

In collaboration with a team of experts in computer graphics and technical artists, we develop the tools needed to generate our own synthetic data. Our proposal generates new photo-realistic synthetic images modeling a set of urban scenarios aimed at the training and validation of vision-based models in the context of autonomous driving. We set the number of classes to the 19 classes specified in the training of Cityscapes to facilitate the experimental setup on semantic segmentation.

Dataset Description

Our virtual scenarios are assembled by grouping assets into layers. A layer L includes assets that either belong to the same class or, while belonging to different classes, they typically appear together (i.e. traffic-sign and pole). We distinguish three types of layers: $L = \{E, S, D\}$, where E represents the environment lighting present in the scene, S are the layers belonging to static agents, and D are those layers representing dynamic agents. Within each layer we exploit procedural and/or semi-automated scattering tools to place the assets within the scenario, varying their appearance and/or poses. We will refer to these variations as intra-class variations. In addition, we handle layers containing elements from classes that could interfere with each other by ensuring that their respective intra-class variations never overlap with other layer variations (right now this is a human-in-the-loop task susceptible to further automation). As a result, our system can combine different inter-class variations to generate a whole range of scenarios.

We generated 2 different scenarios that act as template layouts for our scenarios, which are mainly defined by our static layers (S). We included (i) a roundabout; (ii) a main straight street with slight curves and L-intersections; and (iii) a street with a 4-ways intersection. Those static scenarios are finally populated with variations of the dynamic layers (D). While it is already easy to unintentionally introduce undesired bias in 2D datasets, even more often through 3D assets, so we ensured to achieve a minimum degree of variety for certain critical classes (i.e. gender and race for pedestrians).

The procedure for assembling the whole scenarios is performed through a sampling process exploiting all the available intra-class variations. While multiple choices are possible for this sampling procedure we adopted a simple random sampling strategy for now since it has already shown enough variability in the generated scenarios. Finally, we adopted modern GPU-accelerated path-tracing algorithms to render complex light transport phenomena relying on physically-based definitions for the geometries and materials within our recreated virtual worlds.

Scenario content creation

Our scenario layouts are inspired by real locations in the real world. We use Open-StreetMaps in order to get georeferenced map data with realistic building lot distributions. Then, for each layout, we use the third-party software RoadRunner to refine the main road layout (parking lots, number of lanes, sidewalks, etc.) and the location of fixed traffic signs and/or traffic lights. These data are exported into the OpenDrive format. Currently, our set of modeling tools is devised to assemble the main scenario layout using the Unity game engine. Next, we proceed to detail how the 19 different classes, are currently being treated in our content creation pipeline:

- Sky (scene illumination). Our lighting setup mainly relies on using a real captured High-Dynamic Range (HDR) environment image that is mapped on a sphere surrounding the virtual scenario. We usually prefer to use pure sky images.

- Roads and sidewalks. We start from the main geometry authored in RoadRunner and increase their amount of detail and realism with our own custom tessellation and decal-projection tools.

- Traffic signs and traffic lights. Our system can either utilize locations defined by RoadRunner, labeled by humans, or a combination of both.

- Poles. They may belong to different objects like street lamps, thick posts, or bollards. We simply instantiate predefined models in the scene.

- Buildings. We created our own asset in Houdini which is in charge of sampling and adapting subsets of available Unity prefabs modeling building block variations and assembling the final building configuration. Our asset also takes care of assigning different materials. We have distinguished basements from standard building floors as well as modeled some representative roof elements.

- Vegetation, terrain, fences, and walls. Vegetation is currently instantiated by hand depending on the scenario and terrain layouts. In some cases, a variation in the terrain may require reconsidering a different arrangement in terms of the inter-class compatible variations. Fences and walls can be easily placed to not overlap with other elements in the scene.

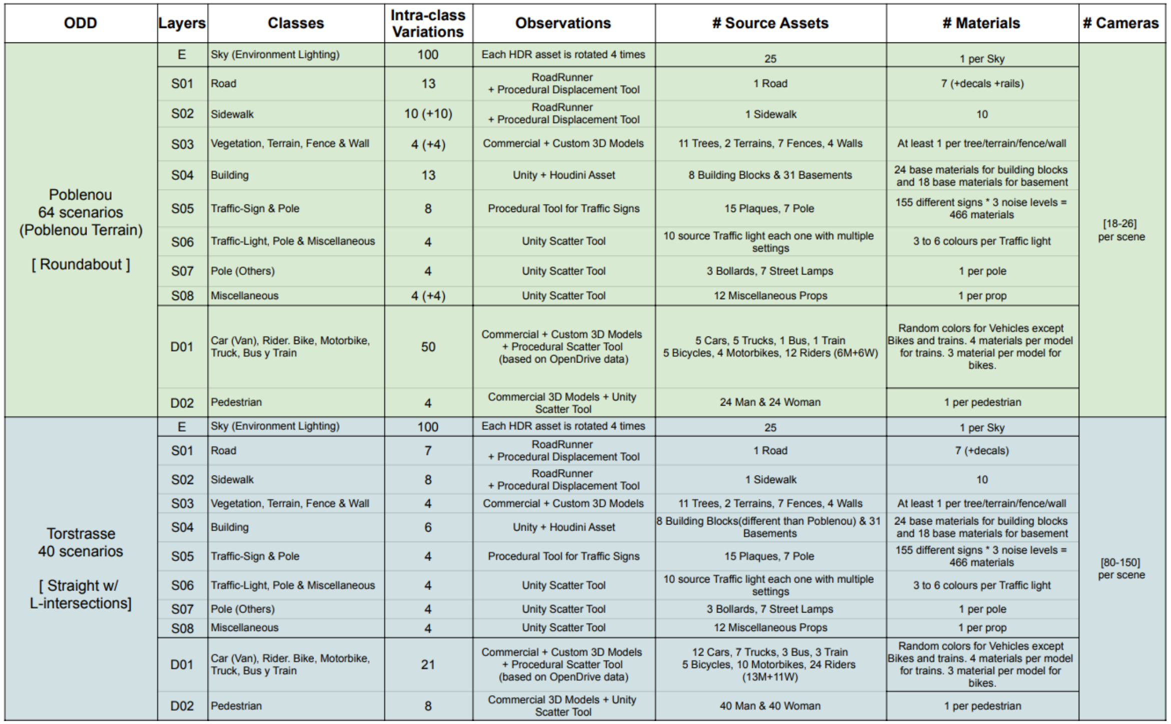
- Vehicles (includes cars, vans, buses, trucks, and trains). We begin with an original 3D asset and create several variations mainly related to their material appearance (color, dust level, wetness, etc.). Using an in-house tool we position those resulting vehicles based on the available OpenDrive specifications for the road (i.e. according to the number of lanes, lane direction, parking lots, etc.). Thus, we can easily generate variations by simply running our tool several times and exporting the various results. We typically need to account for the results in the generation of vehicles before generating other dynamic elements in the scene (i.e. pedestrians, riders, etc.). The rendered views are often sampled by placing our cameras in the virtual cars.

- Motorbikes, bicycles, and riders. We typically exploit empty positions in the road to place elements of these classes. Some bikes are often placed also in sidewalks since this is also common in the real world.

- Pedestrians. We leverage our own tool to place humans across our scene. Right now, human intervention is needed particularly when creating very crowded scenarios like crosswalks, etc. For shared spaces like the roads, these variations are tuned to be compatible with existing vehicle variations.

- Urban or miscellaneous elements. Variations of other elements in the scene (i.e. benches, terrace chairs, umbrellas, tables, etc.) are usually scattered over the place, taking care of not overlapping with other classes of interest potentially present in the scenario.

The final format describing our scenarios is simply a text file pointing to the specific intra-class variation files (stored in a render-friendly format). The sampling process can be guided by constraining the picking probability of some of the classes or even specifying the inter-class dependency of some variations (i.e. among the motorbike and rider classes). Since we are mostly interested in observing as many different image layouts as possible that contain the main classes of interest, the images generated in our dataset do not follow any driven path through the virtual world. Instead, we build our scenarios through a mixture of procedural generation techniques, partially hand-crafted intra-class variations, and a recombination process of all those variations through a statistical sampling procedure.



ODD	Layers	Classes	Intra-class Variations	Observations	# Source Assets	# Materials	# Cameras
Poblenou 64 scenarios (Poblenou Terrain) [Roundabout]	E	Sky (Environment Lighting)	100	Each HDR asset is rotated 4 times	25	1 per Sky	[18-26] per scene
	S01	Road	13	RoadRunner + Procedural Displacement Tool	1 Road	7 (+decals +rails)	
	S02	Sidewalk	10 (+10)	RoadRunner + Procedural Displacement Tool	1 Sidewalk	10	
	S03	Vegetation, Terrain, Fence & Wall	4 (+4)	Commercial + Custom 3D Models	11 Trees, 2 Terrains, 7 Fences, 4 Walls	At least 1 per tree/terrain/fence/wall	
	S04	Building	13	Unity + Houdini Asset	8 Building Blocks & 31 Basements	24 base materials for building blocks and 18 base materials for basement	
	S05	Traffic-Sign & Pole	8	Procedural Tool for Traffic Signs	15 Plaques, 7 Pole	155 different signs * 3 noise levels = 466 materials	
	S06	Traffic-Light, Pole & Miscellaneous	4	Unity Scatter Tool	10 source Traffic light each one with multiple settings	3 to 6 colours per Traffic light	
	S07	Pole (Others)	4	Unity Scatter Tool	3 Bollards, 7 Street Lamps	1 per pole	
	S08	Miscellaneous	4 (+4)	Unity Scatter Tool	12 Miscellaneous Props	1 per prop	
	D01	Car (Van), Rider. Bike, Motorbike, Truck, Bus y Train	50	Commercial + Custom 3D Models + Procedural Scatter Tool (based on OpenDrive data)	5 Cars, 5 Trucks, 1 Bus, 1 Train 5 Bicycles, 4 Motorbikes, 12 Riders (6M+6W)	Random colors for Vehicles except Bikes and trains. 4 materials per model for trains. 3 material per model for bikes.	
	D02	Pedestrian	4	Commercial 3D Models + Unity Scatter Tool	24 Man & 24 Woman	1 per pedestrian	
Torstrasse 40 scenarios [Straight w/ L-intersections]	E	Sky (Environment Lighting)	100	Each HDR asset is rotated 4 times	25	1 per Sky	[80-150] per scene
	S01	Road	7	RoadRunner + Procedural Displacement Tool	1 Road	7 (+decals)	
	S02	Sidewalk	8	RoadRunner + Procedural Displacement Tool	1 Sidewalk	10	
	S03	Vegetation, Terrain, Fence & Wall	4	Commercial + Custom 3D Models	11 Trees, 2 Terrains, 7 Fences, 4 Walls	At least 1 per tree/terrain/fence/wall	
	S04	Building	6	Unity + Houdini Asset	8 Building Blocks(different than Poblenou) & 31 Basements	24 base materials for building blocks and 18 base materials for basement	
	S05	Traffic-Sign & Pole	4	Procedural Tool for Traffic Signs	15 Plaques, 7 Pole	155 different signs * 3 noise levels = 466 materials	
	S06	Traffic-Light, Pole & Miscellaneous	4	Unity Scatter Tool	10 source Traffic light each one with multiple settings	3 to 6 colours per Traffic light	
	S07	Pole (Others)	4	Unity Scatter Tool	3 Bollards, 7 Street Lamps	1 per pole	
	S08	Miscellaneous	4	Unity Scatter Tool	12 Miscellaneous Props	1 per prop	
	D01	Car (Van), Rider. Bike, Motorbike, Truck, Bus y Train	21	Commercial + Custom 3D Models + Procedural Scatter Tool (based on OpenDrive data)	12 Cars, 7 Trucks, 3 Bus, 3 Train 5 Bicycles, 10 Motorbikes, 24 Riders (13M+11W)	Random colors for Vehicles except Bikes and trains. 4 materials per model for trains. 3 material per model for bikes.	
	D02	Pedestrian	8	Commercial 3D Models + Unity Scatter Tool	40 Man & 40 Woman	1 per pedestrian	

Figure 4.2: Summary table containing meaningful information related to the scenarios utilized to generate our image dataset.

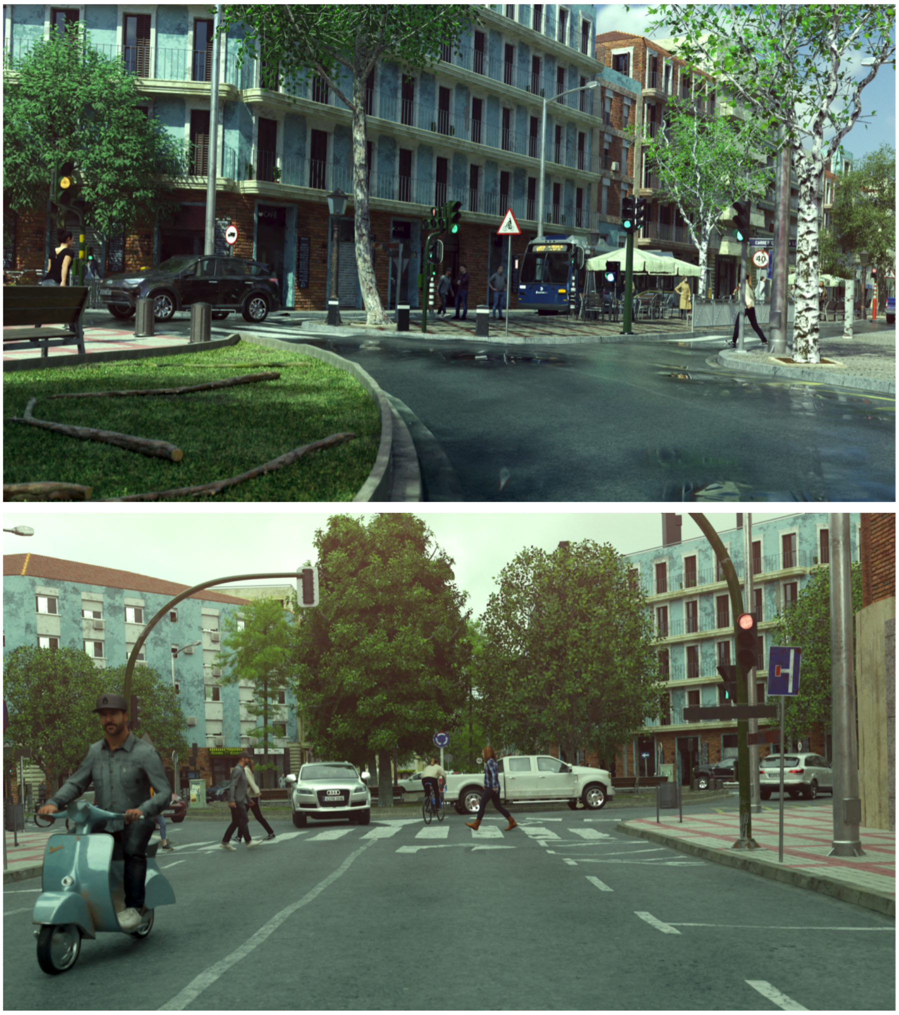


Figure 4.3: Image samples of our dataset for the Poblenou layout.

Our image dataset

Since the proposed system can already generate a large number of scenarios by just shuffling intra-class variations, we just randomly sampled a subset of all those

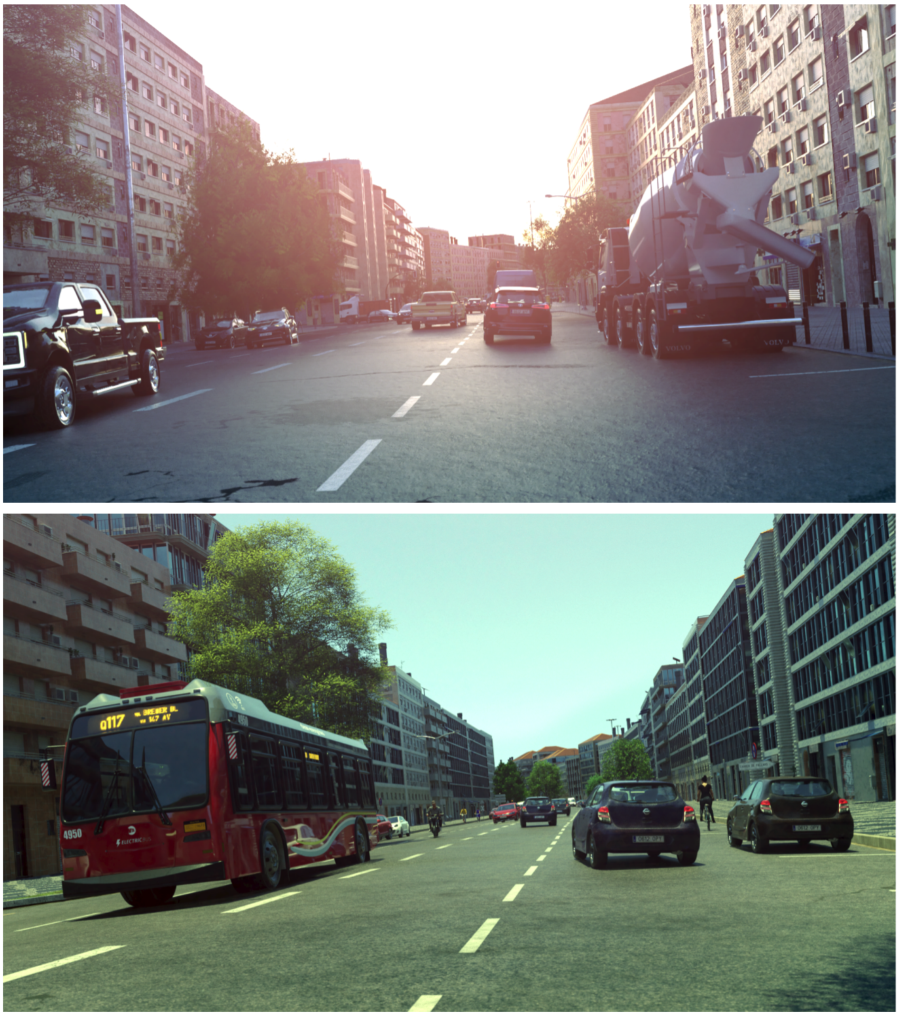


Figure 4.4: Image samples of our dataset for the Torstrasse layout.

possible scenarios distributed among the two main template layouts proposed in the scenario content creation. Then, we generated a finite number of images representative of the different image layouts that the proposed system can generate. A summary of the proposed scenarios and some of the meaningful numbers and

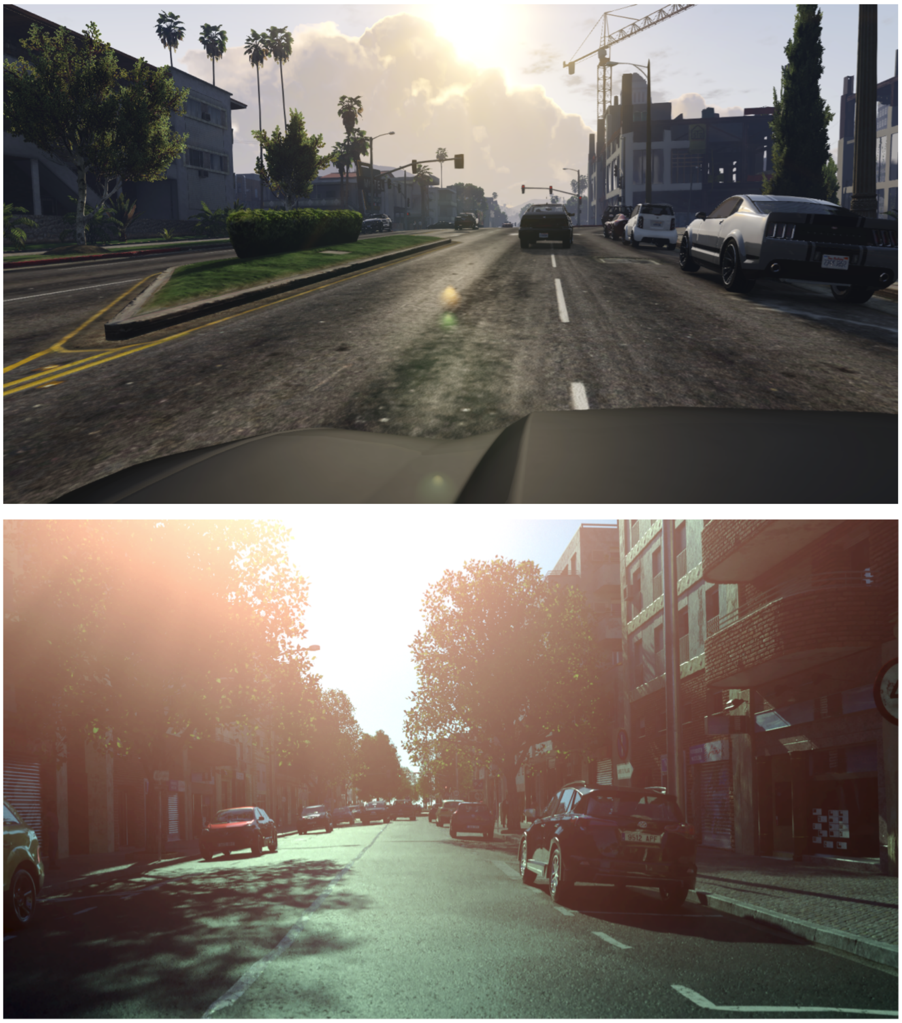


Figure 4.5: Qualitative comparative between an image from GTAV (top) and another from our dataset (bottom).

observations related to their creation process are provided in Figure 4.2. Figures 4.3 and 4.4 display several examples of the photo-realism of our data. In addition, figures 4.5 and 4.6 compare visually our data versus GTAV and Synscapes,

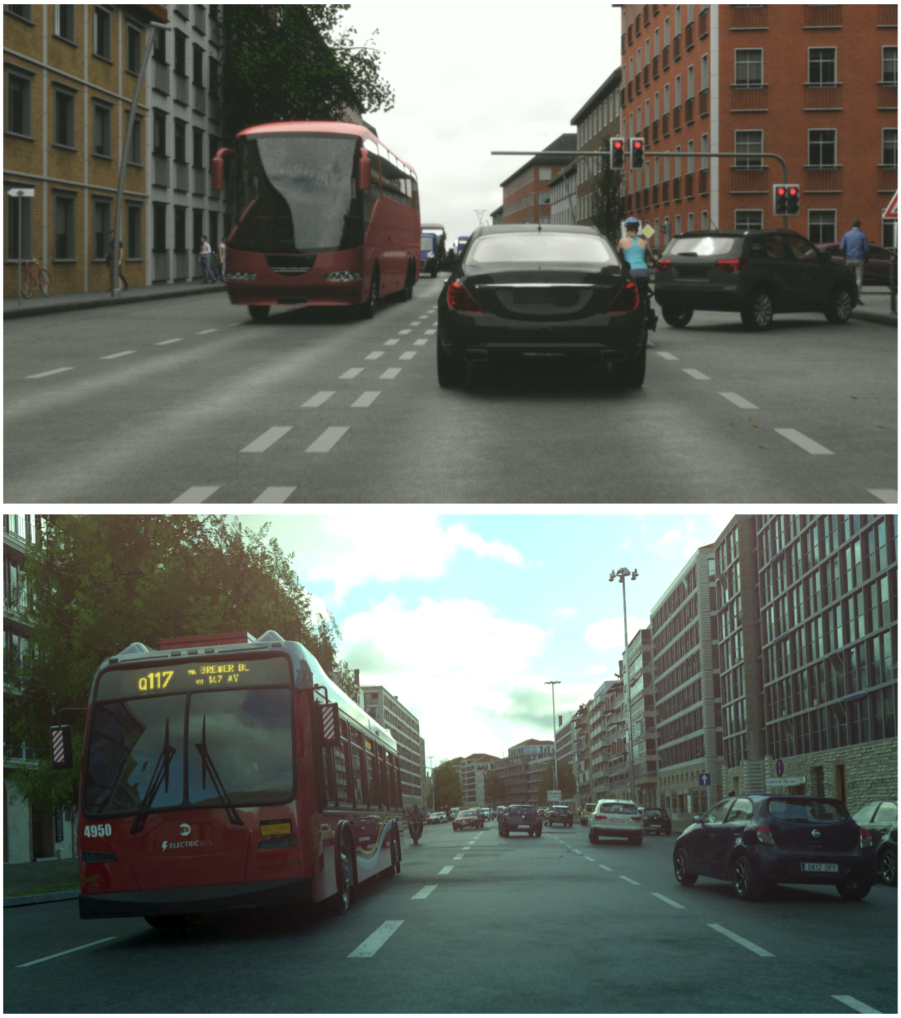


Figure 4.6: Qualitative comparative between an image from Synscapes (top) and another from our dataset (bottom).

respectively.

Table 4.1: Content statistics for the considered datasets. For each class, we indicate the percentage (%) of: (1) Images containing samples of the class, and (2) Pixels in the dataset with the class label.

		Road	Sidewalk	Building	Wall	Fence	Pole	Traffic light	Traffic Sign	Vegetation	Terrain	Sky	Person	Rider	Car	Truck	Bus	Train	Motorbike	Bicycle
Ours	Images	100.0	100.0	100.0	50.85	62.12	99.04	94.24	98.17	98.08	89.98	87.03	49.79	59.69	46.14	46.71	50.06	12.88	34.73	40.84
	Pixels	25.90	7.30	37.09	0.16	0.59	1.53	0.15	0.79	8.70	1.18	6.77	0.67	0.13	2.83	0.76	0.84	0.33	0.08	0.07
GTAV	Images	99.3	97.3	99.8	95.7	84.2	99.2	74.3	60.5	99.3	97.8	99.2	93.9	13.1	91.7	67.0	20.4	4.5	16.7	1.8
	Pixels	32.1	8.29	16.91	1.85	0.63	1.06	0.13	0.08	7.6	2.14	13.53	0.36	0.03	2.51	1.13	0.37	0.06	0.03	0.1
Synscapes	Images	99.9	98.7	99.1	32.8	98.2	99.6	96.7	98.9	97.6	48.2	98.0	99.7	82.3	98.5	86.9	80.8	65.8	86.5	89.5
	Pixels	28.29	6.82	22.57	1.13	1.37	2.07	0.37	0.53	13.93	0.88	8.31	3.18	0.79	5.59	0.79	1.08	1.17	0.6	0.52
Cityscapes	Images	98.6	94.5	98.6	32.6	43.6	99.1	55.7	94.4	70.3	55.6	90.3	78.8	34.4	95.2	12.1	9.2	4.8	17.2	55.3
	Pixels	32.63	5.39	20.19	0.58	0.78	1.09	0.18	0.49	14.08	1.03	3.55	1.08	0.12	6.19	0.24	0.21	0.21	0.09	0.37
BDD100K	Images	96.5	66.7	88.4	15.4	30.6	95.0	47.1	75.3	91.7	36.7	94.8	34.7	5.2	97.3	30.5	15.0	0.7	3.8	6.4
	Pixels	21.26	2.03	13.24	0.48	1.03	0.94	0.18	0.34	13.2	1.03	17.26	0.25	0.02	8.13	0.97	0.56	0.01	0.02	0.05
Mapillary	Images	98.8	72.2	91.3	46.2	62.7	98.6	51.0	91.0	96.8	46.9	98.9	50.5	19.0	93.4	24.0	15.8	1.3	15.1	16.7
	Pixels	19.42	2.99	12.47	0.75	1.26	0.9	0.18	0.45	14.94	1.05	29.33	0.31	0.06	3.36	0.37	0.26	0.02	0.06	0.07

4.5 Experimental results

4.5.1 Datasets and evaluation

We selected two well-known realistic synthetic datasets, GTAV [85] and Synscapes [119], to compare and combine them with our synthetic dataset (source domain). GTAV is composed by 24,904 images with a resolution of 1914 × 1052 pixels directly obtained from the render engine of the videogame GTA V. Synscapes is composed of 25,000 images with a resolution of 1440 × 720 pixels of urban scenes, obtained by using a physic-based rendering pipeline. Our dataset is composed of 6,236 images with a resolution of 2048 × 1024 pixels. As real-world datasets (target domain) we select Cityscapes [15], BDD100K [131] and Mapillary Vistas [70]. Cityscapes is a dataset well-known for its semantic segmentation challenge, composed of 3,475 labeled images split in 2.975 for training and 500 for validation, with an additional 1,525 non-labeled images for testing purposes. The images are obtained from different cities in Germany with the car onboard cameras under favorable conditions (e.g., no heavy occlusions or bad weather). Cityscapes images have a resolution of 2048 × 1024 pixels. Another dataset is BDD100K, which contains challenging onboard images taken from different vehicles, in different US cities, and under diverse weather conditions and daytime. The dataset is divided into 7,000 images for training purposes and 1,000 for validation with a resolution of 1280×720 pixels. Due to the layout of the synthetic datasets used, we filter manually the training images for our experiments and select daytime images, without heavy occlusions and favorable weather obtaining a subset of 1,777 images. The last dataset used is Mapillary Vistas composed of a large number of images around the world. These images were obtained with different camera devices generating a large variety of resolutions and aspect ratios. For simplicity, we only consider those images with an aspect ratio of 4:3, which, in practice, comprises more than 75% of all the dataset images. Hence, we have 14,716 images for training and 1,617 for validation. In Table 4.1 we summarize the composition of the datasets by the percentage of images and pixels of each class in all the datasets.

On other hand, we evaluate the results of our experiments on the validation set of each real-world dataset (target) using the 19 official classes defined for Cityscapes that are common in all our datasets. Any other classes that do not belong to these 19 classes are ignored during training and evaluation. Note that, even when we have available the labels of the target datasets, we are performing UDA and only use them on their respective validation sets to report exclusively quantitative results.

As is standard, we use the PASCAL VOC intersection-over-union metric $IoU = TP/(TP+FP+FN)$ [22], where TP, FP, and FN refer to true positives, false positives, and false negatives, respectively. IoU can be computed per class while using a mean IoU (mIoU) to consider all the classes at once. In addition, we display a confusion matrix to show the precision and recall metrics of each class. We display precision in a confusion matrix checking which pixels of all the pixels from the ground truth are matched by the predictions (predictions w.r.t ground truth) and recall checking which pixels of all the pixels from the prediction are matched by the ground truth (ground truth w.r.t predictions).

4.5.2 Implementation details

We use the Detectron2 [121] framework and leverage its implementation of DeepLabV3+ for semantic segmentation. Analogously, we apply similar hyper-parameters initialization for self-training and co-training specified in section 3.4.2 of this dissertation, we specify the hyper-parameters in Table 4.2. Furthermore, DeeplabV3+ follows exactly the same hyper-parameters specified in its original repository. Only adjusting the learning rates and training duration to our needs.

For training the semantic segmentation models, we use SGD optimizer with a

Table 4.2: Hyper-parameters of our framework from Sect. 3.3 and Alg. 2–4. Datasets: GTAV (G), Synscapes (S), Ours (O), Cityscapes (C), BDD (B), Mapillary Vistas (M).

								$\mathcal{M}_{df}$		$\mathcal{H}_{st}$				$\mathcal{H}_{ct}$			
Source	***Target***	N	n	Δp	C_m	C_M	N_{MB}	p_{MB}	p_{CM}	P_m	P_M	K_m	K_M	P_m	P_M	K	λ
G+S	C	500	200	0.05	0.5	0.9	4	0.5	0.5	0.5	0.7	1	10	0.5	0.5	5	0.8
G+S	B	500	200	0.05	0.5	0.9	4	0.5	0.5	0.4	0.6	1	10	0.5	0.5	5	0.8
G+S	M	500	200	0.05	0.5	0.9	16	0.5	0.5	0.5	0.7	1	10	0.5	0.5	5	0.8
G+S+O	C	500	200	0.05	0.5	0.9	4	0.5	0.5	0.5	0.7	1	10	0.5	0.5	5	0.8
G+S+O	B	500	200	0.05	0.5	0.9	4	0.5	0.5	0.5	0.7	1	10	0.5	0.5	5	0.8
G+S+O	M	500	200	0.05	0.5	0.9	16	0.5	0.5	0.5	0.7	1	10	0.5	0.5	5	0.8

starting learning rate of 0.002 and momentum 0.9. We crop the training images to 1024 × 512 pixels, 816 × 608, and 1280 × 720; and batch sizes to 4, 16, and 4 images when we work with Cityscapes, Mapillary Vistas, and BDD100K, respectively. Note that we set Mapillary Vistas batch size to 16 instead of 4 because receives a notable boost in performance, thus we updated all the related experiments with this dataset.

All baselines for DeepLabV3+ are trained during 90K iterations. Self-training and co-training training during 8K iterations targeting Cityscapes and BBD100K; and 4K iterations targeting Mapillary Vistas.

The hyper-parameter values of our self-training step of the specialized models are analogous to the self-training ones from section 3.3.1, the only difference is the number of cycles set to five ($K_M = 5$). In addition, analogous to co-training, the final model trained with the pseudo-labels from the training set of the target data uses the same values used to compute the baselines.

Note that all the experiments apply LAB adjustment of the source data with respect to to the target and multiple data sources are treated as a single (heterogeneous) source.

4.5.3 Study of our synthetic data

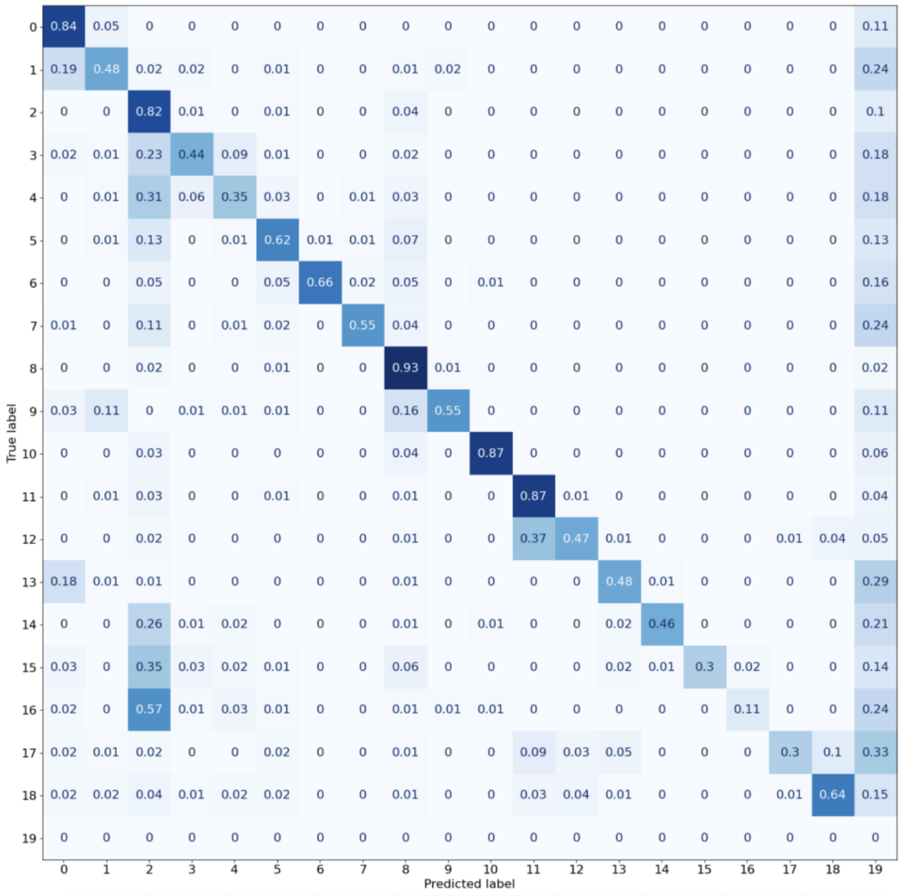
In Table 4.3 we compare the results obtained on Cityscapes as the target domain and the different synthetic datasets as a source: GTAV, Synscapes, and Ours. The first block of the table compares the results obtained from training models in a single source scenario and evaluates them on the Cityscapes validation set. We obtain the best mIoU with ~53 points, ~7 and ~10 points above Synscapes and GTAV, respectively. We surpass the other datasets in several background classes (Building, Sky, Sidewalk, Road, Fence, and Terrain), Truck, and Traffic lights. Our only drawback appears in the Car class. In fact, if we observe Figure 4.7 that displays the confusion matrix of the recall, class Car has a low recall, being confused by road (we do not account for the Void/Unlabeled class because is ignored in the metrics).

Other classes with low recall are Truck, Bus, and Train which are mostly confused by Building due to the sizes of these vehicles. In addition, Figure 4.8 shows qualitative results for each of these models on two samples of the Cityscapes validation set. Our results are less noisy in the background classes (Sky, Building, and Road) as we observed with the IoU metrics. Moreover, we visualize the problem with Cars where due to a bias in our dataset, the ego vehicle in cityscapes propagates noise as Class Car in the center of the road, explaining the lower IoU and recall. Nevertheless, the accuracy of Cars is as good as in the other datasets.

The second block of Table 4.3 compares the multi-source scenario where we combine multiple source datasets to train the models. As we expect, the addition of our dataset to Synscapes and GTAV boosts the final performance notably by ~7 points of mIoU in comparison to Synscapes + GTAV. More in detail, GTAV + Synscapes is the best combination of two sources with the best global mIoU, followed by Ours + Synscapes, only ~1 point behind. In addition, there are some noticeable properties to remark on with the possible combinations of these datasets. One of the most notable properties appears in class Train where we combine Syncapes and GTAV, the IoU jumps from less than ~5 points individually to ~40 points when combined. If we add our data improves even further, reaching ~62 points. The fact that the model is able to learn the class Train when we combine Synscapes and GTAV is not due to an increment of samples or variability of this class (GTAV samples of trains are scarce and highly occluded) but a disentangle between similar classes like Bus or Truck (more variability in theses classes help the model to learn Train). However, using our data does not produce the same outcome with Train, thus the variability from Synscapes and GTAV is needed for this class. Nevertheless, using our dataset boosts several other classes when we combine it with one of the other two, e.g.traffic sign, Sky, and Rider. In addition, we notice how important is GTAV to boost performance in Truck and Bus when combined with others. The variability added in these classes is better with GTAV than combining Synscapes and Our dataset, where we do not improve the single-source scenario. Analogous to the GTAV case, we found similar behavior in Synscapes with classes like Fence, Terrain, and Car. In conclusion, combining all the synthetic data bring benefits and mitigates the drawbacks of each one.

Table 4.3: mIoU of the baselines models trained on the synthetic datasets: GTAV (G), Synscapes (S), and Ours; and evaluated on the validation set of Cityscapes. We apply the LAB adjustment pre-processing to all the baselines. The first block reports the mIoU using only one source dataset and the second block reports the results by combining them.

Source data	Road	Sidewalk	Building	Wall	Fence	Pole	Traffic light	Traffic Sign	Vegetation	Terrain	Sky	Person	Rider	Car	Truck	Bus	Train	Motorbike	Bike	mIoU
Single Source → Cityscapes																				
Ours (O)	85.10	45.03	82.04	21.55	33.40	49.45	54.85	58.18	86.49	39.69	89.60	69.23	40.76	65.57	45.79	33.62	13.04	34.26	64.94	53.29
GTAV (G)	73.34	26.30	73.50	29.57	21.16	35.04	42.78	20.09	84.64	26.48	53.20	63.02	40.77	81.90	34.16	31.56	4.74	36.05	34.07	42.76
Synscapes (S)	78.39	37.47	67.39	16.45	19.09	48.5	51.79	58.54	83.18	29.89	64.79	70.17	29.27	85.39	18.42	10.42	3.32	36.48	64.61	45.98
Δ(O vs. G)	11.76	18.73	8.54	-8.02	12.24	14.41	12.07	38.09	1.85	13.21	36.40	6.21	-0.01	-16.33	11.63	2.06	8.30	-1.79	30.87	10.53
Δ(O vs. S)	6.71	7.56	14.65	5.10	14.31	0.95	3.06	-0.36	3.31	9.80	24.81	-0.94	11.49	-19.82	27.37	23.2	9.72	-2.22	0.33	7.31
Mutiple sources → Cityscapes																				
S+G	87.97	47.45	85.14	34.31	43.16	49.82	57.16	47.85	88.88	45.00	82.53	72.58	38.22	89.16	51.91	61.31	40.06	43.64	60.85	59.32
G+O	82.20	44.32	84.85	38.20	34.76	46.13	52.76	51.96	86.20	38.41	89.36	70.92	41.45	62.12	68.63	60.57	11.26	38.96	61.83	56.05
S+O	89.51	51.80	82.35	35.41	37.24	56.01	61.55	66.66	87.76	30.56	89.00	76.74	54.42	79.80	36.84	37.93	10.90	48.76	71.83	58.16
S+G+O	90.29	50.11	89.01	48.57	49.19	55.76	60.83	63.31	89.20	45.41	90.69	77.29	53.15	88.99	70.07	70.91	46.76	47.04	68.08	66.03
Δ(G+O vs. S+G)	-5.77	-3.13	-0.29	3.89	-8.40	-3.69	-4.40	4.11	-2.68	-6.59	6.83	-1.66	3.23	-27.04	16.72	-0.74	-28.80	-4.68	0.98	-3.27
Δ(S+O vs. S+G)	1.54	4.35	-2.79	1.10	-5.92	6.19	4.39	18.81	-1.12	-14.44	6.47	4.16	16.20	-9.36	-15.07	-23.38	-29.16	5.12	10.98	-1.16
Δ(S+G+O vs. S+G)	2.32	2.66	3.87	14.26	6.03	5.94	3.67	15.46	0.32	0.41	8.16	4.71	14.93	-0.17	18.16	9.60	6.70	3.40	7.23	6.71



True \ Predicted	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
0	0.84	0.05	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.11
1	0.19	0.48	0.02	0.02	0	0.01	0	0	0.01	0.02	0	0	0	0	0	0	0	0	0	0.24
2	0	0	0.82	0.01	0	0.01	0	0	0.04	0	0	0	0	0	0	0	0	0	0	0.1
3	0.02	0.01	0.23	0.44	0.09	0.01	0	0	0.02	0	0	0	0	0	0	0	0	0	0	0.18
4	0	0.01	0.31	0.06	0.35	0.03	0	0.01	0.03	0	0	0	0	0	0	0	0	0	0	0.18
5	0	0.01	0.13	0	0.01	0.62	0.01	0.01	0.07	0	0	0	0	0	0	0	0	0	0	0.13
6	0	0	0.05	0	0	0.05	0.66	0.02	0.05	0	0.01	0	0	0	0	0	0	0	0	0.16
7	0.01	0	0.11	0	0.01	0.02	0	0.55	0.04	0	0	0	0	0	0	0	0	0	0	0.24
8	0	0	0.02	0	0	0.01	0	0	0.93	0.01	0	0	0	0	0	0	0	0	0	0.02
9	0.03	0.11	0	0.01	0.01	0.01	0	0	0.16	0.55	0	0	0	0	0	0	0	0	0	0.11
10	0	0	0.03	0	0	0	0	0	0.04	0	0.87	0	0	0	0	0	0	0	0	0.06
11	0	0.01	0.03	0	0	0.01	0	0	0.01	0	0	0.87	0.01	0	0	0	0	0	0	0.04
12	0	0	0.02	0	0	0	0	0	0.01	0	0	0.37	0.47	0.01	0	0	0	0.01	0.04	0.05
13	0.18	0.01	0.01	0	0	0	0	0	0.01	0	0	0	0	0.48	0.01	0	0	0	0	0.29
14	0	0	0.26	0.01	0.02	0	0	0	0.01	0	0.01	0	0	0.02	0.46	0	0	0	0	0.21
15	0.03	0	0.35	0.03	0.02	0.01	0	0	0.06	0	0	0	0	0.02	0.01	0.3	0.02	0	0	0.14
16	0.02	0	0.57	0.01	0.03	0.01	0	0	0.01	0.01	0.01	0	0	0	0	0	0.11	0	0	0.24
17	0.02	0.01	0.02	0	0	0.02	0	0	0.01	0	0	0.09	0.03	0.05	0	0	0	0.3	0.1	0.33
18	0.02	0.02	0.04	0.01	0.02	0.02	0	0	0.01	0	0	0.03	0.04	0.01	0	0	0	0.01	0.64	0.15
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

0	1	2	3	4	5	6	7	8	9
Road	Sidewalk	Building	Wall	Fence	Pole	Traffic light	Traffic sign	Vegetation	Terrain

10	11	12	13	14	15	16	17	18
Sky	Person	Rider	Car	Truck	Bus	Train	Motorcycle	Bicycle

Figure 4.7: Confusion matrix of the recall of the baseline trained with our synthetic data and evaluated on Cityscapes. The recall is computed as a percentage of all the pixels from a class predicted which pixels match with the ground truth. Category 19 corresponds to the Void class.

4.5.4 Using our dataset for synth-to-real UDA

In Tables 4.4, 4.5 and 4.6; we summarize the results obtained in the multi-source scenario (with and without adding our data to Synsapes + GTAV) and applying co-training on Cityscapes, BDD100K and Mapillary Vistas as a target. We redo all the baselines, final models, and upper bounds, using Synscapes + GTAV as the source on BDD100K and Mapillary Vistas, to match the new hyper-parameters with larger training steps and batch sizes. These new hyper-parameters improve considerably the upper bounds and baselines in comparison to the previous chapter of this dissertation (see Table 3.7 from chapter 3.4.4), where the upper bounds were slightly under-trained for a deeplabV3+ model. In addition, increasing the batch sizes have a considerable improvement, especially on Mapillary Vistas where either baselines and upper-bounds increase ~7 mIoU points in comparison to previous chapters ones. Note, that as we stated previously, in the UDA scenario we do not rely on knowledge from the target (we only use labels to report results), thus the hyper-parameters selected do not rely on target information. These hyper-parameters changes are motivated by more resources available. Analogously, the hyper-parameters selection for our co-training method is done qualitatively, analyzing intermediate results in the pseudo-labeling process.

The impact of our dataset is clear in all the experiments done, where we boost globally the mIoU in all the baselines and co-training steps, being consistent with the results obtained in section 4.5.3. More in detail, Table 4.4 shows how co-training is still effective when we add our data to Syncapes + GTAV and target Cityscapes. However, the initial difference of ~6 mIoU points between baselines (SrcLAB) is reduced to only 2.47 points between co-trainings. We are above or similar in practically all the classes, except in Sidewalk and Terrain. We observe that the baseline, using our data, is better in all the classes, including Sidewalk and Terrain. Thus, discrepancies in these classes on co-training, emerge during the unsupervised process, where performance oscillations are common during self-training/co-training stages due to randomization in the training process. Furthermore, the self-training step is only 1.59 points below the co-training. These results indicate that we are reaching the upper bound in several classes with the pseudo-labeling methodology used. Several classes like Car, Bus, Building, Sky are only ~1 point from the upper-bond, others are between 2 to 8 points and the ones with the most improvement margin are Truck, Train, Sidewalk, Terrain, and Fence, where the distance is above 10 points.

In Table 4.5 differences using our dataset on BDD100K as a target are similar to the Cityscapes scenario. We are 5.24, 3.61, and 2.71 mIoU points better in baselines, self-training, and co-training, respectively, similar to Table 4.4. Analogous to the Sidewalk and Terrain case on Cityscapes with co-training, we are below in Traffic light and Sidewalk compared to the baseline. In addition, the gap with the

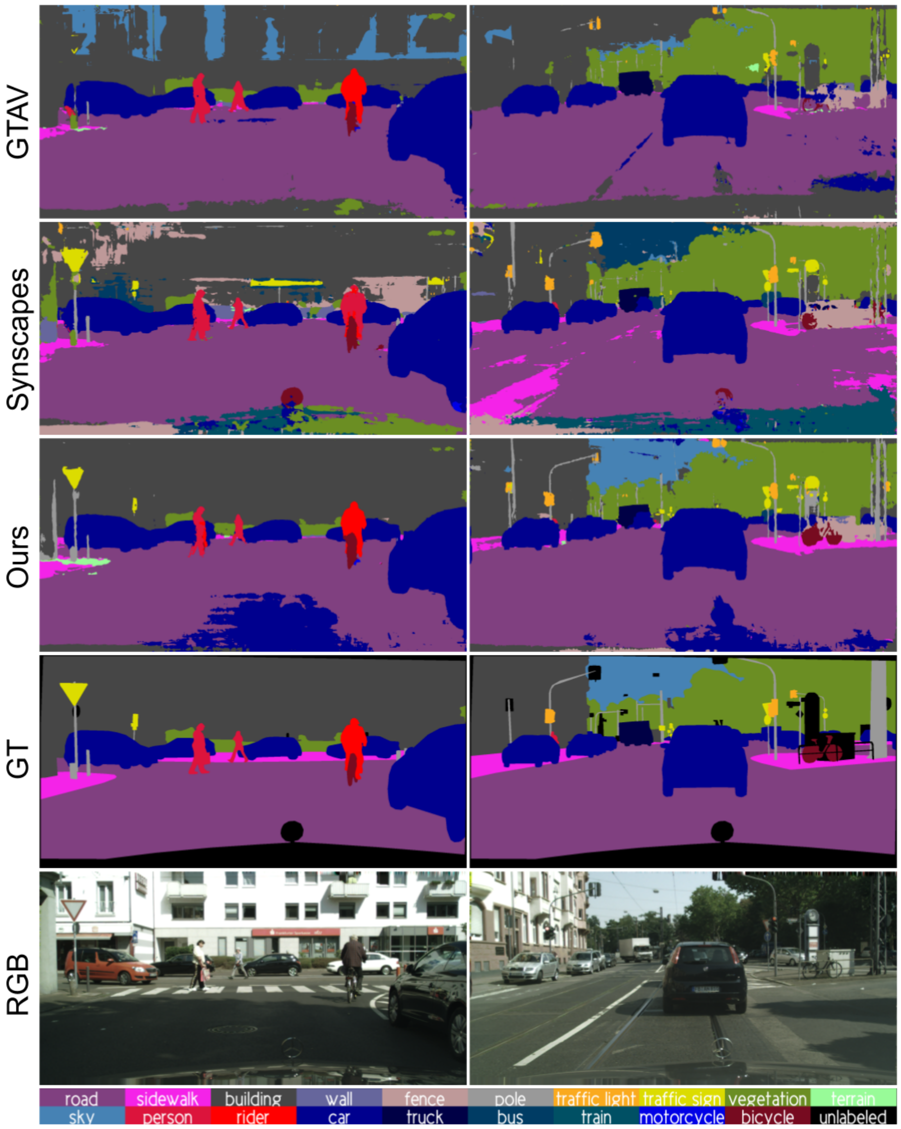


Figure 4.8: Qualitative results using the single-source models trained on GTAV, Synscapes, and Ours, respectively, on two samples of the Cityscapes validation set.

upper bound increased, in comparison to our previous chapter due to the hyper-parameters improvements, leaving a clear improvement margin in classes like Traffic lights, Traffic signs, Vegetation, Sidewalk, Bus, and Truck. This gap difference compared to the Cityscape scenario, can be explained because BDD100K is more challenging as we state in section 4.5.1. Moreover, all real datasets have important labeling bias (e.g. discrepancies between labels like Pickup vehicles labeled as trucks or cars depending on the Dataset) aggravating discrepancies between them, difficult to replicate in the synthetic scenario where the labels are objectives. These discrepancies generate lower upper bounds than expected in several classes that are difficult to label manually.

Finally, in Table 4.6 we analyze the same cases on Mapillary Vistas as a target. In this case, we improve by 8.64, 4.50, and 2.53 points in baselines, self-training, and co-training, respectively; then using only GTAV + Synscapes. In this scenario, we are above in practically all the classes, except Train, where the co-training step messed up obtaining a really low score. In addition, the gap between co-training and the upper bound is wider than the other real-world datasets, probably due to higher discrepancies in the labeling policies in comparison to the synthetic datasets. Thus, addressing the labeling discrepancies from Mapillary Vistas and the synthetic data could reduce the gap with respect to the upper bound.

4.5.5 OCSM experimental results

A large amount of diverse synthetic data available from the three synthetic datasets give us really good baselines. We found it interesting to explore the viability of category-specialized models. The reasoning behind this is to reduce the complexity of the problem expecting that a system trained with fewer classes should respond better to them, than a model trained with all the classes (global model). For this purpose, in Figures 4.9-4.11 we compare the confusion matrices of specialized models versus a single global model with all the classes, instead of relying only on the mIoU that can be misleading. In these figures, we show the accuracy and recall confusion matrices of the specialized model in each category (vehicles, humans/cycles, and traffic) and compare it with the confusion matrices of the global model. The most notorious characteristic of the specialized models is that generally are less accurate than the global model, but they have a higher recall. In fact, a high recall maintaining a reasonable accuracy is more desirable to improve the pseudo-label generation. In Figure 4.9 the recall of Cars, Truck, Bus, and Train is clearly higher than the global model and the accuracy is only slightly lower. Analogous in Figure 4.10, where class Traffic lights have better precision than its global counterpart, and in Figure 4.11 Rider present the same particularity.

Table 4.4: Co-training results compared to source LAB adjustment pre-processing (SrcLAB), self-training stage, and upper-bound (SrcLAB + Target) in the multi-source scenario (GTAV + Synscapes and GTAV + Syncapes + Ours) evaluated on target Cityscapes. Note that Target refers to using 100% of the target domain images labeled for training by a human oracle. .

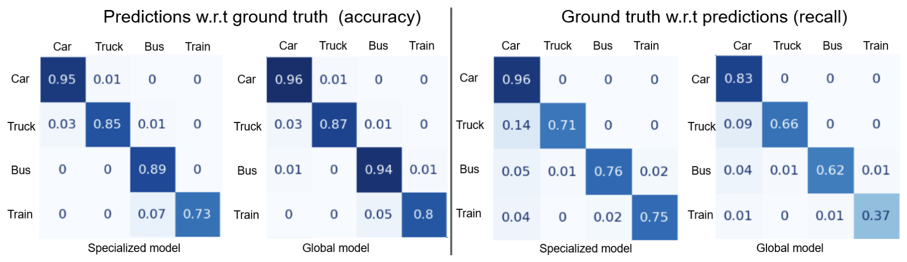
Methods	Road	Sidewalk	Building	Wall	Fence	Pole	Traffic light	Traffic Sign	Vegetation	Terrain	Sky	Person	Rider	Car	Truck	Bus	Train	Motorbike	Bike	mIoU
GTAV + Synscapes (Source) → Cityscapes																				
SrcLAB	87.97	47.45	85.14	34.31	43.16	49.82	57.16	47.85	88.88	45.00	82.53	72.58	38.22	89.16	51.91	61.31	40.06	43.64	60.85	59.32
Self-training	93.93	66.08	89.95	46.40	48.13	56.30	59.65	65.16	90.25	52.22	93.33	75.97	41.15	90.40	44.98	75.08	65.52	55.52	71.98	67.47
Co-training	96.30	74.72	90.44	48.89	49.15	58.36	61.52	67.05	90.75	54.75	93.52	79.48	57.71	90.48	45.61	85.11	59.95	60.41	70.96	70.23
SrcLAB + Target	98.24	85.85	93.08	59.45	66.58	66.36	70.99	78.67	92.64	65.38	94.98	82.99	65.68	95.44	78.98	90.49	79.71	68.57	77.90	79.58
Δ(Co-t vs. SrcLAB)	8.33	27.27	5.30	14.58	5.99	8.54	4.36	19.20	1.87	9.75	10.99	6.9	19.49	1.32	-6.3	23.8	19.89	16.77	9.28	10.91
Δ(Co-t vs. Self-t)	2.37	8.64	0.49	2.49	1.02	2.06	1.87	1.89	0.5	2.53	0.19	3.51	16.56	0.08	0.63	10.03	-5.57	4.89	-1.85	2.76
Δ(Co-t vs. SrcLAB + Tgt)	-1.94	-11.13	-2.64	-10.56	-17.43	-8.00	-9.47	-11.62	-1.89	-10.63	-1.46	-3.51	-7.97	-4.96	-33.37	-5.38	-19.76	-8.16	-7.77	-9.35
GTAV + Synscapes + Ours (Source) → Cityscapes																				
SrcLAB	90.29	50.11	89.01	48.57	49.19	55.76	60.83	63.31	89.2	45.41	90.69	79.17	59.46	92.69	66.80	73.11	61.94	52.91	67.65	66.03
Self-training	95.81	68.97	90.26	51.00	51.95	58.75	64.96	68.51	90.06	49.72	93.59	79.00	56.04	94.11	79.90	80.29	46.94	57.31	73.85	71.11
Co-training	95.59	67.95	91.24	56.26	53.16	60.22	64.84	70.27	90.04	48.74	93.96	79.44	57.56	94.38	70.03	89.42	64.66	59.71	73.84	72.70
SrcLAB + Target	98.14	85.22	92.90	61.68	64.50	65.41	70.31	78.58	92.66	65.98	94.77	82.42	64.40	95.29	85.93	90.62	82.00	68.27	77.47	79.82
Δ(Co-t vs. SrcLAB)	5.30	17.84	2.23	7.69	3.97	4.46	4.01	6.96	0.84	3.33	3.27	2.15	4.41	5.39	-0.04	18.51	17.90	12.67	5.76	6.67
Δ(Co-t vs. Self-t)	-0.22	-1.02	0.98	5.26	1.21	1.47	-0.12	1.76	-0.02	-0.98	0.37	0.44	1.52	0.27	-9.87	9.13	17.72	2.40	-0.01	1.59
Δ(Co-t vs. SrcLAB + Tgt)	-2.55	-17.27	-1.66	-5.42	-11.34	-5.19	-5.47	-8.31	-2.62	-17.24	-0.81	-2.98	-6.84	-0.91	-15.9	-1.2	-17.34	-8.56	-3.63	-7.12
Self-t Δ(S+G+O vs. S+G)	1.88	2.89	0.31	4.60	3.82	2.45	5.31	3.35	-0.19	-2.50	0.26	3.03	14.89	3.71	34.92	5.21	-18.58	1.79	1.87	3.64
Co-t Δ(S+G+O vs. S+G)	-0.71	-6.77	0.80	7.37	4.01	1.86	3.32	3.22	-0.71	-6.01	0.44	-0.04	-0.15	3.90	24.42	4.31	4.71	-0.70	3.71	2.47

Table 4.5: Analogous to Table 4.4 targeting BDD100K.

Methods	Road	Sidewalk	Building	Wall	Fence	Pole	Traffic light	Traffic Sign	Vegetation	Terrain	Sky	Person	Rider	Car	Truck	Bus	Train	Motorbike	Bike	mIoU
GTAV + Synscapes (Source) → BDD100K																				
SrcLAB	67.90	27.84	73.96	8.68	27.46	34.90	36.09	26.65	74.79	27.31	86.05	55.64	33.88	64.23	20.72	26.96	0.05	45.10	45.19	41.23
Self-training	87.45	45.20	78.46	18.24	38.52	39.75	41.62	26.38	73.54	35.75	87.63	57.57	27.94	78.37	34.38	56.35	0.11	45.44	57.49	48.96
Co-training	89.22	54.65	79.06	20.42	40.50	40.37	39.81	23.32	72.81	30.43	87.52	63.47	42.82	81.93	44.80	65.06	0.35	43.59	58.34	51.50
SrcLAB + Target	94.89	66.46	86.97	33.07	53.69	51.96	56.51	54.71	87.07	50.88	95.49	68.87	48.16	91.01	61.18	83.41	0.00	63.18	51.32	63.10
Δ(Co-t vs. SrcLAB)	21.32	26.81	5.10	11.74	13.04	5.47	3.72	-3.33	-1.98	3.12	1.47	7.83	8.94	17.70	24.08	38.10	0.30	-1.51	13.15	10.27
Δ(Co-t vs. Self-t)	1.77	9.45	0.60	2.18	1.98	0.62	-1.81	-3.06	-0.73	-5.32	-0.11	5.90	14.88	3.56	10.42	8.71	0.24	-1.85	0.85	2.54
Δ(Co-t vs. SrcLAB + Tgt)	-5.67	-11.81	-7.91	-12.65	-13.19	-11.59	-16.70	-31.39	-14.26	-20.45	-7.97	-5.40	-5.34	-9.08	-16.38	-18.35	0.35	-19.59	7.02	-11.60
GTAV + Synscapes + Ours (Source) → BDD100K																				
SrcLAB	79.24	35.02	73.51	16.86	27.22	41.17	42.64	32.42	69.41	31.00	85.29	61.51	22.22	69.04	35.48	52.49	0.12	54.69	53.51	46.47
Self-training	91.45	48.65	80.11	19.12	40.95	45.23	40.46	37.86	72.17	38.27	87.20	58.78	35.20	84.73	42.38	72.00	0.43	48.25	55.67	52.57
Co-training	92.31	51.73	80.15	23.31	46.37	46.20	31.72	37.29	70.42	37.78	85.92	66.04	49.52	86.55	44.37	72.04	1.36	52.56	54.32	54.21
SrcLAB + Target	95.11	66.53	86.75	31.46	53.74	52.93	56.92	54.29	87.07	49.45	95.47	69.17	55.39	90.64	57.24	83.98	0.00	55.08	56.89	63.06
Δ(Co-t vs. SrcLAB)	13.07	16.71	6.64	6.45	19.15	5.03	-10.92	4.87	1.01	6.78	0.63	4.53	27.30	17.51	8.89	19.55	1.24	-2.13	0.81	7.74
Δ(Co-t vs. Self-t)	0.86	3.08	0.04	4.19	5.42	0.97	-8.74	-0.57	-1.75	-0.49	-1.28	7.26	14.32	1.82	1.99	0.04	0.93	4.31	-1.35	1.64
Δ(Co-t vs. SrcLAB + Tgt)	-2.80	-14.80	-6.60	-8.15	-7.37	-6.73	-25.20	-17.00	-16.65	-11.67	-9.55	-3.13	-5.87	-4.09	-12.87	-11.94	1.36	-2.52	-2.57	-8.85
SrcLAB Δ(S+G+O vs. S+G)	11.34	7.18	-0.45	8.18	-0.24	6.27	6.55	5.77	-5.38	3.69	-0.76	5.87	-11.66	4.81	14.76	25.53	0.07	9.59	8.32	5.24
Self-t Δ(S+G+O vs. S+G)	4.00	3.45	1.65	0.88	2.43	5.48	-1.16	11.48	-1.37	2.52	-0.43	1.21	7.26	6.36	8.00	15.65	0.32	2.81	-1.82	3.61
Co-t Δ(S+G+O vs. S+G)	3.09	-2.92	1.09	2.89	5.87	5.83	-8.09	13.97	-2.39	7.35	-1.6	2.57	6.70	4.62	-0.43	6.98	1.01	8.97	-4.02	2.71

Table 4.6: Analogous to Table 4.4 targeting Mapillary Vistas.

Methods	Road	Sidewalk	Building	Wall	Fence	Pole	Traffic light	Traffic Sign	Vegetation	Terrain	Sky	Person	Rider	Car	Truck	Bus	Train	Motorbike	Bike	mIoU
GTAV + Synscapes (Source) → Mapillary																				
SrcLAB	70.42	31.99	79.21	18.35	35.55	41.72	57.03	51.87	77.71	37.95	90.89	71.92	39.23	86.12	47.24	41.92	5.99	49.14	50.31	51.82
Self-training	82.81	42.60	85.01	33.28	44.93	47.24	61.39	65.39	81.95	55.66	95.18	75.50	46.70	90.36	56.80	60.50	23.09	49.01	55.35	60.67
Co-training	87.47	50.73	85.67	39.36	48.24	50.44	61.62	69.49	79.83	54.96	94.43	76.82	58.26	91.78	65.03	65.61	24.83	56.29	64.07	64.67
SrcLAB + Target	96.25	79.17	90.95	57.55	67.20	64.49	70.06	81.21	91.98	74.11	98.58	80.46	63.08	93.38	77.78	77.69	65.77	64.85	71.41	77.16
Δ(Co-t vs. SrcLAB)	17.05	18.74	6.46	21.01	12.69	8.72	4.59	17.62	2.12	17.01	3.54	4.90	19.03	5.66	17.79	23.69	18.84	7.15	13.76	12.65
Δ(Co-t vs. Self-t)	4.66	8.13	0.66	6.08	3.31	3.20	0.23	4.10	-2.12	-0.70	-0.75	1.32	11.56	1.42	8.23	5.11	1.74	7.28	8.72	3.80
Δ(Co-t vs. SrcLAB + Tgt)	-8.78	-28.44	-5.28	-18.19	-18.96	-14.05	-8.44	-11.72	-12.15	-19.15	-4.15	-3.64	-4.82	-1.60	-12.75	-12.08	-40.94	-8.56	-7.34	-12.69
GTAV + Synscapes + Ours (Source) → Mapillary																				
SrcLAB	82.42	44.68	82.73	34.72	44.01	49.46	59.81	65.98	80.10	45.01	94.53	75.01	55.00	87.21	56.90	51.70	17.18	57.51	64.55	60.46
Self-training	89.90	54.66	85.68	32.67	52.44	51.82	63.18	73.60	84.16	54.38	96.28	77.08	60.01	88.89	63.02	65.76	22.92	58.15	63.71	65.17
Co-training	91.24	54.65	86.60	45.31	55.02	51.77	62.92	74.44	82.88	56.99	95.67	79.33	63.27	90.57	71.02	75.23	9.12	60.92	65.96	67.00
SrcLAB + Target	96.30	79.10	90.90	57.37	67.40	65.25	70.46	82.49	92.04	73.39	98.60	81.58	66.87	93.88	78.93	81.83	66.56	66.83	72.20	78.00
Δ(Co-t vs. SrcLAB)	8.82	9.97	3.87	10.59	11.01	2.31	3.11	8.46	2.78	11.98	1.14	4.32	8.27	3.36	14.12	23.53	-8.06	3.21	1.41	6.54
Δ(Co-t vs. Self-t)	1.34	-0.01	0.92	12.64	2.58	-0.05	-0.26	0.84	-1.28	2.61	-0.61	2.25	3.26	1.68	8.00	9.47	-13.80	2.77	2.25	1.83
Δ(Co-t vs. SrcLAB + Tgt)	-5.06	-24.45	-4.30	-12.06	-12.38	-13.48	-7.54	-8.05	-9.16	-16.40	-2.93	-2.25	-3.60	-3.31	-7.91	-6.60	-57.44	-5.91	-6.24	-11.00
SrcLAB Δ(S+G+O vs. S+G)	12.00	12.69	3.52	16.37	8.46	7.74	2.78	14.11	2.39	7.06	3.64	3.09	15.77	1.09	9.66	9.78	11.19	8.57	14.24	8.64
Self-t Δ(S+G+O vs. S+G)	7.09	12.06	0.67	-0.61	7.51	4.58	1.79	8.21	2.21	-1.28	1.10	1.58	13.31	-1.47	6.22	5.26	-0.17	9.14	8.36	4.50
Co-t Δ(S+G+O vs. S+G)	3.77	3.92	0.93	5.95	6.78	1.33	1.30	4.95	3.05	2.03	1.24	2.51	5.01	-1.21	5.99	9.62	-15.71	4.63	1.89	2.53


Predictions w.r.t ground truth (accuracy)

Specialized model

	Car	Truck	Bus	Train
Car	0.95	0.01	0	0
Truck	0.03	0.85	0.01	0
Bus	0	0	0.89	0
Train	0	0	0.07	0.73

Global model

	Car	Truck	Bus	Train
Car	0.96	0.01	0	0
Truck	0.03	0.87	0.01	0
Bus	0.01	0	0.94	0.01
Train	0	0	0.05	0.8

Ground truth w.r.t predictions (recall)

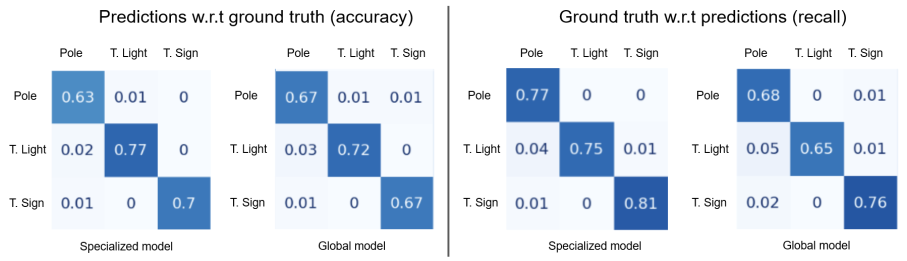
Specialized model

	Car	Truck	Bus	Train
Car	0.96	0	0	0
Truck	0.14	0.71	0	0
Bus	0.05	0.01	0.76	0.02
Train	0.04	0	0.02	0.75

Global model

	Car	Truck	Bus	Train
Car	0.83	0	0	0
Truck	0.09	0.66	0	0
Bus	0.04	0.01	0.62	0.01
Train	0.01	0	0.01	0.37

Figure 4.9: Confusion matrix of the specialized model trained grouping the vehicle categories (Car, Truck, Bus, and Train) compared to the confusion matrix of the global model trained on the 19 classes of Cityscapes. The left part represents the accuracy showing the confusion matrix comparison of the percentage of pixels from the ground truth properly classified (predictions w.r.t ground truth). Analogously, the right part represents the recall showing from all the pixels from a class predicted which percentage match with the ground truth (ground truth w.r.t predictions).


Predictions w.r.t ground truth (accuracy)

Specialized model

	Pole	T. Light	T. Sign
Pole	0.63	0.01	0
T. Light	0.02	0.77	0
T. Sign	0.01	0	0.7

Global model

	Pole	T. Light	T. Sign
Pole	0.67	0.01	0.01
T. Light	0.03	0.72	0
T. Sign	0.01	0	0.67

Ground truth w.r.t predictions (recall)

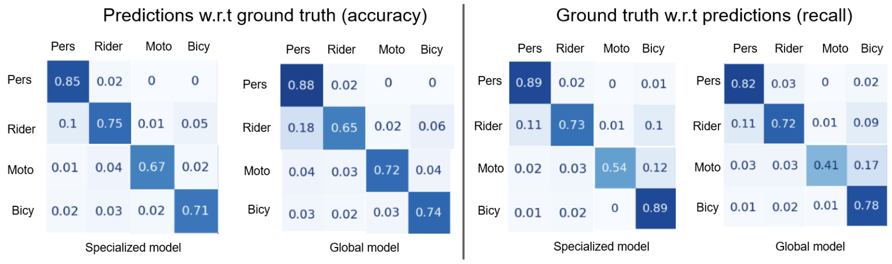
Specialized model

	Pole	T. Light	T. Sign
Pole	0.77	0	0
T. Light	0.04	0.75	0.01
T. Sign	0.01	0	0.81

Global model

	Pole	T. Light	T. Sign
Pole	0.68	0	0.01
T. Light	0.05	0.65	0.01
T. Sign	0.02	0	0.76

Figure 4.10: Analogous to Figure 4.9, comparing the specialized model trained grouping the traffic-related categories (Pole, Traffic light, and Traffic sign) versus its respective classes on the global model.



Predictions w.r.t ground truth (accuracy) — Specialized model

	Pers	Rider	Moto	Bicy
Pers	0.85	0.02	0	0
Rider	0.1	0.75	0.01	0.05
Moto	0.01	0.04	0.67	0.02
Bicy	0.02	0.03	0.02	0.71

Predictions w.r.t ground truth (accuracy) — Global model

	Pers	Rider	Moto	Bicy
Pers	0.88	0.02	0	0
Rider	0.18	0.65	0.02	0.06
Moto	0.04	0.03	0.72	0.04
Bicy	0.03	0.02	0.03	0.74

Ground truth w.r.t predictions (recall) — Specialized model

	Pers	Rider	Moto	Bicy
Pers	0.89	0.02	0	0.01
Rider	0.11	0.73	0.01	0.1
Moto	0.02	0.03	0.54	0.12
Bicy	0.01	0.02	0	0.89

Ground truth w.r.t predictions (recall) — Global model

	Pers	Rider	Moto	Bicy
Pers	0.82	0.03	0	0.02
Rider	0.11	0.72	0.01	0.09
Moto	0.03	0.03	0.41	0.17
Bicy	0.01	0.02	0.01	0.78

Figure 4.11: Analogous to Figure 4.9, comparing the specialized model trained grouping the human-related categories (Pedestrian, Rider, Motorcycle, and Bicycle) versus its respective classes on the global model.

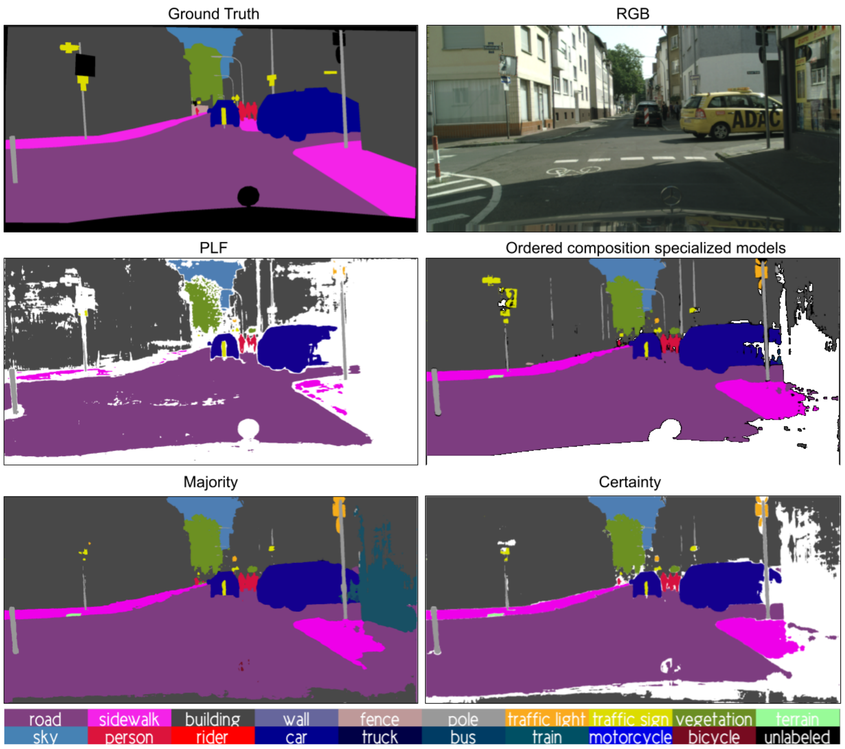


Figure 4.12: Pseudo-label sample generated by four different methods. Majority, Certainly, and PLF (Priority) are ensemble methods from [9], ordered composition of specialized models is our proposal.

Table 4.7: mIoU of the global models (GM) trained on 19 classes and the model trained on the pseudo-labels from the ordered composition of specialized models (OCSM). We refer to the baseline models as Base and the self-training step as Self-t. We report results using Synscapes (S) + GTAV (G) + Our dataset (O).

	Background								Traffic			Human/cycles				Vehicles				
Model	Road	Sidewalk	Building	Wall	Fence	Vegetation	Terrain	Sky	Pole	Traffic light	Traffic Sign	Person	Rider	Motorbike	Bike	Car	Truck	Bus	Train	mIoU
S + G + O → Cityscapes																				
GM - Base.	90.29	50.11	89.01	48.57	49.19	89.2	45.41	90.69	55.76	60.83	63.31	77.29	53.15	47.04	68.08	88.99	70.07	70.91	46.76	66.03
OCSM - base.	91.56	50.90	90.27	48.76	56.85	90.30	47.30	90.43	61.95	67.86	73.77	80.48	58.30	57.38	74.47	93.91	75.15	84.38	60.70	71.30
Δ(OCSM vs. GM) Base.	1.27	0.79	1.26	0.19	7.66	1.10	1.89	-0.26	6.19	7.03	10.46	3.19	5.15	10.34	6.39	4.92	5.08	13.47	13.94	5.27
GM - Self-t	95.81	68.97	90.26	51.00	51.95	90.06	49.72	93.59	58.75	64.96	68.51	79.00	56.04	57.31	73.85	94.11	79.90	80.29	46.94	71.11
OCSM - Self-t	96.05	71.83	90.25	47.50	50.56	90.67	53.75	93.00	62.08	68.93	74.71	79.73	58.19	60.95	74.26	94.22	77.27	87.91	72.00	73.89
Δ(OCSM vs. GM) Self-t	0.24	2.86	-0.01	-3.50	-1.39	0.61	4.03	-0.59	3.33	3.97	6.20	0.73	2.15	3.64	0.41	0.11	-2.63	7.62	25.06	2.78
Δ(Base. vs. Self-t) OCSM	4.49	20.93	-0.02	-1.26	-6.29	0.37	6.45	2.57	0.13	1.07	0.94	-0.75	-0.11	3.57	-0.21	0.31	2.12	3.53	11.30	2.59

After asserting that our specialized models could be useful to generate better pseudo-labels, we apply our method of ordered composition of specialized models (OCSM). First, we compare our approach with the ensemble methods applied to UDA from Chao et al. [9], where they show three different ensemble methods in their end-to-end ensemble distillation framework: Certainty, Majority, and Priority (PLF). In Figure 4.12 we show a pseudo-label sample generated by these three methods using the outputs of our specialized models and our OCSM method. Majority (bottom-left) method fills all the pseudo-label introducing a not desired high amount of false positives (e.g. Train patch on the right) and prioritizing larger classes over smaller ones (e.g. Building over Pole). Certainly, solves the false positives from the majority but keeps prioritizing larger classes. PLF is more conservative and has a low recall. These ensemble methods are not suitable to combine specialized models, where an Unlabeled class is learned with high confidence and pixel conflicts are related to this class, they were designed to ensemble global models. Thus, our OCSM method generates more accurate pseudo-labels because avoids larger classes overriding smaller ones and leverages the Unlabeled class learned in the models to reduce the introduction of false positives.

in Table 4.7, we detail the results obtained with our OCSM method using GTAV + Syncapes + our dataset evaluated on Cityscapes validation set. In the first block of the table, we compare the global model (baseline model trained with all the source data) with a model trained with the OCSM pseudo-labels before the self-training step. We are able to improve by ~5 mIoU points compared to the global model. More in detail, categories where the specialized model responded better like Vehicles, Humans/Cycles, and Traffic: are the ones that receive the highest boost. Overall, the mIoU obtained equalizes the score obtained by applying the self-training step to the global model. The second block of the table compares both, the global model and the OCSM pseudo-labels scores, after a self-training step. As we expected, applying a self-training step to each specialized model benefits them in the same degree as applying it to a global model and therefore, the OCSM step surpasses clearly the global model by ~3 points. Finally, in the last row, we compare our OCSM before and after the self-training step to have a clear view of which classes have improved. The major contributions of the self-training stage fall in the Background category, where Sidewalk receives the most important boost; the vehicle category follows this improvement on Train and Bus; finally, Traffic and Human/Cycle categories receive little to non-improvement.

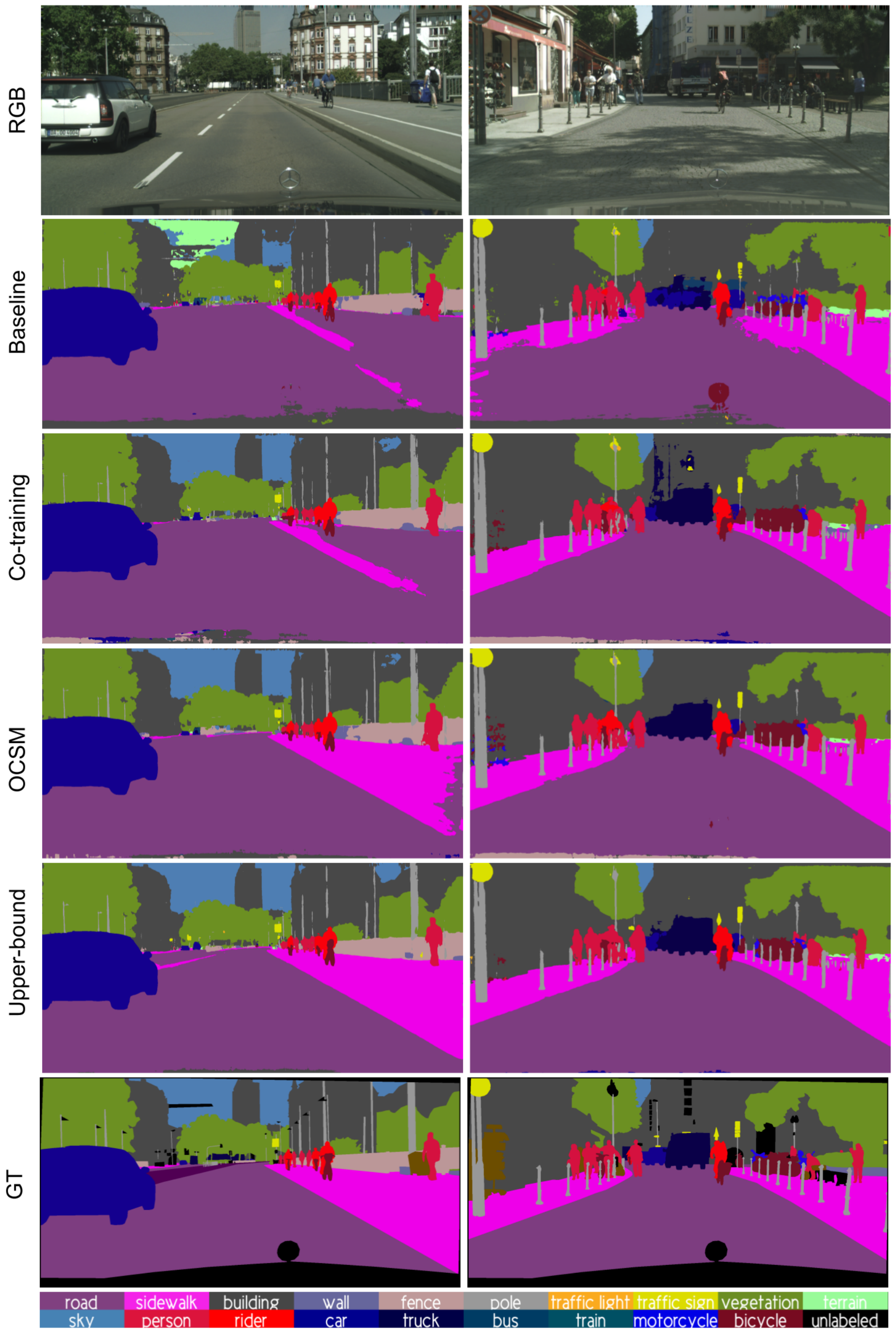


Figure 4.13: Qualitative results on Cityscapes validation samples of Baseline, Co-training, OCSM, and Upper-bound models using GTAV + Synscapes + Ours data as a source.

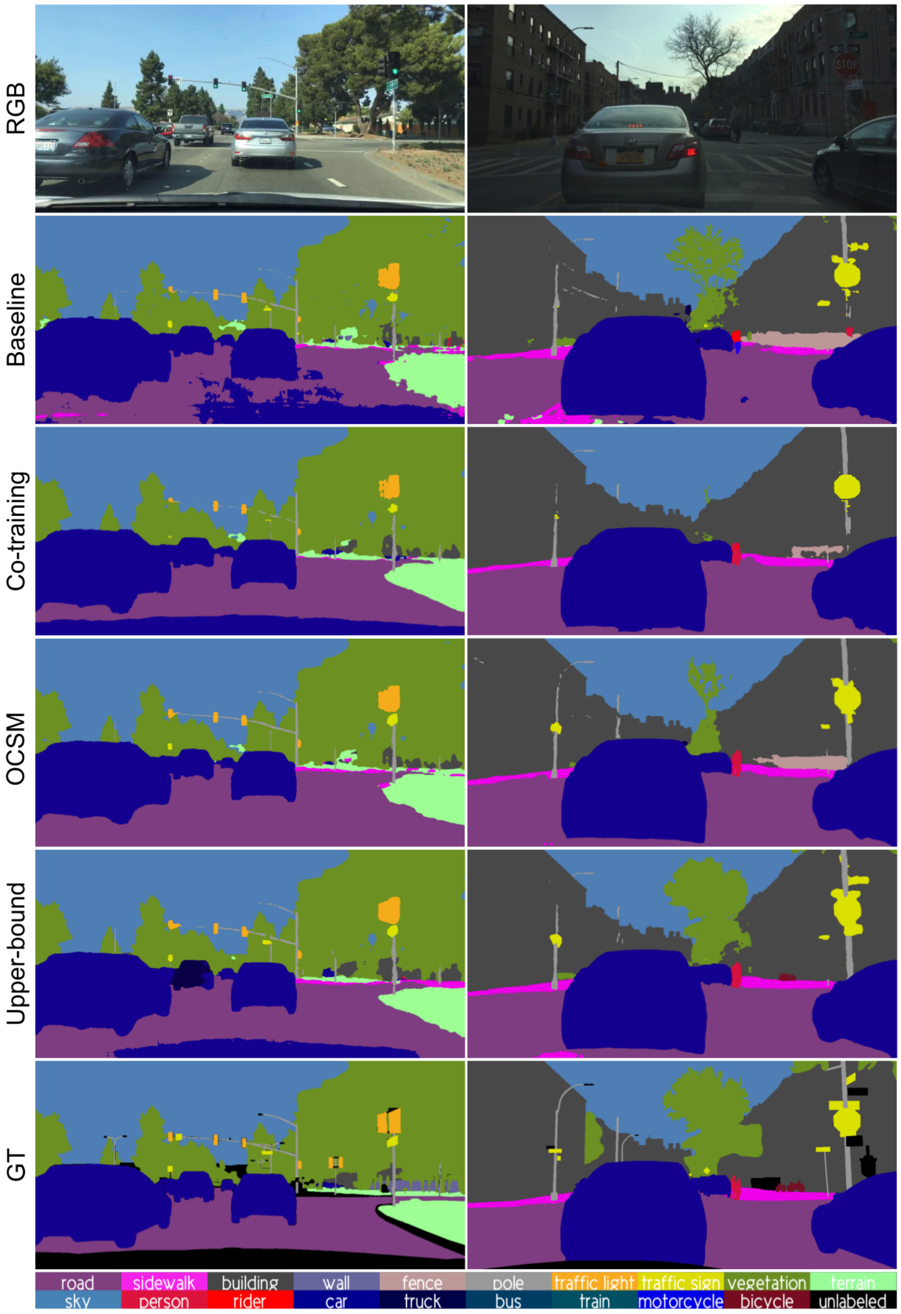


Figure 4.14: Analogous to Figure 4.13 on BDD100K validation samples.

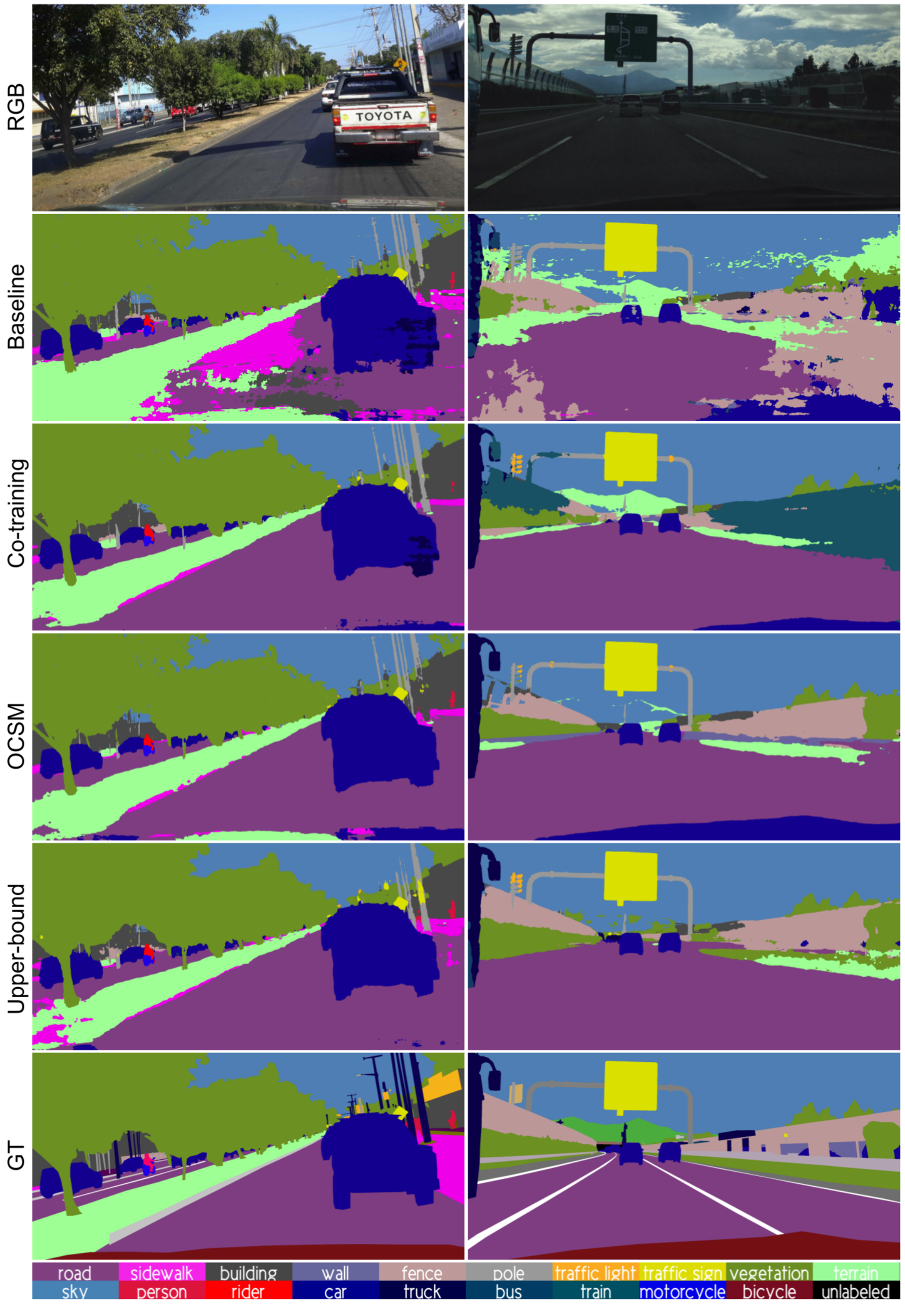


Figure 4.15: Analogous to Figure 4.13 on Mapillary Vistas validation samples.

Table 4.8: mIoU of the ordered composition of specialized models after the self-training step (OCSM (Self-t) using Synscapes + GTAV + Ours data as a source and the real-world datasets Cityscapes, BDD100K and Mapillary Vistas as a target. We added the analogous co-training results for a direct comparison.

	Background								Traffic			Human/cycles				Vehicles				
Model	Road	Sidewalk	Building	Wall	Fence	Vegetation	Terrain	Sky	Pole	Traffic light	Traffic Sign	Person	Rider	Motorbike	Bike	Car	Truck	Bus	Train	mIoU
S + G + O → Cityscapes																				
Co-training	95.59	67.95	91.24	56.26	53.16	90.04	48.74	93.96	60.22	64.84	70.27	79.44	57.56	59.71	73.84	94.38	70.03	89.42	64.66	72.70
OCSM (Self-t.)	96.05	71.83	90.25	47.50	50.56	90.67	53.75	93.00	62.08	68.93	74.71	79.73	58.19	60.95	74.26	94.22	77.27	87.91	72.00	73.89
Δ(OCSM vs. Co-t)	0.46	3.88	-0.99	-8.76	-2.60	0.63	5.01	-0.96	1.86	4.09	4.44	0.29	0.63	1.24	0.42	-0.16	7.24	-1.51	7.34	1.19
S + G + O → BDD100K																				
Co-training	92.31	51.73	80.15	23.31	46.37	70.42	37.78	85.92	46.20	31.72	37.29	66.04	49.52	52.56	54.32	86.55	44.37	72.04	1.36	54.21
OCSM (Self-t.)	92.53	50.87	81.79	20.98	42.13	72.31	39.21	86.93	51.40	49.73	45.69	65.79	49.06	54.79	61.69	87.67	46.62	67.56	0.00	56.15
Δ(OCSM vs. Co-t)	0.22	-0.86	1.64	-2.33	-4.24	1.89	1.43	1.01	5.20	18.01	8.40	-0.25	-0.46	2.23	7.37	1.12	2.25	-4.48	-1.36	1.94
S + G + O → Mapillary Vistas																				
Co-training	91.24	54.65	86.60	45.31	55.02	82.88	56.99	95.67	51.77	62.92	74.44	79.33	63.27	60.92	65.96	90.57	71.02	75.23	9.12	67.00
OCSM (Self-t.)	90.44	53.75	86.14	41.68	56.24	79.65	54.38	94.71	54.09	66.61	76.65	79.04	61.11	63.23	69.06	90.72	67.07	70.96	39.57	68.16
Δ(OCSM vs. Co-t)	-0.80	-0.90	-0.46	-3.63	1.22	-3.23	-2.61	-0.96	2.32	3.69	2.21	-0.29	-2.16	2.31	3.10	0.15	-3.95	-4.27	30.45	1.16

To conclude this line of experiments, Table 4.8 shows the results obtained with the OCSM method after the self-training step also on BDD100K and Mapillary Vistas as a target, using all the synthetic data. We compare them with the reported results of our co-training using the same source data. Overall, the OCSM improves by roughly 1-2 points over the co-training, being a notable improvement, seeing how close to the upper bounds are all the results. More in detail, the category Traffic (Pole, Traffic Lights, and Traffic Signs) is the most beneficial of our pseudo-labeling procedure because we prioritize it over the other categories due to its size during the composition process. In co-training, small classes tend to have lower confidence and be filtered by the thresholding step. Thus, pseudo-labels with these small classes are less accurate. Other benefited classes, analogous to the Traffic category, are Motorbike and Bike because these are objects relatively small and with certain shape complexity. The rest of the classes are a trade-off depending on the target dataset.

We can appreciate the aforementioned details in the qualitative analysis of Figures 4.13, 4.14 and 4.15, where class Pole is better represented in the OCSM case versus co-training. In general terms, the qualitative results show how the baseline models are noisy in certain background classes (e.g. Sky, Road, Sidewalks), but already segment properly foreground objects. It is difficult to find clear differences between co-training and OCSM, the performance differences are in tiny details, and difficult to appreciate qualitatively. Nevertheless, we selected a few helpful samples to appreciate several discrepancies between models. Figure 4.13 shows validation samples on Cityscapes, where OSCM refined certain details like Pole and Sidewalk and the results are similar to the upper-bound. Figure 4.14 shows the analogous case for BDD100K, we selected two different illuminated samples expecting differences, however, co-training and OCSM are solid and visually close to the upper bound. We can appreciate tiny differences in Traffic sign and Pole, where OCSM is better than co-training. Last case, Figure 4.15 on Mapillary Vistas shows how the baseline is noisier and how co-training solves successfully this noise, but introduces a problem with class Train on the sides, explaining the poor performance seen in this class. In contrast, OSCM responds close to the upper bound in all the cases.

4.6 Conclusions

In this chapter, we have proposed OCSM to address the synth-to-real UDA challenge. Moreover, by collaborating with a team of experts on the creation of synthetic environments, we have also introduced a new photo-realistic synthetic dataset which allows obtaining better baselines for semantic segmentation compared to publicly available datasets, both in isolation and combining all of them.

Our experimental setup revealed how a semantic segmentation model trained on our photo-realistic dataset outperforms the equivalent models trained on GTAV and Synscapes, when testing on the Cityscapes validation set. Moreover, for the same validation set, a baseline model trained according to the multi-source setting, i.e., combining our dataset with GTAV and Synscapes, even outperforms a large amount of UDA techniques that assume a single synthetic source (somehow ignoring today's data availability for synth-to-real UDA). In fact, these observations also apply when testing on BDD100K and Mapillary Vistas sets. Moreover, using the multi-source synthetic data and OCSM allows reaching unprecedented results in the synth-to-real UDA setting. Overall, in the multi-source scenario we improve our co-training results by up to ~3 mIoU points in the different target datasets and OCSM outperforms co-training by more than 1 mIoU points.

5 Conclusions

The use of synthetic data is widely extended to solve numerous computer vision tasks. Synthetic worlds allow configuring scenarios and moving on them to capture large amounts of data with all kinds of automatically generated labels. However, when training models using synthetic data which must later perform on real-world data, there is a significant drop on the accuracy of these models. This is known as the synth-to-real domain shift. Domain adaptation techniques (DA) are applied to overcome this problem. In general, in synth-to-real DA it is assumed that there are no labels for real-world data, so we face a specially challenging setting known as unsupervised domain adaptation (UDA). Overall, this has been the topic addressed in this Ph.D. dissertation since it is still an open problem. In particular, we focus on an application context where real-world data labeling is especially costly and time-consuming, namely, vision for autonomous driving and driver assistance. We address object detection and semantic segmentation tasks, which are core onboard tasks. Thus, the overall goal of this Ph.D. is the development of techniques to perform synth-to-real UDA for object detection and semantic segmentation. In particular, we have focused on leveraging and adapting ideas from the general semi-supervised learning (SSL) paradigm. As we are going to summarize, in Chapter 2 we proposed the SSL technique known as co-training for developing a multi-modal synth-to-real UDA procedure for object detection, in Chapter 3 we develop a co-training procedure for semantic segmentation, and in Chapter 4 we reach current state-of-the-art on synth-to-real UDA for semantic segmentation by the combination of a new photo-realistic dataset and a new SSL technique.

In Chapter 2 we focus on onboard object detection, i.e., the image labels we have to produce (pseudo-labels) are object bounding boxes (BBs). We assume that for any image we can use their appearance (RGB) and depth (D), the latter obtained by monocular depth estimation (MDE). Then, we address the synth-to-real UDA challenge following the idea of co-training, where two models collaborate to produce better pseudo-labels as they would do by working in isolation. Both models use synthetic data as prior knowledge, however, one of the models focuses on image appearance and the other on depth. Accordingly, in this chapter, the main goal is to answer these two questions: (Q1) Is multi-modal (RGB/D) co-training effective on the task of providing pseudo-labeled object BBs?; (Q2) How multi-modal (RGB/D) co-training does perform compared to single-modal (RGB) co-training? In order to answer the aforementioned questions we developed our research as follows:

- We generated the MDE view of our synthetic data (SYNTHIA dataset) and the real-world datasets (KITTI and Waymo datasets) from the RGB images using MDE. In addition, we performed GAN-based synth-to-real image translation of the synthetic data to approach their appearance to the corresponding real-world datasets.

- We designed and implemented a multi-modal co-training for object detection, treating the involved convolutional neural network (CNN) architectures as black boxes, so becoming a general procedure.

- We performed an exhaustive set of experiments using different proportions of human-labeled real-world data but without having synthetic data (standard SSL setting), and not having human-provided labels but having labeled synthetic data and unlabeled real-world data (UDA setting).

The obtained results indicate that multi-modal co-training is effective for labeling object BBs. Multi-modal co-training outperforms single-modal co-training in the standard SSL and the synth-to-real UDA settings. By performing GAN-based synth-to-real image translation both co-training modalities are on par, at least when using an off-the-shelf MDE not specifically trained on the translated images.

In Chapter 3 we focus on onboard semantic segmentation, where the image pseudo-labels produced consist of a class label per pixel. We designed and implemented a protocol following the co-training idea too. Thus, again, our synth-to-real UDA is purely data-driven, so treating the semantic segmentation model (CNN architecture) as a black box. Moreover, our method seamlessly leverages multiple synthetic datasets (source domains). We developed this research as follows:

- We designed and implemented a robust self-training procedure for synth-to-real UDA. In all cases, synthetic data is calibrated with real-world data in terms of LAB space, which turned out to be a simplified but effective synth-to-real appearance translation.

- We designed and implemented a co-training procedure that uses the previous self-training as the initial stage to provide two different views for co-training; i.e., two deep models which must collaborate following a co-training policy to provide the expected semantic pseudo-labels.

- Additional steps such as proper class balancing, domain mixes, and domain collages, have been also crucial to have effective self-training and co-training procedures.

- We performed an exhaustive set of experiments considering single-source and multi-source settings (SYNTHIA, GTAV, and Synscapes datasets) and different real-world target domains (Cityscapes, BDD100K, and Mapillary Vistas datasets). Ablation studies were carried out to validate the need for the different steps included in the overall co-training procedure.

The obtained results were state-of-the-art in the single- and multi-source settings, with improvements ranging from ~13 to ~31 mIoU points over baselines (mIoU ranges here from 0 to 100, the higher the better). The multi-source scenario combining GTAV + Synscapes achieved ~70 mIoU on the validation set of Cityscapes, just ~8 below the upper bound, which uses the synthetic data and the 100% of human-labeled data. For BDD100K and Mapillary Vistas there were no previous works reporting results. Moreover, the mentioned ablation studies confirmed that all the steps forming our co-training procedure are helping to improve the final results.

According to our experience in previous chapters, it was clear that improving the source-only baselines and performing some sort of muli-view/model collaborative self-training (co-training in our case), are good strategies to reach the labeling accuracy of upper bounds in the synth-to-real UDA problem. Accordingly, Chapter 4 focuses on a two-fold contribution. On the one hand, we have collaborated with a team of experts on simulation (programmers and technical artists) to design a new synthetic environment from which we have generated a new photo-realistic dataset to support the training of onboard visual models. This dataset complements previously existing ones, so eventually enabling better source-only baselines. On the other hand, we have designed and implemented a new SSL procedure, which is inspired by human labelers. Overall, it consists of an ensemble of category-specific self-trained models, which we call *ordered composition of specialized models* (OCSM). We developed this research as follows:

- As mentioned we collaborated in the generation of a photo-realistic dataset based on path-tracing rendering, proper 3D assets in terms of geometry and materials, and realistic illumination environments. As ground truth for semantic segmentation, we use the same semantic classes as Cityscapes.

- We group the semantic classes into categories. This is inspired by layer-based human labeling, where images are labeled per layer rather than fully at a time. Different human labelers can focus on different layers. Then, to have a fully labeled image, these layers are combined. In our case, a category is composed of a set of semantic classes to be labeled at a time, so labeling an image for a category has a layer of labels as a result.

- We rely on the self-training procedure developed in Chapter 3 to perform synth-to-real UDA per category. Our categories consider the usual size (in pixels) of the instances of the semantic classes in the onboard images, which can be considered a useful inductive bias that our proposal can leverage. In increasing size order, we define the following four categories: Traffic (Poles, Traffic lights, and Traffic signs), humans/cycles (Pedestrian, Rider, Bike, and Motorbike), Vehicles (Car, Truck, Bus, and Train), and Background (Road, Sidewalk, Building, Fence, Wall, Vegetation, Terrain, and Sky).

- After applying self-training to these categories, we run our OCSM policy to obtain fully labeled images. When self-training each category, we consider the class *unlabeled* to represent all the image content corresponding to classes not belonging to the category. When applying the OCSM policy, if a pixel can gather a class label different than *unlabeled* from different self-trained models, it prevails the label corresponding to the category of smaller size.

- We performed an exhaustive set of experiments to assess OCSM, considering single- and multi-source settings, including the new photo-realistic dataset. Other than this, the same source and target dataset used in Chapter 3 are considered in this chapter, so that we can also compare the obtained results to co-training.

The obtained results show that both our OCSM proposal and the new dataset help to establish the current state-of-the-art in the synth-to-real UDA setting for semantic segmentation. On the one hand, when comparing the new dataset with existing ones in a single-source setting, we see that it gives rise to more accurate baselines. On the other hand, when combined with others, it allows pushing the accuracy of such multi-source baselines by ~5 mIoU points. In this multi-source setting, the co-training method itself is able to improve ~2 mIoU points thanks to the new synthetic dataset. Finally, when replacing co-training by OCSM, we improve more than ~1 mIoU points, so establishing unseen performance in the synth-to-real UDA problem. In fact, since we are approaching the upper bound mIoU, improvements tend to be lower but significant.

Overall, in this PhD, we have developed SSL methods that have reached state-of-the-art accuracy in the synth-to-real UDA setting, both for object detection and semantic segmentation. We consider especially relevant the semantic segmentation task, which is critical for onboard perception in the context of autonomous driving and driver assistance. This is why we have also contributed to the creation of a new photo-realistic dataset. In fact, as future work, we want to keep working on the semantic segmentation task since we have not yet reached the upper bound accuracy in real-world target domains such as Cityscapes, BDD100K, and Mapillary Vistas datasets. The immediate actions must focus on incorporating better base deep models for semantic segmentation. In this Ph.D., we have used DeepLab V3+ for being one of the top-performing frameworks for semantic segmentation. However, new frameworks incorporating transformers (e.g., SegFormer [125]) should be tested. Note that performing automatic labeling of images in real-time on lightweight hardware is not a constraint, since the pseudo-labeled images can be used to train any semantic segmentation model. However, so far, the training of transformer-based semantic segmentation models has been too expensive for the hardware available to run this Ph.D. research.

Bibliography

[1] David Acuna, Huan Ling, Amlan Kar, and Sanja Fidler. Efficient interactive annotation of segmentation datasets with Polygon-RNN++. In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[2] Waqar Ahmed, Pietro Morerio, and Vittorio Murino. Cleaning noisy labels by negative ensemble learning for source-free unsupervised domain adaptation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1616–1625, 2022.

[3] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. SegNet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 39(12):2481–2495, 2017.

[4] Lorenzo Berlincioni, Federico Becattini, Leonardo Galteri, Lorenzo Seidenari, and Alberto Del Bimbo. Road layout understanding by generative adversarial inpainting. In Sergio Escalera, Stephane Ayache, Jun Wan, Meysam Madadi, Umut Güçlü, and Xavier Baró, editors, *Inpainting and Denoising Challenges*, 2019.

[5] Avrim Blum and Tom Mitchell. Combining labeled and unlabeled data with co-training. In *Conference on Computational Learning Theory (COLT)*, 1998.

[6] Walid Bousselham, Guillaume Thibault, Lucas Pagano, Archana Machireddy, Joe Gray, Young Hwan Chang, and Xubo Song. Efficient self-ensemble framework for semantic segmentation. *arXiv preprint arXiv:2111.13280*, 2021.

[7] Leo Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

[8] Yong Cao, Chunlei Huo, Nuo Xu, Xin Zhang, Shiming Xiang, and Chunhong Pan. Henet: Head-level ensemble network for very high resolution remote sensing images semantic segmentation. *IEEE Geoscience and Remote Sensing Letters*, 19:1–5, 2022.

[9] Chen-Hao Chao, Bo-Wun Cheng, and Chun-Yi Lee. Rethinking ensemble-distillation for semantic segmentation based unsupervised domain adaption. In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[10] O. Chapelle, B. Schölkopf, and A. Zien. *Semi-supervised learning*. The MIT Press, 2006.

[11] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. arXiv preprint arXiv:1706.05587, 2017.

[12] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *European Conference on Computer Vision (ECCV)*, 2018.

[13] Shuaijun Chen, Xu Jia, Jianzhong He, Yongjie Shi, and Jianzhuang Liu. Semi-supervised domain adaptation based on dual-level domain mixing for semantic segmentation. In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[14] Y. Chen, W. Li, C. Sakaridis, D. Dai, and L. van Gool. Domain adaptive Faster R-CNN for object detection in the wild. In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[15] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The Cityscapes dataset for semantic urban scene understanding. In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[16] Gabriela Csurka. *A comprehensive survey on domain adaptation for visual applications*, chapter 1. Advances in Computer Vision and Pattern Recognition. Springer, 2017.

[17] Gabriela Csurka, Riccardo Volpi, Boris Chidlovskii, et al. Semantic image segmentation: Two decades of research. *Foundations and Trends® in Computer Graphics and Vision*, 14(1-2):1–162, 2022.

[18] Raul de Queiroz Mendes, Eduardo Godinho Ribeiro, Nicolas dos Santos Rosa, and Valdir Grassi Jr. On deep learning techniques to boost monocular depth estimation for autonomous navigation. *Robotics and Autonomous Systems*, 136:103701, February 2021.

[19] A. Dosovitskiy, G. Ros, F. Codevilla, A.M. López, and V. Koltun. CARLA: an open urban driving simulator. In *Conference on Robot Learning (CoRL)*, 2017.

[20] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representation (ICLR)*, 2021.

[21] Gabriel Díaz, Billy Peralta, Luis Caro, and Orietta Nicolis. Co-training for visual object recognition based on self-supervised models using a cross-entropy regularization. *Entropy*, 23(4):423, 2021.

[22] Mark Everingham, SM Eslami, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The PASCAL visual object classes challenge: A retrospective. *International Journal of Computer Vision*, 111(1):98–136, 2015.

[23] David Forsyth and Jean Ponce. *Computer vision: A modern approach*. Prentice hall, 2011.

[24] Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139, 1997.

[25] A. Gaidon, Q. Wang, Y. Cabon, and R. Vig. Virtual worlds as proxy for multi-object tracking analysis. In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[26] Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. In *International Conference on Machine Learning (ICML)*, 2015.

[27] Li Gao, Jing Zhang, Lefei Zhang, and Dacheng Tao. DSP: Dual soft-paste for unsupervised domain adaptive semantic segmentation. In *ACM International Conference on Multimedia*, 2021.

[28] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the KITTI vision benchmark suite. In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[29] Jakob Geyer, Yohannes Kassahun, Mentar Mahmudi, Xavier Ricou, Rupesh Durgesh, Andrew S. Chung, Lorenz Hauswald, Viet Hoang Pham, Maximilian Mühlegg, Sebastian Dorn, Tiffany Fernandez, Martin Jänicke, Sudesh Mirashi, Chiragkumar Savani, Martin Sturm, Oleksandr Vorobiov, Martin Oelker, Sebastian Garreis, and Peter Schuberth. A2D2: Audi Autonomous Driving Dataset. *arXiv preprint arXiv:2004.06320*, 2020.

[30] Ross Girshick, Ilija Radosavovic, Georgia Gkioxari, Piotr Dollár, and Kaiming He. Detectron. https://github.com/facebookresearch/detectron, 2018.

[31] Clément Godard, Oisin Mac Aodha, Michael Firman, and Gabriel J. Brostow. Digging into self-supervised monocular depth estimation. In *International Conference on Computer Vision (ICCV)*, 2019.

[32] Akhil Gurram, Ahmet Faruk Tuna, Fengyi Shen, Onay Urfalioglu, and Antonio M. López. Monocular depth estimation through virtual-world supervision and real-world SfM self-supervision. arXiv:2103.12209, 2021.

[33] U. Guz, D. Hakkani-Tür, S. Cuendet, and G. Tur. Co-training using prosodic and lexical information for sentence segmentation. In *Conference of the International Speech Communication Association (INTERSPEECH)*, 2007.

[34] Jose L. Gómez, Gabriel Villalonga, and Antonio M. López. Co-training for deep object detection: Comparing single-modal and multi-modal approaches. *Sensors*, 21(9), 2021.

[35] Bo Han, Quanming Yao, Xingrui Yu, Gang Niu, Miao Xu, Weihua Hu, Ivor W. Tsang, and Masashi Sugiyama. Co-teaching: robust training of deep neural networks with extremely noisy labels. In *Neural Information Processing Systems (NeurIPS)*, 2018.

[36] Jianzhong He, Xu Jia, Shuaijun Chen, and Jianzhuang Liu. Multi-source domain adaptation with collaborative learning for semantic segmentation. In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[37] Kaiming He, Georgia Gkioxari, Piotr Dollar, and Ross Girshick. Mask R-CNN. In *International Conference on Computer Vision (ICCV)*, 2017.

[38] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[39] D. Hernandez, L. Schneider, A. Espinosa, D. Vázquez, A.M. López, U. Franke, M. Pollefeys, and J.C. Moure. Slanted stixels: representing San Francisco's steepest streets. In *British Machine Vision Conference (BMVC)*, 2017.

[40] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[41] Lukas Hoyer, Dengxin Dai, and Luc Van Gool. DAFormer: Improving network architectures and training strategies for domain-adaptive semantic segmentation. In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[42] Lukas Hoyer, Dengxin Dai, and Luc Van Gool. HRDA: Context-aware high-resolution domain-adaptive semantic segmentation. In *European Conference on Computer Vision (ECCV)*, 2022.

[43] Xiaoxia Hu, Xuefeng Liu, Zhenming He, and Jiahua Zhang. Batch modeling of 3d city based on esri cityengine. In *IET International Conference on Smart and Sustainable City 2013 (ICSSC 2013)*, 2013.

[44] Jiaxing Huang, Dayan Guan, Aoran Xiao, Shijian Lu, and Ling Shao. Category contrast for unsupervised domain adaptation in visual tasks. In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[45] Braden Hurl, Krzysztof Czarnecki, and Steven Waslander. Precise synthetic image and LiDAR (PreSIL) dataset for autonomous vehicle perception. In *Intelligent Vehicles Symposium (IV)*, 2019.

[46] Maksims Ivanovs, Kaspars Ozols, Artis Dobrajs, and Roberts Kadikis. Improving semantic segmentation of urban scenes for self-driving cars with synthetic images. *Sensors*, 22(6):2252, 2022.

[47] Jisoo Jeong, Seungeui Lee, , Jeesoo Kim, and Nojun Kwak. Consistency-based semi-supervised learning for object detection. In *Neural Information Processing Systems (NeurIPS)*, 2019.

[48] Matthew Johnson-Roberson, Charles Barto, Rounak Mehta, Sharath Nittur Sridhar, Karl Rosaen, and Ram Vasudevan. Driving in the matrix: Can virtual worlds replace human-generated annotations for real world tasks? *International Conference on Robotics and Automation (ICRA)*, 2017.

[49] Guoliang Kang, Lu Jiang, Yi Yang, and Alexander G. Hauptmann. Contrastive adaptation network for unsupervised domain adaptation. In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[50] Zhizhong Kang, Juntao Yang, Zhou Yang, and Sai Cheng. A review of techniques for 3d reconstruction of indoor environments. *International Journal of Geo-Information*, 9:330, 2020.

[51] Salman Khan, Hossein Rahmani, Syed Afaq Ali Shah, and Mohammed Bennamoun. A guide to convolutional neural networks for computer vision. *Synthesis lectures on computer vision*, 8(1):1–207, 2018.

[52] Samin Khan, Buu Phan, Rick Salay, and Krzysztof Czarnecki. Procsy: Procedural synthetic dataset generation towards influence factor studies of semantic segmentation networks. In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2019.

[53] Rawal Khirodkar, Brandon Smith, Siddhartha Chandra, Amit Agrawal, and Antonio Criminisi. Sequential ensembling for semantic segmentation. *arXiv preprint arXiv:2210.05387*, 2022.

[54] Alexander Kolesnikov, Xiaohua Zhai, and Lucas Beyer. Revisiting self-supervised visual representation learning. In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[55] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

[56] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[57] J. Li, C. Wang, H. Zhu, Y. Mao, H.-S. Fang, and C. Lu. CrowdPose: Efficient crowded scenes pose estimation and a new benchmark. In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[58] Kunming Li, Yu Li, Shaodi You, and Nick Barnes. Photo-realistic simulation of road scene for data-driven methods in bad weather. In *International Conference on Computer Vision (ICCV)*, 2017.

[59] W. Li, C. W. Pan, R. Zhang, J. P. Ren, Y. X. Ma, J. Fang, F. L. Yan, Q. C. Geng, X. Y. Huang, H. J. Gong, W. W. Xu, G. P. Wang, D. Manocha, and R. G. Yang. AADS: Augmented autonomous driving simulation using data-driven algorithms. *Science Robotics*, 4(28), 2019.

[60] Yunsheng Li, Lu Yuan, and Nuno Vasconcelos. Bidirectional learning for domain adaptation of semantic segmentation. In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[61] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *International Conference on Computer Vision (ICCV)*, 2017.

[62] Li Liu, Wanli Ouyang, Xiaogang Wang, Paul Fieguth, Jie Chen, Xinwang Liu, and Matti Pietikäinen. Deep learning for generic object detection: A survey. *International Journal of Computer Vision*, 128:261–318, 2020.

[63] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. Berg. SSD: single shot multibox detector. In *European Conference on Computer Vision (ECCV)*, 2016.

[64] Vishnu Suresh Lokhande, Songwong Tasneeyapant, Abhay Venkatesh, Sathya N. Ravi, and Vikas Singh. Generating accurate pseudo-labels in semi-supervised learning and avoiding overconfident predictions via Hermite polynomial activations. In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[65] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[66] Mingsheng Long, Han Zhu, Jianmin Wang, and Michael I Jordan. Unsupervised domain adaptation with residual transfer networks. In *Neural Information Processing Systems (NeurIPS)*, 2016.

[67] Yawei Luo, Liang Zheng, Tao Guan, Junqing Yu, and Yi Yang. Taking a closer look at domain shift: Category-level adversaries for semantics consistent domain adaptation. In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[68] Dimitrios Marmanis, Jan D Wegner, Silvano Galliani, Konrad Schindler, Mihai Datcu, and Uwe Stilla. Semantic segmentation of aerial images with an ensemble of cnss. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences, 2016*, 3:473–480, 2016.

[69] Tom M Mitchell and Tom M Mitchell. *Machine learning*, volume 1. McGraw-hill New York, 1997.

[70] Gerhard Neuhold, Tobias Ollmann, Samuel Rota Bulò, and Peter Kontschieder. The Mapillary Vistas dataset for semantic understanding of street scenes. In *International Conference on Computer Vision (ICCV)*, 2017.

[71] Le Thanh Nguyen-Meidine, Atif Belal, Madhu Kiran, Jose Dolz, Louis-Antoine Blais-Morin, and Eric Granger. Unsupervised multi-target domain adaptation through knowledge distillation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 1339–1347, January 2021.

[72] Viktor Olsson, Wilhelm Tranheden, Juliano Pinto, and Lennart Svensson. ClassMix: Segmentation-based data augmentation for semi-supervised learning. In *Winter conf. on Applications of Computer Vision (WACV)*, 2021.

[73] Fei Pan, Inkyu Shin, Francois Rameau, Seokju Lee, and In So Kweon. Unsupervised intra-domain adaptation for semantic segmentation through self-supervision. In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[74] Fabrizio J Piva and Gijs Dubbelman. Exploiting image translations via ensemble self-supervised learning for unsupervised domain adaptation. *arXiv preprint arXiv:2107.06235*, 2021.

[75] Koutilya PNVR, Hao Zhou, and David Jacobs. SharinGAN: Combining synthetic and real data for unsupervised geometry estimation. In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[76] Siyuan Qiao, Wei Shen, Zhishuai Zhang, Bo Wang, and Alan Yuille. Deep co-training for semi-supervised image recognition. In *European Conference on Computer Vision (ECCV)*, 2018.

[77] Can Qin, Lichen Wang, Yulun Zhang, and Yun Fu. Generatively inferential co-training for unsupervised domain adaptation. In *International Conference on Computer Vision (ICCV) Workshops*, 2019.

[78] Sayan Rakshit, Biplab Banerjee, Gemma Roig, and Subhasis Chaudhuri. Unsupervised multi-source domain adaptation driven by deep adversarial ensemble learning. In *German Conference on Pattern Recognition (GCPR)*, 2019.

[79] René Ranftl, Katrin Lasinger, David Hafner, Konrad Schindler, and Vladlen Koltun. Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 44(3):1623–1637, 2022.

[80] J. Redmon and A. Farhadi. YOLO9000: better, faster, stronger. In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[81] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[82] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[83] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: towards real-time object detection with region proposal networks. In *Neural Information Processing Systems (NeurIPS)*, 2015.

[84] S.R. Richter, Z. Hayder, and V. Koltun. Playing for benchmarks. In *International Conference on Computer Vision (ICCV)*, 2017.

[85] Stephan R. Richter, Vibhav Vineet, Stefan Roth, and Vladlen Koltun. Playing for data: Ground truth from computer games. In *European Conference on Computer Vision (ECCV)*, 2016.

[86] G. Ros, L. Sellart, J. Materzyska, D. Vázquez, and A.M. López. The SYNTHIA dataset: a large collection of synthetic images for semantic segmentation of urban scenes. In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[87] S. Roy, A. Unmesh, and V.P. Namboodiri. Deep active learning for object detection. In *British Machine Vision Conference (BMVC)*, 2018.

[88] Fatemeh Sadat Saleh, Mohammad Sadegh Aliakbarian, Mathieu Salzmann, Lars Petersson, and Jose M. Alvarez. Effective use of synthetic data for urban scene semantic segmentation. In *European Conference on Computer Vision (ECCV)*, 2018.

[89] Bhavani Sambaturu, Ashutosh Gupta, C.V. Jawahar, and Chetan Arora. Scribblenet: Efficient interactive annotation of urban city scenes for semantic segmentation. *Pattern Recognition*, 133:109011, 2023.

[90] Swami Sankaranarayanan, Yogesh Balaji, Arpit Jain, Ser Nam Lim, and Rama Chellappa. Learning from synthetic data: Addressing domain shift for semantic segmentation. In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[91] Robert E Schapire. The strength of weak learnability. *Machine learning*, 5(2):197–227, 1990.

[92] Ahmed Rida Sekkat, Yohan Dupuis, Varun Ravi Kumar, Hazem Rashed, Senthil Yogamani, Pascal Vasseur, and Paul Honeine. Synwoodscape: Synthetic surround-view fisheye camera dataset for autonomous driving. *Int. Conf. on Intelligent Robots and Systems (IROS)*, 2022.

[93] Ozan Sener, Hyun Oh Song, Ashutosh Saxena, and Silvio Savarese. Learning transferrable representations for unsupervised domain adaptation. In *Neural Information Processing Systems (NeurIPS)*, 2016.

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

[95] N. Sharma, V. Jain, and A. Mishra. An analysis of convolutional neural networks for image classification. *Procedia Computer Science*, 132:377–384, 2018.

[96] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *International Conference on Learning Representation (ICLR)*, 2015.

[97] C. Sun, A. Shrivastava, S. Singh, and A. Gupta. Revisiting unreasonable effectiveness of data in deep learning era. In *International Conference on Computer Vision (ICCV)*, 2017.

[98] K. Sun, B. Xiao, D. Liu, and J. Wang. Deep high-resolution representation learning for human pose estimation. In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[99] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, Vijay Vasudevan, Wei Han, Jiquan Ngiam, Hang Zhao, Aleksei Timofeev, Scott Ettinger, Maxim Krivokon, Amy Gao, Aditya Joshi, Yu Zhang, Jonathon Shlens, Zhifeng Chen, and Dragomir Anguelov. Scalability in perception for autonomous driving: Waymo open dataset. In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[100] Yunchao Tang, Mingyou Chen, Chenglin Wang, Lufeng Luo, Jinhui Li, Guoping Lian, and Xiangjun Zou. Recognition and localization methods for vision-based fruit picking robots: A review. *Frontiers in Plant Science*, 11:510, 2020.

[101] Y. Tian, X. Li, K. Wang, and F.-Y. Wang. Training and testing object detectors with virtual images. *IEEE/CAA Journal of Automatica Sinica*, 5(2):539–546, 2018.

[102] Marco Toldo, Andrea Maracani, Umberto Michieli, and Pietro Zanuttigh. Unsupervised domain adaptation in semantic segmentation: A review. *Technologies*, 8(2), 2020.

[103] Wilhelm Tranheden, Viktor Olsson, Juliano Pinto, and Lennart Svensson. DACS: Domain adaptation via cross-domain mixed sampling. In *Winter conf. on Applications of Computer Vision (WACV)*, 2021.

[104] I. Triguero, S. García, and F. Herrera. Self-labeled techniques for semi-supervised learning: Taxonomy, software and empirical study. *Signal Processing*, 42(2):245–284, 2015.

[105] Yi-Hsuan Tsai, Wei-Chih Hung, Samuel Schulter, Kihyuk Sohn, Ming-Hsuan Yang, and Manmohan Chandraker. Learning to adapt structured output space for semantic segmentation. In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[106] Gokhan Tur. Co-adaptation: Adaptive co-training for semi-supervised learning. In *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2009.

[107] Jesper E. van Engelen and Holger H. Hoos. A survey on semi-supervised learning. *Machine Learning*, 109:373–440, 2020.

[108] G. Varol, J. Romero, X. Martin, N. Mahmood, M.J. Black, I. Laptev, and C. Schmid. Learning from synthetic humans. In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[109] D. Vázquez, A.M. López, D. Ponsa, and J. Marin. Cool world: domain adaptation of virtual and real worlds for human detection using active learning. In *Neural Information Processing Systems (NeurIPS) Workshops*, 2011.

[110] Gabriel Villalonga and Antonio M. López. Co-training for on-board deep object detection. *IEEE Accesss*, 8:194441–194456, 2020.

[111] Jingdong Wang, Ke Sun, Tianheng Cheng, Borui Jiang, Chaorui Deng, Yang Zhao, Dong Liu, Yadong Mu, Mingkui Tan, Xinggang Wang, et al. Deep high-resolution representation learning for visual recognition. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 43(10):3349–3364, 2020.

[112] T. Wang, M. Liu, J. Zhu, A. Tao, J. Kautz, and B. Catanzaro. High-resolution image synthesis and semantic manipulation with conditional GANs. In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[113] Zhonghao Wang, Yunchao Wei, Rogerio Feris, Jinjun Xiong, Wen-Mei Hwu, Thomas S Huang, and Honghui Shi. Alleviating semantic-level shift: A semi-supervised domain adaptation method for semantic segmentation. In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2020.

[114] Zhonghao Wang, Mo Yu, Yunchao Wei, Rogerio Feris, Jinjun Xiong, Wen mei Hwu, Thomas Huang, and Honghui Shi. Differential treatment for stuff

and things: A simple unsupervised domain adaptation method for semantic segmentation. In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[115] WHO. Global status report on road safety. https://apps.who.int/iris/bitstream/handle/10665/277370/WHO-NMH-NVI-18.20-eng.pdf?ua=1, 2018.

[116] Garrett Wilson and Diane J. Cook. A survey of unsupervised deep domain adaptation. *ACM Transactions on Intelligent Systems and Technology*, 11(5):1–46, 2020.

[117] Patrick Henry Winston. *Artificial intelligence*. Addison-Wesley Longman Publishing Co., Inc., 1984.

[118] L. Woensel and G. Archer. Ten technologies which could change our lives: potential impacts and policy implication. European Parliamentary Research Service, 2015.

[119] Magnus Wrenninge and Jonas Unger. Synscapes: A photorealistic synthetic dataset for street scene parsing. arXiv preprint arXiv:1810.08705, 2018.

[120] Weimin Wu, Jiayuan Fan, Tao Chen, Hancheng Ye, Bo Zhang, and Baopu Li. Instance-aware model ensemble with distillation for unsupervised domain adaptation. *arXiv preprint arXiv:2211.08106*, 2022.

[121] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. https://github.com/facebookresearch/detectron2, 2019.

[122] Bernhard Wymann, Eric Espié, Christophe Guionneau, Christos Dimitrakakis, Rémi Coulom, and Andrew Sumner. Torcs, the open racing car simulator. *Software available at http://torcs. sourceforge. net*, 2000.

[123] Yu Xiang, Wongun Choi, Yuanqing Lin, and Silvio Savarese. Data-driven 3D voxel patterns for object category recognition. In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[124] Enze Xie, Peize Sun, Xiaoge Song, Wenhai Wang, Ding Liang, Chunhua Shen, and Ping Luo. PolarMask: Single shot instance segmentation with polar representation. In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[125] Enze Xie, Wenhai Wang, Zhiding Yu, Anima Anandkumar, Jose M Alvarez, and Ping Luo. SegFormer: Simple and efficient design for semantic segmentation with transformers. In *Neural Information Processing Systems (NeurIPS)*, 2021.

[126] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[127] Yanchao Yang and Stefano Soatto. FDA: Fourier domain adaptation for semantic segmentation. In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[128] D. Yarowsky. Unsupervised word sense disambiguation rivaling supervised methods. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 1995.

[129] Dewei Yi, Hui Fang, Yining Hua, Jinya Su, Mohammed Quddus, and Jungong Han. Improving synthetic to realistic semantic segmentation with parallel generative ensembles for autonomous urban driving. *IEEE Transactions on Cognitive and Developmental Systems*, 2021.

[130] Wei Yin, Yifan Liu, Chunhua Shen, and Youliang Yan. Enforcing geometric constraints of virtual normal for depth prediction. In *International Conference on Computer Vision (ICCV)*, 2019.

[131] Fisher Yu, Haofeng Chen, Xin Wang, Wenqi Xian, Yingying Chen, Fangchen Liu, Vashisht Madhavan, and Trevor Darrell. BDD100K: A diverse driving dataset for heterogeneous multitask learning. In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[132] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. *International Conference on Learning Representation (ICLR)*, 2016.

[133] Xingrui Yu, Bo Han, Jiangchao Yao, Gang Niu, Ivor W. Tsang, and Masashi Sugiyama. How does disagreement help generalization against label corruption? In *International Conference on Machine Learning (ICML)*, 2019.

[134] Yuhui Yuan, Xilin Chen, and Jingdong Wang. Object-contextual representations for semantic segmentation. *European Conference on Computer Vision (ECCV)*, 2020.

[135] Hang Zhang, Chongruo Wu, Zhongyue Zhang, Yi Zhu, Haibin Lin, Zhi Zhang, Yue Sun, Tong He, Jonas Mueller, R Manmatha, et al. Resnest: Split-attention networks. In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[136] Jingyi Zhang, Jiaxing Huang, Zichen Tian, and Shijian Lu. Spectral unsupervised domain adaptation for visual recognition. In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[137] Kai Zhang, Yifan Sun, Rui Wang, Haichang Li, and Xiaohui Hu. Multiple fusion adaptation: A strong framework for unsupervised semantic segmentation adaptation. In *British Machine Vision Conference (BMVC)*, 2021.

[138] Pan Zhang, Bo Zhang, Ting Zhang, Dong Chen, Yong Wang, and Fang Wen. Prototypical pseudo label denoising and target structure learning for domain adaptive semantic segmentation. In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[139] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[140] Shanshan Zhao, Huan Fu, Mingming Gong, and Dacheng Tao. Geometry-aware symmetric domain adaptation for monocular depth estimation. In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[141] Sicheng Zhao, Bo Li, Xiangyu Yue, Yang Gu, Pengfei Xu, Runbo Hu, Hua Chai, and Kurt Keutzer. Multi-source domain adaptation for semantic segmentation. In *Neural Information Processing Systems (NeurIPS)*, 2019.

[142] Kaiyang Zhou, Yongxin Yang, Yu Qiao, and Tao Xiang. Domain adaptive ensemble learning. *IEEE Transactions on Image Processing*, 30:8008–8018, 2021.

[143] Z.-H. Zhou and M. Li. Semi-supervised learning by disagreement. *Knowledge and Information Systems*, 24(3):415–439, 2010.

[144] J. Zhu, T. Park, P. Isola, and A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *International Conference on Computer Vision (ICCV)*, 2017.

[145] Xiaojin Jerry Zhu. Semi-supervised learning literature survey. 2005.

[146] Y. Zou, Z. Yu, B.V. Kumar, and J. Wang. Unsupervised domain adaptation for semantic segmentation via class-balanced self-training. In *European Conference on Computer Vision (ECCV)*, 2018.

[147] Yang Zou, Zhiding Yu, Xiaofeng Liu, B.V.K. Vijaya Kumar, and Jinsong Wang. Confidence regularized self-training. In *International Conference on Computer Vision (ICCV)*, 2019.