OBJECT DETECTION FOR AUTONOMOUS SYSTEMS OPERATING UNDER CHALLENGING CONDITIONS

By

Mazin Hnewa

A DISSERTATION

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

Electrical Engineering - Doctor of Philosophy

2023

ABSTRACT

Advanced Driver-Assistance Systems (ADAS) and autonomous systems, in general, such as emerging autonomous vehicles rely heavily on visual data and state-of-the-art deep learning approaches to classify and localize objects such as pedestrians, traffic signs and lights, and other nearby cars, to assist the corresponding vehicles maneuver safely in their environments. However, due to the well-known *domain shift* problem, the performance of object detection methods could degrade rather significantly under challenging scenarios such as low light and adverse weather conditions. The domain shift problem arises due to the difference between the distributions of source data used for training in comparison with target data used during realistic testing scenarios. The area of domain adaptation has been instrumental in addressing the domain shift problem encountered by many applications. In fact, domain adaptation frameworks for object detection methods have been providing powerful tools for handling a variety of underlying changes in probability distribution between training and testing data. In this dissertation, we first propose a novel integrated Generative-model based unsupervised training and Domain Adaptation (GDA) framework that improves the performance of a region-proposal based object detector under challenging scenarios. In particular, we exploit unsupervised image-to-image translation to generate annotated visuals that are representatives of a target challenging domain. Then, we use these generated annotated visuals in addition to unlabeled target domain data to train a domain adaptive region-proposal based object detector. We show that using this integrated approach outperforms both methods, unsupervised image translation, and domain adaptation, when they are used separately.

Despite the popularity of region-proposal based object detectors, such as Faster R-CNN and many of its variants, these detectors suffer from a long inference time. Therefore, such approaches are not the optimal choice for time-critical, real-time applications such as autonomous driving. As a result, in the second part of this dissertation, we propose a novel MultiScale Domain Adaptive

YOLO (MS-DAYOLO) framework for the popular state-of-the-art real time object detector YOLO. MS-DAYOLO employs multiple domain adaptation paths and corresponding domain classifiers at different scales of the recently introduced YOLOv4 object detector. Building on our baseline MS-DAYOLO architecture, we introduce three novel deep learning architectures for a Domain Adaptation Network (DAN) that generates domain-invariant features. In particular, we propose a Progressive Feature Reduction, a Unified Domain Classifier, and an Integrated architecture.

While RGB cameras represent the most popular imaging sensors used by ADAS systems and autonomous vehicles due to cost and related practical reasons, employing other modalities such as thermal and gated imaging sensors can significantly improve the detection performance under challenging conditions. However, these other types of sensors are expensive, and incorporating them into ADAS and autonomous vehicle platforms may cause design and manufacturing challenges. As a result, in the third part of this dissertation, we propose a new framework that utilizes Cross Modality Knowledge Distillation (CMKD) to improve the performance of RGB-only pedestrian detection in low light and adverse weather conditions without increasing computational complexity during inference. Specifically, we develop two CMKD methods that rely on feature-based knowledge distillation and adversarial training to transfer knowledge from a detector (teacher) that is trained using multiple modalities to a single modality detector (student) that is trained using RGB images only.

To validate the proposed approaches, we train and test them using popular datasets captured by vehicles driving under different conditions including challenging scenarios. Our experiments with the proposed approaches show significant improvements in object detection performance in comparison with state-of-the-art methods.

ACKNOWLEDGMENTS

First of all, I would like to express my deep gratitude to my advisor Professor Hayder Radha for offering me the great opportunity to join the Connected and Autonomous Networked Vehicle for Active Safety (CANVAS) research program at Michigan State University. I would like to thank him for his guidance, knowledge, and experience during my Ph.D. study. His support and encouragement made this work possible. Moreover, I would like to express my appreciation to my committee members: Professor Arun Ross, Professor Daniel Morris, and Professor Gary Bente for their comments and feedback.

In addition, I would like to acknowledge the funding support from Ford Motor Company that made this work possible. Specifically, I would like to thank Jon Diedrich, Alireza Rahimpour, and Justin Miller from Ford Motor Company for their support and encouragement.

Furthermore, I am grateful to my colleagues in WAVES lab: Daniel Kent, Su Pang, and Xiaohu Lu for their help, and the informative discussions we had in the lab.

Finally, a special thanks to my family for their patience, encouragement, and support during the challenging times I had in my study.

TABLE OF CONTENTS

Chapter 1

Introduction

1.1 Background

Visual data plays a critical role in enabling automotive Advanced Driver Assistance Systems (ADAS) and autonomous vehicles to achieve high levels of safety while maneuvering in their environments. Hence, emerging autonomous systems are employing cameras, and deep learning based methods for object detection [1, 2, 3]. These methods predict bounding boxes that surround detected objects as well as class probabilities associated with each bounding box. In particular, Convolutional Neural Network (CNN) based approaches have shown very promising results and achieved tremendous success in the detection of pedestrians, vehicles, and other objects [4, 5, 6, 7, 8, 9, 10].

In general, state-of-the-art CNN-based object detection models can be classified into two groups: one-stage and two-stage based methods. One-stage object detectors predict bounding boxes of objects and class probabilities associated with these objects directly from a full image in single computation via a unified neural network. Examples of well-known models of one-stage object detectors include YOLO [8, 11, 12, 13], SSD [9], and RetinaNet [7]. On the other hand, two-stage object detectors generate proposal bounding boxes that potentially have an object using Region Proposal Network (RPN) in the first stage. Then, the proposals are fed to a second stage where cropped features are used to classify objects and fine-tune the bounding boxes. The most well-

known models of two-stage object detectors are the R-CNN [4] series, including Fast R-CNN [5], Faster R-CNN [6], R-FCN [10], and Mask R-CNN [14].

These object detection models have been achieving exceedingly improved performance for object detection in terms of classifying and localizing a variety of objects in a scene [4, 5, 6, 9, 8, 7, 10]. However, under a domain shift, when the testing data has a different distribution from the training data distribution, the performance of state-of-the-art object detection methods drops noticeably and sometimes significantly. Such domain shift could occur due to capturing the data under different lighting or weather conditions, or due to viewing the same objects from different viewpoints leading to changes in object appearance and background. For example, an object detection method trained using a large amount of visual data captured by a vehicle driving in clear and favorable lighting conditions may degrade significantly if the same trained method is tested under realistic challenging conditions such as adverse weather or low light conditions. This is because the quality of the captured visual signals is impaired and distorted. In that context, the domain under which training is done is known as the *source domain* while the new domain under which testing is conducted is referred to as the *target domain*.

A straightforward solution to the domain shift problem is training with target domain data. However, annotated target domain data is unavailable, and it is expensive and time consuming to annotate data. An alternative solution is applying prepossessing methods to enhance the quality of the captured images [15, 16, 17, 18, 19]. However, the enhanced images may not lead to improve detection performance, and sometimes even make it worse [20, 21]. Another solution is to use other modalities (*e.g.* radar, lidar, thermal, and gated imaging) that improve the overall system perception capabilities. [22, 23, 24, 25, 26, 27, 28]. For example, Liao *et al.* [25] proposed a cross-collaboration enhancement strategy to learn robust object detection by fusing RGB and thermal

images. Furthermore, Chaturvedi *et al.* [28] fused data of lidar, RGB, and gated cameras at early and late stages by proposing a global-local attention framework to improve detection performance in adverse weather. In addition, Bijelic *et al.* [27] proposed deep feature exchange and adaptive fusion of four modalities (radar, lidar, RGB, and gated cameras) to develop robust multi-modal object detection. Although these approaches achieved some improvement, they are not feasible to support for many applications with constrained sensing modalities (*e.g.* systems that rely on RGB cameras only).

The lack of target domain data, and especially annotated data, led to the emergence of the area of *domain adaptation* [29, 30, 31, 32, 33, 34], which has been widely studied to solve the problem of domain shift without the need to annotate data for new target domains. In general, domain adaptation solutions have relied on an adversarial network and other strategies that are designed to generate domain-invariant features. In fact, they attempt to learn a robust object detector using labeled data from the source domain and unlabeled data from the target domain. Domain adaptation approaches for object detection can mainly be classified into reconstruction-based, and adversarial-based solutions [35]. Reconstruction based domain adaptation attempts to improve the performance of an object detector for target domain by using image-to-image translation models [36, 37, 38, 39, 40]. In particular, it utilizes image-to-image translation methods to generate artificial (fake) samples of the target domain from the corresponding source labeled samples. Consequently, translating labeled source data into corresponding target data will help in the training of an object detector in a target domain, and this should improve the performance of object detection in that domain.

In adversarial-based, a domain discriminator is trained to classify whether a data point is from the source or target domain. In contrast, the feature extractor of the object detector is trained

to confuse the domain discriminator [41]. Consequently, the feature extractor generates domain invariant features as a result of this training strategy. Many adversarial-based domain adaptation methods have been proposed for the Faster R-CNN object detector [42, 43, 44, 45, 46, 47, 48, 49, 50]. The state-of-the-art approach of adversarial-based domain adaptation is Domain Adaptive Faster R-CNN [42]. Subsequently, many other approaches were proposed. For example, Zhu *et al.*[43] proposed region mining and adjusted region-level alignment to develop a region-level adaptation. Wang *et al.*[44] proposed Few-shot Adaptive Faster R-CNN that required only a few target domain images with limited annotation. Saito *et al.*[45] combined strong local alignment with weak global alignment to develop an adaptive object detector. He and Zhang [46] proposed multiple adversarial submodules for both domain and proposal features alignment. Furthermore, Zhao *et al.*[49] proposed a collaborative self-training method that can propagate the loss gradient through the whole detection network, and mutually enhance the region proposal network and the region proposal classifier. In addition, Xu *et al.*[50] utilized elaborate prototype representations to achieve category-level domain alignment. In that context, we observe that domain adaptation has been studied rather extensively for the Faster R-CNN object detector, and its variants. However, other popular object detection schemes, and in particular YOLO-based architectures, have received little or no attention.

Furthermore, many previous works used Knowledge Distillation (KD) to compress an object detection model [51, 52, 53, 54, 55, 56]. Specifically, they attempted to transfer the learned knowledge of a complex model to a simpler and faster model. Chen *et al.* [51] proposed a weighted classification loss, and a bounded regression loss to distill the knowledge between detection models. Additionally, an adaptation layer is used to allow the simple student model to learn the distribution of features extracted from the more complex teacher model. Wang *et al.* [53] developed a fine-

grained feature imitation method that transfers knowledge via imitating features of local regions near objects. Moreover, Guo *et al.* [54] proposed a decoupled method that separates the features that have objects from features of background regions, and then applies KD to each of them individually to learn a better student detector. In summary, the main objective of these approaches is to speed up detectors during inference without a significant drop in performance by using KD techniques. Instead, in this dissertation, we exploit KD in a different manner to improve an RGB-based detector in low light and adverse weather conditions. To accomplish this, we distill and capture the learned knowledge of a multi-modal object detector into an RGB-based detector.

1.2 Contribution Summary

The main contribution of this dissertation can be summarized as follows:

- We propose an integrated Generative Domain Adaptation (GDA) that combines adversarial domain adaptation and image-to-image translation approaches in one framework for detecting objects under realistic rainy conditions. We present the performance of our proposed GDA for Faster R-CNN object detector as compared to other mitigation techniques including deraining, domain adaptation, and image-to-image translation using real driving data.

- We introduce a MultiScale Domain Adaptive YOLO (MS-DAYOLO) architecture that supports domain adaptation at different layers of the feature extraction stage within the YOLOv4 backbone network. To the best of our knowledge, this is the first proposed work that employs domain adaptation to improve the performance of YOLO for cross-domain object detection. The MS-DAYOLO architecture, including a Domain Adaptive Network (DAN) with multiscale feature inputs and multiple domain classifiers, represents our baseline architecture. Moreover, we pro-

pose three novel domain adaptation architectures that further improve YOLOv4 object detection performance when tested on challenging target data. These architectures are Progressive Feature Reduction, Unified Domain Classifier, and Integrated that combines the benefits of the other two architectures.

- we propose a novel framework that is based on Cross Modality Knowledge Distillation (CMKD) to improve the performance of RGB-based pedestrian detection in low light and adverse weather conditions. We achieve this by transferring the knowledge of a teacher detector that is trained using both RGB and gated images to a student detector, which is trained using RGB images only. The proposed CMKD framework makes the student model generate features that are similar to the features of the teacher model. To accomplish this, we develop two methods within the proposed CMKD framework. The first one is based on using a KD loss, while the second one incorporates adversarial training with knowledge distillation

- We conduct extensive experiments to train and test the proposed approaches using many popular datasets such as Cityscapes, KITTI, BDD100K, Waymo, and Seeing Through Fog datasets. These experiments show that the proposed frameworks provide significant improvements relative to the state-of-the-art approaches when tested on the target domain.

1.3 Dissertation Organization

The remainder of this dissertation is organized as follows:

- In Chapter 2, we first provide an insight into the impact of rain on two major classes of object detection frameworks. Then, we present a proposed Generative Domain Adaptation (GDA) framework in term of improving the performance of Faster R-CNN object detector under rainy

conditions. Furthermore, we compare its performance with the state-of-the-art techniques that represent leading candidates for mitigating the influence of rainy conditions on an autonomous system's ability to detect objects.

- In Chapter 3, we explain a novel MultiScale Domain Adaptive YOLO (MS-DAYOLO) framework that employs multiple domain adaptation paths and corresponding domain classifiers at different scales of the YOLOv4 object detector.

- In Chapter 4, we introduce a novel Cross Modality Knowledge Distillation (CMKD) framework that transfers the knowledge of a teacher detector that is trained using both RGB and gated images to a student detector, which is trained using RGB images only. We demonstrate that CMKD improves the performance of RGB-based pedestrian detection under challenging conditions by making the student model generate features that are similar to the features of the teacher model.

- Finally, in Chapter 5, we state the conclusion of this dissertation and some potential future work.

Chapter 2

Object Detection under Rainy Conditions

The quality of visual signals captured by autonomous vehicles can be impaired and distorted in adverse weather conditions, most notably under rain, snow, and fog. Such conditions minimize scene contrast and visibility, and this could lead to a significant degradation in the ability of the vehicle to detect critical objects in the environment. Depending on the visual effect, adverse weather conditions can be classified into: steady (such as fog, mist, and haze) and dynamic, which has more complex effects (such as rain and snow) [57].

In this chapter, we focus on rain because it is the most common dynamic challenging weather condition that impacts virtually every populated region of the globe. Furthermore, there has been a great deal of recent efforts that attempt to mitigate the effect of rain in the context of visual processing. While addressing the effect of other weather conditions has been receiving some, yet minimal attention, the volume of work regarding the mitigation of rain is by far more prevalent and salient within different research communities. It is worth noting that rain comprises of countless drops that have a wide range of sizes and complex shapes; and rain spreads quite randomly with varying speeds when falling on roadways, pavements, vehicles, pedestrians, and other objects in the scene. Moreover, raindrops naturally cause intensity variations in images and video frames. In particular, every raindrop blocks some of the light that is reflected by objects in a scene. In addition, rain streaks lead to low contrast and elevated levels of whiteness in visual data. Consequently, mitigating the effect of rain on visual data is arguably one of the most challenging tasks that

autonomous vehicles will have to perform due to the fact that it is quite challenging to detect and isolate raindrops, and it is equally difficult to restore the information that is lost or occluded by rain.

Meanwhile, there has been noticeable progress in the development of advanced visual deraining algorithms [58, 59, 60, 61, 62, 63]. Thus, one natural and intuitive solution for mitigating the effect of rain on active safety systems and autonomous vehicles is to employ robust deraining algorithms and then apply the desired object detection approach on the resulting derained signal. State-of-the-art deraining algorithms, however, are designed to remove the visual impairments caused by rain while attempting to restore the original signal with minimal distortion. Hence, the primary objective of these algorithms, in general, is to preserve the visual quality as measured by popular performance metrics, such as Peak-Signal-to-Noise-Ratio (PSNR) and structure similarity index (SSIM)[64]. However, these metrics do not reflect a viable measure for analyzing the system's performance for more complex tasks such as object detection.

The first objective of this chapter is to present a tutorial on state-of-the-art and emerging techniques that are leading candidates for mitigating the influence of rainy conditions on an autonomous vehicle's ability to detect objects. In that context, our goal includes analyzing the performance of object detection methods that are representatives of state-of-the-art frameworks, which are being considered for integration into autonomous vehicles' artificial intelligence (AI) platforms. Furthermore, we highlight the inherent limitations of leading deraining algorithms, deep learning based domain adaptation, and image translation frameworks in the context of rainy conditions, which are summarized in Figure 2.1. We present experimental results for applying the leading candidates of the used techniques with the objective of highlighting the urgent need for developing new paradigms for addressing the challenges of autonomous driving under severe

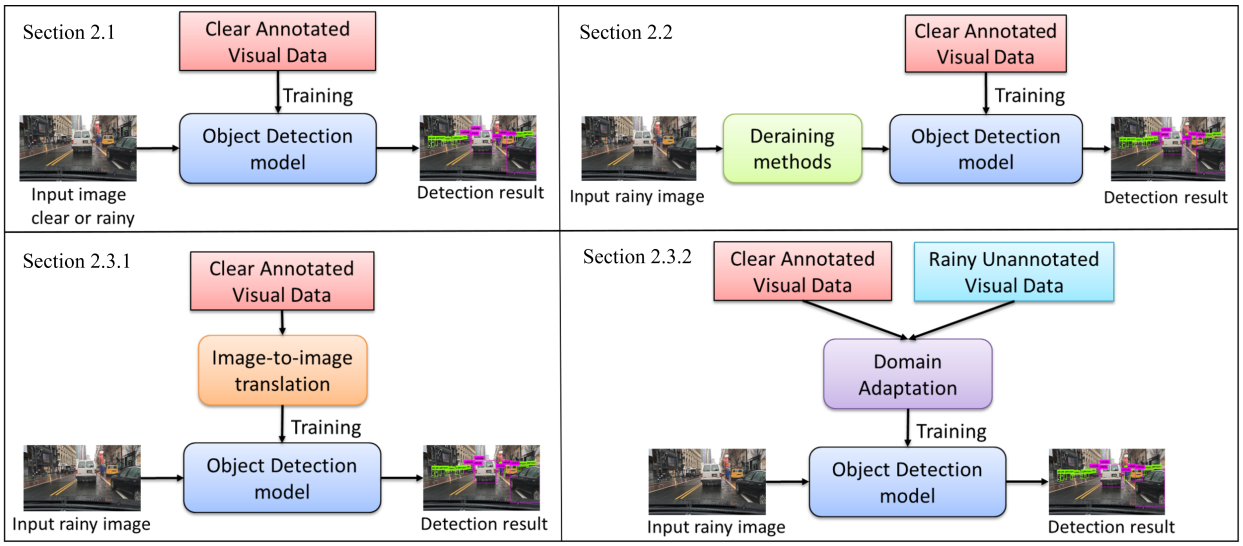


Figure 2.1: The frameworks highlighting the first three sections of this chapter.

weather conditions. Because generative model based image translation and domain adaptation approaches show some promising results, we propose a novel Generative Domain Adaptation (GDA) framework that combines generative model based image translation and domain adaptation [65].

2.1 Object detection for autonomous vehicles under clear and rainy conditions

The level of degradation in the performance of an object detection method, trained under certain conditions, is influenced heavily by: (a) How different the training and testing domains are; and (b) The type of deep learning based architecture used for object detection. Most recent object detectors are CNN based networks such as SSD [9], R-FCN [10], YOLO [8], RetinaNet [7], and Faster R-CNN[6]. To that end, we review two major classes of object detection frameworks that are both popular and representative of deep learning based approaches. As we will see later in this section,

these two classes of architectures exhibit different levels and forms of degradation in response to challenging rainy conditions, and they also perform rather differently in conjunction with potential rain mitigation frameworks. In particular, we briefly describe the underlying architectures for Faster R-CNN and YOLO as representatives of two major classes of object detection algorithms. Faster R-CNN is arguably the most popular among object detection algorithms that are based on a two-stage deep learning architecture, one stage is for identifying *region proposals* and the second stage is for refining and assigning class probabilities for the corresponding regions. On the other hand, YOLO is a representative of detection frameworks that operate on the whole image directly.

2.1.1 Deep learning based methods for object detection

The utility of Convolutional Neural Networks (CNNs) for object detection was well established prior to the introduction of the notion of *region proposals*, or commonly known as R-CNN [4], where the "R" stands for regions or region proposals. A fast version of R-CNN was later introduced [5], and then Ren *et al.* [6] introduced the idea of Region Proposal Network (RPN) that shares convolutional layers with Fast R-CNN [5]. RPN is merged with Fast R-CNN into one unified network that is known as *Faster R-CNN* to achieve more computationally efficient detection. Under Faster R-CNN, an input image is fed to a feature extractor such as the ZF model [66], or VGG-16 [67] to produce a feature map. Then, RPN utilizes this feature map to predict region proposals (regions in the image that could potentially contain objects of interest). In that context, many region proposals are quite overlapped with each other with significant numbers of pixels common among multiple region proposals. To filter out the substantial redundancy that might occur with such framework, Non-Maximum Suppression (NMS) [68] is used to remove redundant regions while keeping the ones that have the highest prediction scores. Subsequently, each regional proposal

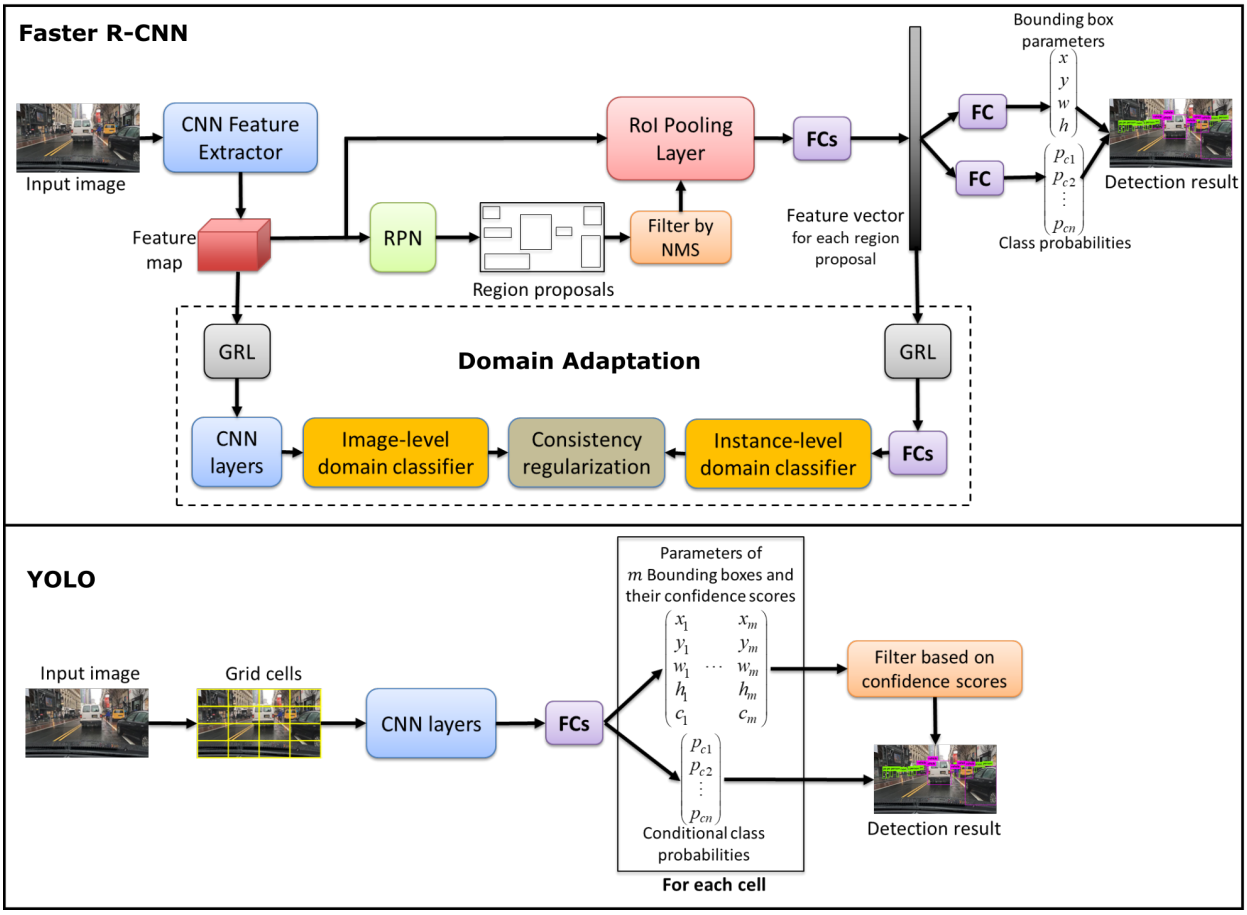


Figure 2.2: The high-level architectures of the detection methods that are used in this chapter. The domain adaptation of Faster R-CNN is explained in Section 2.3.2.

that survives the NMS process is used by a Region of Interest (RoI) pooling layer to crop the corresponding features from the feature map. This cropping process produces a feature vector that is fed to two fully connected layers: one layer predicts offset values of a bounding box of an object with respect to the regional proposal, and the other layer predicts class probabilities for the predicted bounding box. Figure 2.2 shows a high-level architecture of Faster R-CNN.

On the other hand, Redmon *et at.* [8] proposed to treat object detection as a regression problem, and they developed a unified neural network that is called *YOLO* (stands for You Only Look Once) to predict bounding boxes and class probabilities directly from a full image in one evaluation. Under YOLO, an input image is divided into a specific set of grid cells. Each cell is responsible

for detecting objects where their centers are located within that cell. To that end, each cell predicts a certain number of bounding boxes, and it also predicts the confidence scores for these boxes in terms of the likelihood that they contain an object. Furthermore, it predicts conditional class probabilities given it has an object of a particular class. In this case, there are potentially many wrongly predicted bounding boxes. To filter them out and provide the final detection result, a threshold is used on the confidence scores of the predicted bounding boxes. Figure 2.2 shows the general architecture of YOLO.

2.1.2 Object detection performance for neural network architectures under clear and rainy conditions

Here, we provide an insight into the level of degradation caused by rainy conditions on the performance of the two major deep learning architectures described above. In particular, we focus on the following fundamental question: how much degradation a deep neural network, which is trained under clear conditions, will suffer when tested under rainy weather. In that context, we first describe the dataset that we used for training and testing; and this is followed by presenting some visual and numerical results. As a result, we needed a rich dataset that is captured under diverse weather conditions. Despite the fact that there are few notable datasets [69, 70, 71], which are quite popular among the computer vision and AI research communities in terms of training deep neural networks, there is only one (arguably two [72][73]) that is properly labeled and annotated for our purpose, and hence, it could be used for training and testing for different weather conditions. In particular, we use the Berkeley Deep Drive dataset (BDD100K) [72] because it contains image tagging for the weather (i.e., each image in the dataset is labeled with its weather condition such as clear, rainy, foggy, etc). Meanwhile, although some other datasets, such as nuScenes [73],

might contain some visuals captured under rainy conditions, these datasets do not have weather tagging. Hence, choosing the BDD100k dataset was influenced by the fact that we can select images under a specific weather condition. Moreover, BDD100K has 100,000 video clips captured under diverse geographic, environmental, and weather conditions. It is worth noting that only one selected frame from each video is annotated with object bounding boxes as well as image-level tagging. Examples of annotated frames in clear and rainy weathers are shown in Figure 2.3. In this chapter, we consider the four classes (vehicle, pedestrian, traffic light, and traffic sign) that are labeled and provided as ground truth objects within the BDD100K dataset. Naturally, these four classes are among the most critical objects for an autonomous vehicle. In this chapter, we use 12454 images that are captured in clear weather from the designated training set of BDD100K to form our underlying training dataset. We refer to this training data as the *train clear* set, which we used consistently to train the detection methods for the different scenarios covered in this chapter. For testing, we use a set of clear weather images from the testing set of BDD100K. We refer to this later set as the *test clear* set. Table 2.1 shows the number of annotated objects in the *train clear* and *test clear* datasets.

One approach to demonstrate the impact of rain on object detection methods, which are trained under clear conditions, is by rendering synthetic rain [74, 75, 76] within the images of the *test clear* set. Then, the synthetic rainy data can be used to test the already trained object detection methods. The benefit of this approach is that one would have the exact same underlying content in both testing datasets in terms of the objects within the scene, but one set representing the original clear weather content when the data was captured, and another set with the synthetic rain. This would clearly show the impact of rain as the visual objects are the same in both tested sets (the *test clear* set and a *test synthetic rain* set). However, from our extensive experience in this area, we noticed

Figure 2.3: Examples of annotated images in BDD100K dataset [72]. Images in the top row are tagged as *clear weather*, and images in the middle and bottom rows are tagged as being captured in *rainy weather*. However, images in the bottom row are wrongly tagged as being rainy weather, but they are actually in clear or cloudy weather.

that most well-known rain simulation methods do not render realistic rain that viably captures actual and true rainy weather conditions, especially for a driving vehicle. Thus, when comparing the two scenarios, this discrepancy between synthetic and natural (real) rainy conditions will lead to domain mismatch. As a result, we do not test the detection methods using synthetic rain in our study because this will not demonstrate the impact of true natural rain on a driving vehicle.

Alternatively, we use images captured under real rainy conditions from the training and testing sets of the BDD100K dataset to test the object detection methods. It is worth noting that several images in the dataset are wrongly tagged as being ”rainy weather”, but they are actually in clear or cloudy weather such as the examples shown in the bottom row of Figure 2.3. To solve this problem, we manually select the images that are truly captured in rainy weather to form what we

Table 2.1: Number of annotated objects in training and testing sets that are used in our study.

Set	Vehicles	Pedestrians	Traffic signs	Traffic lights
Train clear	149,548	16777	43866	26002
Test clear	13721	2397	3548	4239
Test rainy	13724	2347	3551	4246

refer to as the *test rainy* set. Equally important, we elected to have both the *test clear* and *test rainy* sets approximately having the same number of annotated objects as shown in Table 2.1 in order to provide statistically comparable results.

It is important to make one final critical note regarding the currently available datasets for training neural networks designed for object detection. The lack of datasets captured under diverse conditions including rain, snow, fog, and other weather scenarios represents one of the most challenging aspects of achieving a viable level of training for autonomous vehicles. Even for the BDD100K dataset, which is one of very few publicly available datasets with properly annotated objects captured under different weather conditions, there is not sufficiently annotated visual content within BDD100K that is truly viable for *training* under rainy weather. This fundamental issue with the lack of real training data for rainy and other conditions has become clearly a major obstacle to the extent that leading high-tech companies working in the area have begun a focused effort designated specifically for collecting data under rainy conditions.

2.1.3 Performance Metric

To evaluate the performance of detection, we compute the mean Average Precision (mAP). This metric has been the most popular performance measure since the time when it was originally defined in the PASCAL Visual Object Classes Challenge 2012 for evaluating detection methods [77]. To determine mAP, the precision/recall curve is firstly computed based on the prediction

result against the ground truth. A prediction is considered a true positive if its bounding box: (a) has Intersection-over-Union (IoU) value greater than 0.5 relative to the corresponding ground truth bounding box, and (b) has the same class label as the ground truth. Then, the curve is updated by making precision monotonically decreasing. This is achieved by setting the precision for recall r to the maximum precision obtained for any recall $r' > r$. Average Precision (AP) is the area under the updated precision/recall curve, and it is computed by numerical integration. Finally, mAP is the mean of AP among all classes.

2.1.4 Results and Discussion

We trained the detection methods (Faster R-CNN and YOLO) using the *train clear* set, which is described in section 2.1.2. We used the same training settings and hyper-parameters that were used in the original papers [6][8]. Then, we tested the trained models using the *test clear* and *test rainy* sets to illustrate the impact of rain. Table 2.2 shows average precision (AP) for each class as well as the mean average precision (mAP) evaluated based on the AP values of the classes. From the table, we observe that mAP clearly declines in rainy weather as compared to clear weather using both Faster R-CNN and YOLO. Consequently, these results undoubtedly illustrate that the performance of an object detection framework, which is trained using clear visuals, could significantly degrade under rainy weather conditions. The performance decreases due to the fact that rain covers and distorts important details of the underlying visual features, which are used by detection methods to classify and localize objects. Figure 2.4 shows examples when the detection methods fail to detect most objects under rainy conditions.

Moreover, one can notice that under rainy conditions, the average precision for the pedestrian and traffic light classes declines more significantly than the decline in performance for vehicle and

Table 2.2: Quantitative results of the proposed GDA framework and other mitigation techniques for comparison, based on adaptation from clear to rainy weather of the BDD100K dataset. Average precision (AP) for each class, and mean average precision (mAP) evaluated based on the AP values of the classes. V: vehicle, P: pedestrian, TL: traffic light, and TS: traffic sign. *The top row shows the performance under clear conditions (i.e., using the *test clear* set), while all other rows show the performance under rainy conditions (i.e., using the *test rainy* set). **Significant degradation in performance can be observed due to rainy conditions (text in red) relative to the performance under clear conditions (top row). Improvements in performance by mitigating the effect of rain can be observed using generative model-based image translation and/or domain adaptation, and significant improvements can be achieved under the proposed GDA framework. Meanwhile, deraining algorithms do not improve, and most of the time further degrade the performance.

Mitigating technique	Faster R-CNN					YOLOv3				
	V	P	TL	TS	mAP	V	P	TL	TS	mAP
None(clear*)	72.61	40.99	26.07	38.12	44.45	76.57	37.12	46.22	50.56	52.62
None (rainy**)	67.84	32.58	20.52	35.04	39.00	74.15	32.07	41.07	50.27	49.39
Deraining DDN [60]	67.00	28.55	20.02	35.55	37.78	73.07	29.89	40.05	48.74	47.94
Deraining DeRaindrop [61]	64.37	29.27	18.32	33.33	36.32	70.77	30.16	37.70	48.03	46.66
Deraining PReNet [62]	63.69	24.39	17.40	31.68	34.29	70.83	27.36	35.49	43.78	44.36
Image translation UNIT [78]	**68.47**	32.76	18.85	36.20	39.07	74.14	**34.19**	**41.18**	48.41	**49.48**
Domain adaptation [42]	67.36	34.89	19.24	35.49	39.24	Not applicable				
The proposed GDA	68.04	**36.16**	20.29	**38.24**	**40.68**					

traffic sign classes. This discrepancy in performance degradation for different objects is due to a variety of factors. For example, vehicles usually occupy larger regions within an image frame than other types of objects; and hence, even when raindrops or rain streaks cover a visual of a vehicle, there are still sufficient features that can be extracted by the detection method. Furthermore, traffic signs are typically made from materials that have high reflectivity, which makes it easier for an object detection method to achieve higher accuracy even when a traffic sign visual is distorted with some rain. Overall, in both cases, the important features needed for reliable detection are still salient within the underlying deep neural networks of the detection algorithms. Nevertheless, rain still could impact the detection of vehicle and traffic signs as shown in the bottom three rows of

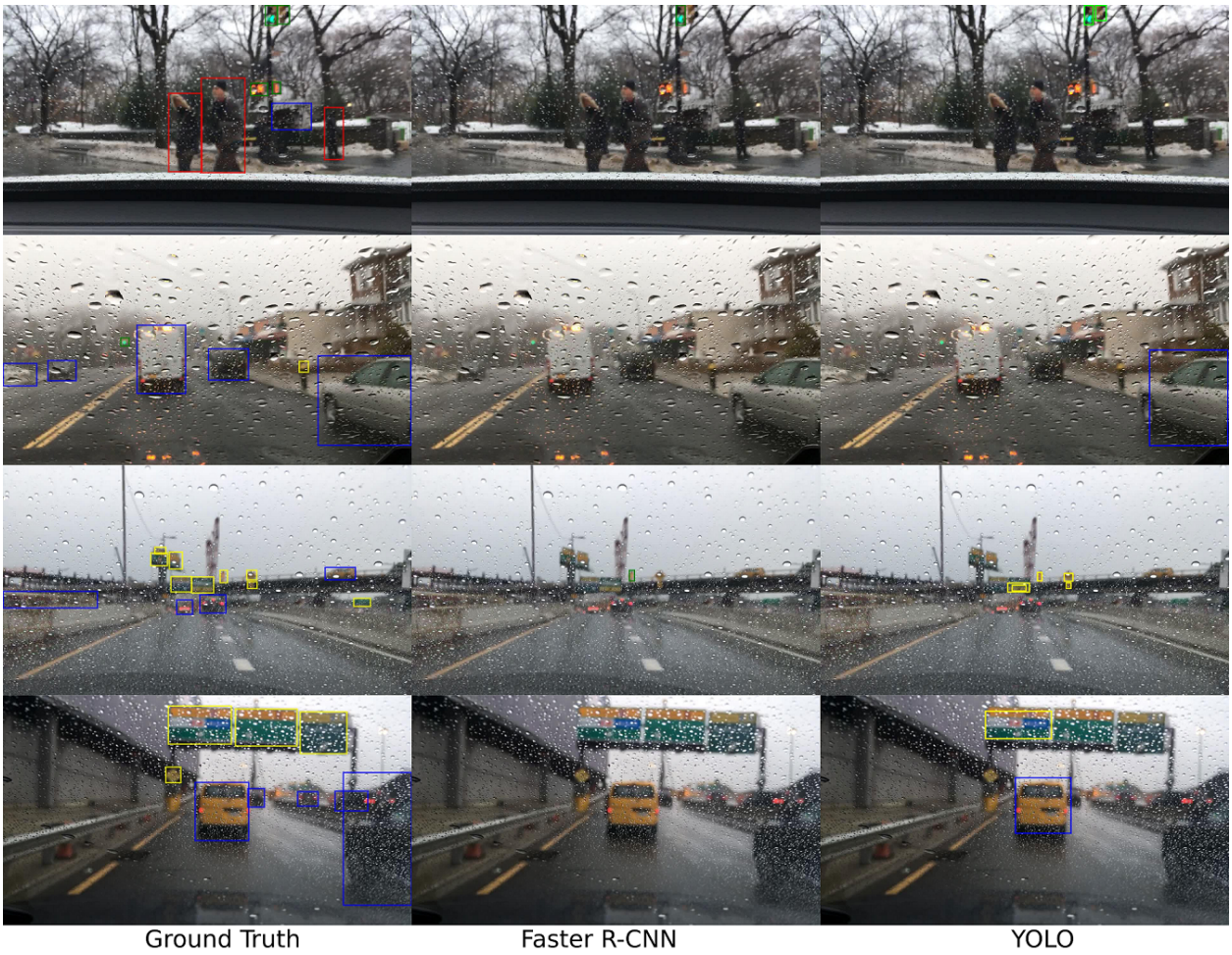


Figure 2.4: Examples of detection results using Faster R-CNN and YOLO for different visual scenes from *test rainy set*.

Figure 2.4.

2.2 Deraining in conjunction with object detection

Deraining methods attempt to remove the effect of rain and restore an image of a scene that has been distorted by raindrops or rain streaks while preserving important visual details. In this chapter, we review three recently developed deraining algorithms [60, 61, 62] that employ deep learning frameworks for the removal of rain from a scene. The high-level architectures of these methods

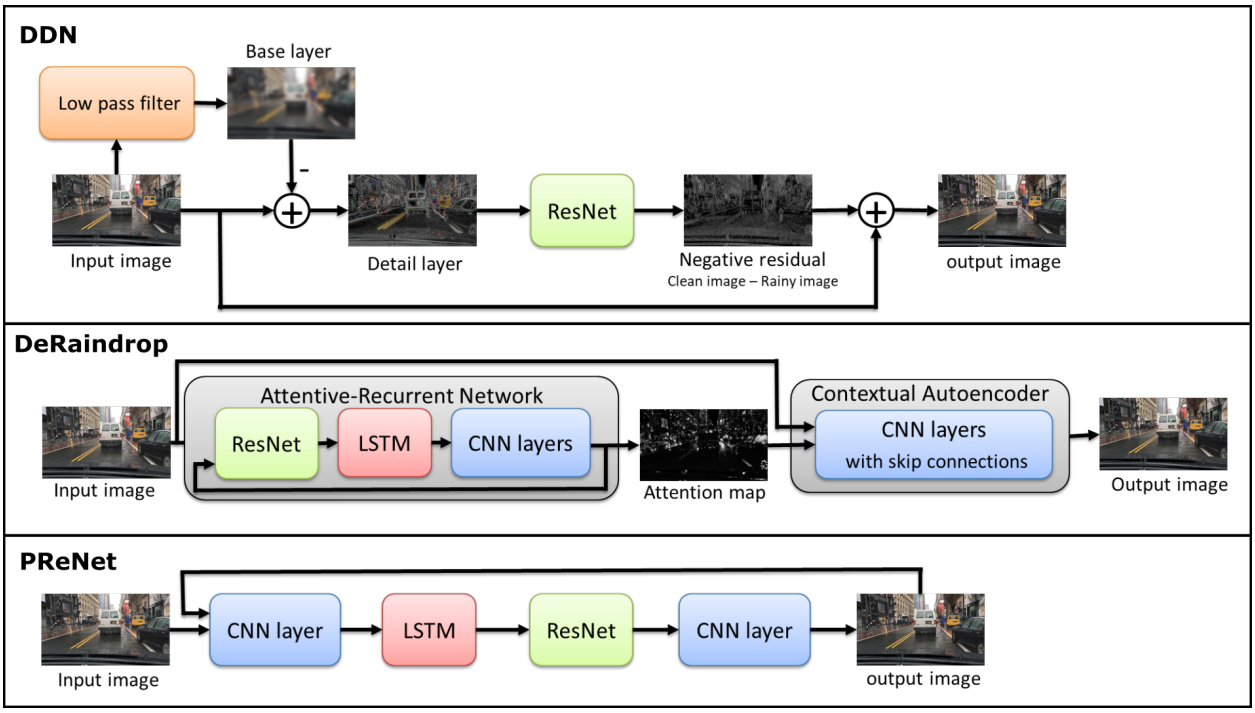


Figure 2.5: General architectures of the used deraining methods (DDN [60], DeRaindrop [61], and PReNet [62]).

are shown in Figure 2.5. Below, we briefly describe these three deraining methods and highlight their limitations when employing them in conjunction with object detection methods.

2.2.1 Deep Detail Network

Fu *et at.* [60] proposed a Deep Detail Network (DDN) to remove rain from a single image. They employed a convolutional neural network (CNN), which is ResNet [79] to predict the difference between clear and rainy images, and used this difference to remove rain from a scene. In particular, DDN exploits the rainy image's high-frequency details only, and it uses such details as input to ResNet while ignoring the low-frequency background (interference) of the same underlying scene.

2.2.2 Attentive Generative Adversarial Network

Qian *et at.* [61] proposed an attentive generative adversarial network that is called ”DeRaindrop” to remove raindrops from images. In this method, a generative adversarial network (GAN) [80] with visual attention is employed to learn raindrop areas and their surroundings. The first part of the generative network, known as the Attentive-Recurrent Network (ARN), produces an attention map to guide the next stage of the DeRaindrop framework. ARN includes ResNet, a Long Short-Term Memory (LSTM) [81], and CNN layers. The second stage of DeRaindrop, which is known as Contextual Autoencoder, operates on the attention map, and hence it focuses on (or ”pay more attention” to) the raindrop areas. The overall process from the two stages is expected to clean images free of raindrops. The architecture also includes a discriminative network, which assesses the generated rain-free images to verify that they are similar to real ones that have been used during the training process.

2.2.3 Progressive Image Deraining Network

Ren *et at.* [62] proposed a Progressive Recurrent Network (PReNet) to recursively remove rain from a single image. At each iteration, some rain is removed, and the remaining rain can be progressively removed in subsequent iterations. Consequently, after a certain number of iterations, most of the rain should be removed leading to a rain-free quality image. In addition to several residual blocks of ResNet, PReNet includes a CNN layer that operates on both the original rainy image and the current output image. PReNet also includes another CNN layer to generate the output image. Furthermore, a recurrent layer is appended to exploit dependencies of deep features across iterations via convolutional LSTM. To train PReNet, a single negative SSIM [64] or MSE loss is used.

2.2.4 Results and Discussion

To demonstrate the performance of the deraining methods outlined above, we apply the pretrained deraining models provided by the corresponding authors to *test rainy* set as a prepossessing step. After applying the deraining algorithms, which are anticipated to remove the rain from the visual input data and generate rain-free "clear" visuals, we feed the derained images into the object detection methods. Table 2.2 shows the performance of the detection methods after applying the deraining approaches. It can be seen that the deraining algorithms actually degrade the detection performance when compared to directly using the rainy images as input into the corresponding detection frameworks. This is true for both Faster R-CNN and YOLO. One important factor for this degradation in performance is that the deraining process tends to smooth out the input image, and hence, it distorts meaningful information and distinctive features of a scene while attempting to remove the effect of rain from the same scene. In particular, it is rather easy to observe that state-of-the-art deraining algorithms smooth out edges of objects in an image, which leads to a loss of critical information and features. Such information and features are essential for enabling detection algorithms to classify and localize objects. The images in the top two rows of Figure 2.6, which represent outputs of Faster R-CNN and YOLO, show some of the objects that are not detected after using the deraining methods, while they are successfully detected if rainy images are directly used as input into the detection algorithms.

A related critical issue to highlight about current deraining algorithms is their inability to remove natural raindrops found in realistic scenes captured by moving vehicles. The root cause of this issue is the fact that deraining algorithms have been largely designed and tested using synthetic rain visuals superimposed on the underlying scenes. What aggravates this issue is that, at least in some cases, the background environments used to design and test deraining algorithms are

Figure 2.6: Examples of detection results for different visual scenes without employing any deraining methods, and with employing deraining methods (DDN [60], DeRaindrop [61], and PReNet [62]) in conjunction with the detection methods. Objects were detected using Faster R-CNN [6] in the first group of images, and YOLO [12] in the second group.

predominantly static scenes with minimal moving objects. Consequently, the salient differences between such synthetic scenarios and the realistic environment encountered by a vehicle that is moving under natural rain represent a domain mismatch that is too much to handle by current deraining algorithms, and this leads to their failure under realistic conditions for autonomous vehicles. Hence, overall, we believe that relying purely on state-of-the-art deraining solutions does not represent a viable approach for mitigating the impact of rain on object detection. The images shown in Figure 2.6, especially some of the cases in the bottom two rows, illustrate examples of the failure of deraining methods to improve the performance of object detection.

2.3 Alternative training approaches for deep learning based object detection

The requirement that autonomous driving systems are expected to work reliably under different weather conditions, is at odds with the fact that the training data is usually collected in dry weather with good visibility. Thus, the performance of object detection algorithms degrades under challenging weather conditions as we showed in Section 2.1.4.

A straightforward approach to address this problem is to train a given CNN to detect objects using images captured in real rainy weather. As we highlighted earlier, sufficient annotated datasets captured by moving vehicles in realistic urban environments under natural rainy conditions are not readily available. To that end, and in spite of the fact that some datasets are available, the very few datasets captured under real rainy conditions are not properly annotated [72]. Having such small datasets inherently makes them inadequate to reliably train deep learning architectures for objection detection. Furthermore, annotating whatever available data captured under real rainy

conditions with accurate bounding boxes is an expensive and time-consuming process.

An alternative approach for addressing the lack of real data is training detection methods using synthetic rain data. However, as we highlighted earlier, the trained methods generalize poorly on real data due to the domain shift between synthetic and natural rain. To solve this issue, we review approaches that can be employed for training the detection methods using annotated clear data in conjunction with unannotated rainy data. In particular, we review two emerging frameworks for addressing this critical issue: image translation and domain adaptation.

2.3.1 Unsupervised Image-to-Image Translation

Image-to-Image Translation (I2IT) is a well-known computer vision framework that translates images from one domain (e.g., captured under clear weather) to another domain (e.g., rainy conditions) while preserving underlying and critical visual contents of the input images[82, 83, 84, 78, 85, 86, 87]. In short, I2IT attempts to learn a joint distribution of images in different domains. The learning process can be classified into: *supervised* setting where the training dataset consists of paired examples of the same exact scene captured in both domains (e.g., clear and rainy conditions), and *unsupervised* setting where different examples of both domains are used for training; hence, these examples do not have to be taken from the same corresponding scene. The unsupervised case is inherently more challenging than supervised learning. More importantly, to address the main issue we are facing in the context of the lack of data needed for training object detection architectures under realistic conditions, we consequently need an unsupervised setting. In particular, the requirement of having a very large set of image pairs, where each pair of images must be of the exact same scene captured under different domains, render supervised I2IT solutions virtually useless for our purpose. In fact, this requirement imposes more constraints than the lack of data

issue that we are already trying to address. Hence, and despite the availability of well-known supervised learning based techniques in this area [82, 88], we have to resort to unsupervised solutions to address the problem at hand.

Recently, Generative Adversarial Networks (GANs) [80] have been accomplishing promising results in the field of unsupervised I2IT [83, 84, 78, 85]. In general, a GAN consists of a *generator* and a *discriminator*. The generator is trained to *fool* the discriminator, while the latter is attempting to distinguish (or discriminate) between real natural images on one hand and fake images, which are generated by the trained generator, on the other hand. By doing this, GANs align the distribution of translated images with real images in the target domain.

As mentioned above, datasets that have paired clear-rainy images in a driving environment are not publicly available and difficult to collect. As a result, we use UNsupervised Image-to-image Translation (UNIT) [78] to translate clear images to rainy ones since the training process for the UNIT framework does not require paired images of the same scene. In other words, training UNIT requires two independent sets of images where one consists of images in one domain, and another consists of images in another domain. Moreover, the UNIT model has an excellent performance in the translation of images, which are captured by driving vehicles, from one domain to another.

The high-level architecture of the UNIT model is shown in Figure 2.7. The UNIT model consists of three networks: encoder, generator, and discriminator. First, the encoder maps an input image to a shared latent code (a shared compact representation of an image in both domains). Then, the generator uses the shared latent code to generate an image in the desired domain. Both encoder and generator are trained to *fool* the discriminator, while the latter aims to discriminate between real images and generated ones. By doing this, UNIT aligns the distribution of generated images with real ones in the target domain.

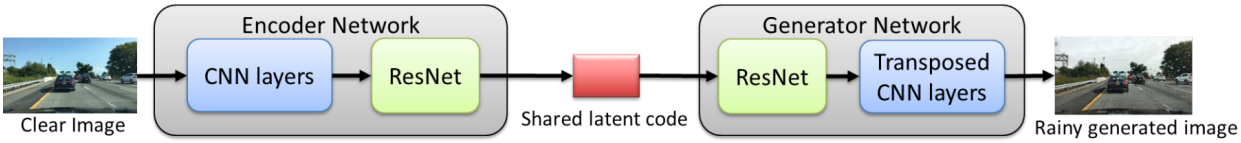


Figure 2.7: The high-level architecture of the UNIT model [78] to generate images.

Figure 2.8: Examples of generated images by the trained UNIT model, left: original clear images, right: generated rainy images, which have less domain shift with respect to real rainy images than the domain shift they exhibit with respect to original clear images.

To train the UNIT model that learns the mapping from clear images to rainy ones, we use the *train clear* set that consists of clear-weather annotated images as the source domain. For the target rainy domain, we extract a sufficiently large number of images from the rainy videos of the BDD100K dataset. Subsequently, we apply images in the *train clear* set to the trained UNIT model to generate rainy images. We refer to the images that are generated by the UNIT model as the *train gen rainy* set. Examples of generated rainy images are shown in Figure 2.8.

Eventually, we use the *train gen rainy* dataset to train the detection methods. This is followed

by using the *test rain* dataset to evaluate the average precision performance of the detection methods, which are now trained using the generated rainy set. We also calculate the mean average precision (mAP) as we have done for other approaches. Table 2.2 shows the performance of detection methods that are trained using generated rainy images by the UNIT model.

2.3.2 Domain Adaptation

Domain adaptation is another potentially viable framework that could be considered to address the major challenges that we have been highlighting in this chapter regarding: (a) the salient mismatch between the two domains, clear and rainy weather conditions, and (b) the lack of annotated training data captured under rainy conditions. The area of domain adaptation (DA) [31, 89, 32, 33, 90, 34] addresses a fundamental and practical problem that is related to the availability of annotated training data. In particular, there is usually sufficient annotated training data captured under the source domain, but a significantly smaller amount of annotated data for the target domain. For example, there is an abundance of annotated data available for vehicles driving under clear weather. However, and comparatively speaking, there is much less data, and especially annotated data, captured by vehicles driving under severe weather conditions.

DA training is designed to compensate for the shift in probability distribution between the two domains while utilizing labeled source domain data and unlabeled target domain data. In general, the convolutional neural network (CNN) stage of a domain adaptive object detector is trained to generate domain-invariant features that can be used by the detection stage regardless of whether the test is being conducted under the source or target domains.

Domain Adaptation (DA) has been used in deep learning-based object detectors to *adapt* them to a new desired *target* domain that is dissimilar from the original training domain, without the

need to annotate visual data of the target domain [42, 44, 45, 46, 91]. Most domain adaptation approaches in literature utilize the adversarial training approach [80] to produce robust domain-invariant features. In particular, a domain classifier is optimized to identify whether a data point is from the source or target domain. Meanwhile, the feature extractor of the object detector is optimized to generate feature maps that are indistinguishable, regardless of the domain, in order to confuse the domain classifier. This strategy makes the feature extractor learn to produce domain invariant features. In other words, the distribution of features extracted from images becomes indistinguishable in both domains.

In particular, a domain adaptation framework [42] has been designed and developed specifically for Faster R-CNN due to the fact that it is among the most popular object detection approaches. The framework developed in [42] adapts deep learning based object detection to a target domain that is different from the training domain without requiring any annotations in the target domain. In particular, it employs the adversarial training strategy [80] to learn robust features that are domain-invariant. In other words, it makes the distribution of features extracted from images in the two domains indistinguishable.

The architecture for Domain Adaptive Faster R-CNN model [42] is shown in Figure 2.2. There are two levels of domain adaptation that are employed. First, an *image-level* domain classifier is used. At this level, the global attributes (such as the image style and illumination) of the input image are used to distinguish between the source and target domains. Thus, the (global) feature map resulting from the common CNN feature extractor of the Faster R-CNN detector is used as input toward the image-level domain classifier. Second, an *instance-level* domain classifier is employed to reduce the local domain shift occurring to an object (such as its appearance, viewpoint, and size). This classifier uses the specific features associated with a particular region to distinguish

between the two domains. Hence, the instance-level domain classifier uses the feature vector resulting from the fully connected layers (FCs) at the output of the RoI Pooling Layer of the Faster R-CNN detector. The two classifiers, the image- and instance-level ones, should naturally agree in terms of their binary classification decision regarding if the input image belongs to the source or target domain. Consequently, a *consistency regularization* stage combines the output of the two classifiers to promote consistency between the two classifier outcomes.

Domain Adaptation Network (DAN) is attached to the original object detection network during training only. DAN has three components: image-level adaptation, instance-level adaptation, and consistency regularization. This DAN includes several layers for predicting the domain class (either source or target) for both levels. Then, domain classification losses are computed via cross-entropy for image-level and instant level as follows:

$$\mathcal{L}_{img} = -\sum_{i,x,y}[t_i \ln p_i^{(x,y)} + (1-t_i)\ln(1-p_i^{(x,y)})] \tag{2.1}$$

where t_i is the ground truth domain label for i-th training image, with $t_i = 1$ for source domain and $t_i = 0$ for target domain. $p_i^{(x,y)}$ is predicted image-level domain class probability for i-th training image at location (x, y) of the feature map.

$$\mathcal{L}_{ins} = -\sum_{i,j}[t_i \ln p_{i,j} + (1-t_i)\ln(1-p_{i,j})] \tag{2.2}$$

Here, $p_{i,j}$ is the predicted instance domain class probability for j-th region proposal that may have an object in i-th training image.

The domain classifiers of these two adaptation levels need to produce the same classification prediction about the domain of the input image, which may be from the source or target domain.

Therefore, to impose consistency between outcomes of the two domain classifiers, a *consistency regularization* stage is used to join the output of the two classifiers as follows:

$$\mathcal{L}_{cst} = \sum_{i,j} ||\frac{1}{N} \sum_{x,y} p_i^{(x,y)} - p_{i,j}||_2 \tag{2.3}$$

where N is the total number of activations in a feature map. Therefore, the total loss of DAN can be written as follows:

$$\mathcal{L}_{DAN} = \mathcal{L}_{img} + \mathcal{L}_{ins} + \mathcal{L}_{cst} \tag{4}$$

While the two domain adaptation classifiers are optimized to differentiate between the source and target domains, the Faster R-CNN detector must be optimized such that it becomes domain-independent or *domain-invariant*. In other words, the Faster R-CNN detector must detect objects regardless of the input image domain (clear or rainy). Hence, the feature map resulting from the Faster R-CNN feature extractor must be domain-invariant. To that end, this feature extractor should be trained and optimized to *maximize* the domain classification error achieved by the domain adaptation stage. Thus, while both the image- and instance-level domain classifiers are designed to minimize the binary-classification error (between the source and target domains), the Faster R-CNN feature extractor is designed to maximize the same binary-classification error. To achieve these contradictory objectives, a Gradient Reversal Layer (GRL) [34, 92] is employed. Thus, GRL is a bidirectional operator that is used to realize two different optimization objectives. In the feed-forward direction, the GRL acts as an identity operator. This leads to the standard objective of minimizing the classification error when performing *local* backpropagation within the domain adaptation network. On the other hand, for backpropagation toward the Faster R-CNN network, the GRL becomes a negative scalar. Hence, in this case, it leads to maximizing

the binary-classification error; and this maximization promotes the generation of domain-invariant feature map by the Faster R-CNN feature extractor.

Consequently, we developed and employed a domain adaptive faster R-CNN [42] under rainy conditions. To train this model, we prepare the training data to include two sets: *source data*, which consists of images captured in clear weather (and this set includes data annotations in terms of bounding boxes coordinates and object categories), and *target data*, which only consists of images captured under rainy conditions without any annotations. To validate the trained model using domain adaptation, we tested it using the *test rainy* set. The performance of the detection method (Faster R-CNN) that is trained by the domain adaptation approach is shown in Table 2.2.

2.3.3 Results and Discussion

Based on the results in Table 2.2, we observe that while deraining algorithms degrade the average precision performance when tested on scenes distorted by natural rain, improvements can be achieved when employing image-to-image translation and domain adaptation as mitigating techniques. Different cases are shown in Figure 2.9.

In terms of average precision, and as an example, rainy conditions degrade the pedestrian detection capabilities for YOLO by more than five percent (from around 37 to around 32); but by using image translation, the performance is improved to an average precision of more than 34, and consequently, narrowing the gap between clear and rainy conditions' performances. Similarly, both image translation and domain adaptation improve the traffic-signs detection performance for Faster R-CNN. Furthermore, image translation seems to improve the vehicle detection performance under Faster R-CNN.

In other cases, for example, traffic-light detection performance under Faster R-CNN, domain

Figure 2.9: Examples of detection results using alternative training approaches for Faster R-CNN and YOLO are shown in the rightmost column. The top-right image shows improvement in vehicle and traffic sign detection when generated images by I2IT (the UNIT model [78]) are used to train Faster R-CNN. The middle-right image shows improvement in pedestrian detection when domain adaptation [42] is used to train Faster R-CNN. The bottom-right image shows improvement in pedestrian detection when generated images by I2IT (the UNIT model [78]) are used to train YOLO.

adaptation, and image translation do not seem to perform well when tested on natural rainy images (even when using natural rainy images as the target domain for training these techniques). One potential factor for this poor performance in some of these cases is the fact that small objects such as traffic lights are quite challenging to detect to start with. This can be seen from the very low numbers of average precision, even under clear conditions, which is a mere 26%. Naturally, the impact of raindrops or rain streaks on such small objects in the scene could be quite severe to the extent that a mitigating technique might not be able to recover the salient features of these objects.

In summary, employing domain adaptation or generating rainy-weather visuals using unsuper-

vised image-to-image translation, and then using these visuals for training seems to narrow the gap in performance due to the domain mismatch between clear and rainy weather conditions. This promising observation becomes especially clear when considering the disappointing performance of deraining algorithms. Nevertheless, it is also clear that there is still much room for improvements toward reaching the performance under clear conditions. There are key challenges that need to be addressed, though, when designing any new mitigating techniques for closing the aforementioned gap. These challenges include the broad and diverse scenarios for "rainy conditions", especially in driving environments. These diverse cases and scenarios cannot be learned in a viable way by using state-of-the-art approaches. For example, raindrops have a wide range of possible appearances, and they come in various sizes and shapes, especially when falling on the windshield of a vehicle. Another factor is the influence of windshield wipers on altering the amount, and even shapes and sizes of raindrops, in-between wipe cycles. Other external factors include reflections from the surrounding wet pavement, mist in the air, and splash effects. Hence, current state-of-the-art image translation techniques and domain adaptation are not robust enough to capture this wide variety of rain effects. Figure 2.10 shows images from the *test rainy* set, where these examples illustrate several scenarios and effects of rainy weather for driving vehicles.

2.4 Integrated Generative-Model Domain-Adaptation

In this section, we address the problem of object detection of objects in a driving environment under rainy conditions using a novel Generative Domain Adaptation (GDA) framework that combines: (a) Generative-model based image translation and (b) Domain Adaptation.

Figure 2.10: Images from test-rainy set that illustrate several scenarios and effects of rainy weather for driving vehicles.

2.4.1 Proposed GDA framework

We exploit unsupervised image-to-image translation to generate visuals that are representatives of a challenging target domain, and then we use these generated visuals in addition to unlabeled target domain data to train a domain adaptive object detection method. A high-level architecture of the proposed (GDA) framework is shown in Figure. 2.11. We show that using this novel integrated approach outperforms both methods, unsupervised image translation, and domain adaptation, when they are used separately.

In Section 2.3.1, we show that the trained UNIT model generates visuals that reduce the domain shift between the source and target domains. This will help to increase the performance of domain adaptive object detection for the target domain. Therefore, in this section, we employ the UNIT model as a GAN-based unsupervised I2IT method that we integrate with our GDA framework.

On the other hand, if the domain shift between source and target domain is significant, especially under different weather conditions, a domain adaptive model fails to improve detection in the target domain. Because the generated visuals of the developed I2IT model in section 2.3.1 reduce

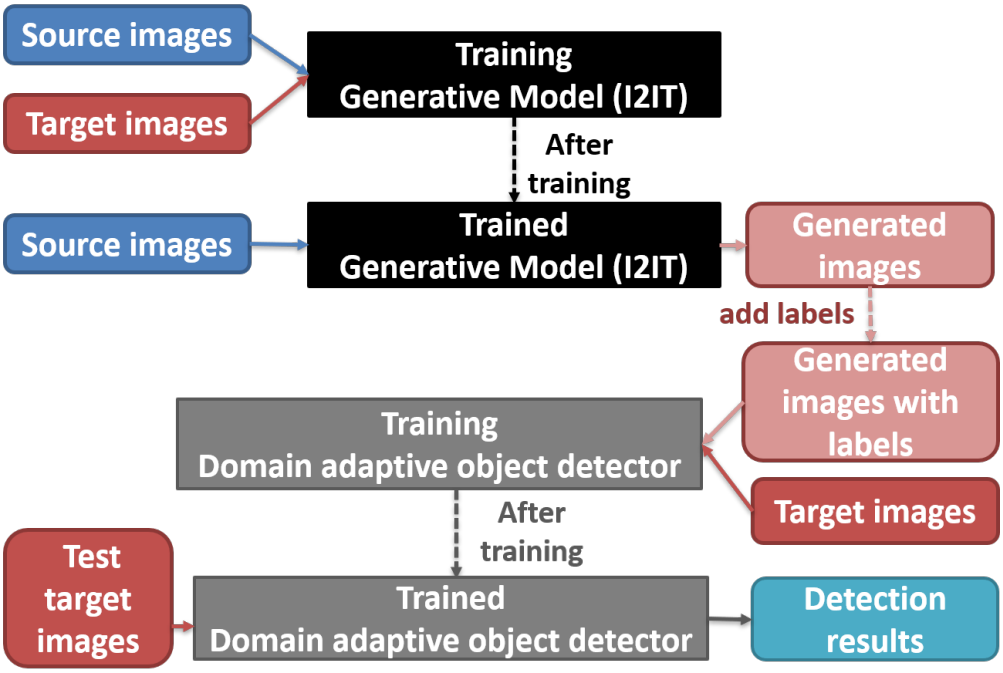


Figure 2.11: The proposed Generative Domain Adaptation (GDA) framework.

the domain shift between source and target domains, we propose the GDA framework that uses these generated visuals as source domain during training. In other words, the generative visuals in conjunction with unlabeled target data are used to train the domain adaptive object detector. This will help to improve the detection performance. For inference, and during testing, trained weights are loaded into the original object detector architecture (without the DAN network). Therefore, our proposed framework will not increase the underlying detector complexity during inference, which is an essential factor for many real-time applications such as autonomous driving. In fact, the proposed GDA framework is targeted to improve detection of objects (such as pedestrians and traffic signs) for autonomous vehicles driving under diverse weather conditions, and in particular rainy weather.

To evaluate the performance of the proposed GDA framework, we used the trained UNIT model in Section 2.3.1 to translate clear-weather annotated images into a corresponding set of rainy-

weather annotated images. Examples of the generated images are shown in Figure 2.8. It is important to note that these generated images have less domain shift with respect to real rainy images than the domain shift that they exhibit with respect to original images captured under clear weather. Afterward, we trained Domain adaptive Faster R-CNN using: (a) the generated images as representative of the source domain in conjunction with (b) unlabeled rainy images as the target domain.

2.4.2 Results and Discussion

We compare the GDA framework with other mitigation techniques in the previous sections including:

- Three state-of-the-art deraining algorithms [60, 61, 62].
- Training original Faster R-CNN object detection using generated images by the UNIT image-translation method instead of clear images.
- Domain adaptation without using generative model based training. In other words, we trained domain adaptive Faster R-CNN using the original annotated clear data as the source domain and the rainy data without annotation as the target domain.

The final results for these performance evaluation tests are summarized in Table 2.2.

The top row of the table includes the performance results for applying Faster R-CNN on clear weather data as a reference. As can be seen, even under clear weather, object detection performance is quite low for pedestrian, traffic light, and traffic sign. The second row shows the results of applying Faster R-CNN on the real rainy test data without any mitigation technique. It is clear that rainy conditions degrade the (already low) performance significantly. For example, AP for

pedestrian detection drops by 8. It is also clear that deraining algorithms fail miserably in these realistic scenarios. The table also shows that domain adaptation or image translation-based training provides some improvements. Meanwhile, significant improvements can be achieved under the GDA framework. When comparing the pedestrian detection performance of GDA with no-mitigation (second row), GDA improves the performance by around 50% toward the ”ideal” clear-weather performance. Hence, instead of a drop of 8 without any mitigation, under GDA, we have a drop of around four relative to clear conditions. For traffic sign detection, GDA performance is even more impressive. It reaches similar levels of performance as the clear-weather AP. For other classes (vehicle and traffic light), GDA achieves almost similar performance as other approaches. The overall mAP is also closing the gap toward the ideal clear-weather performance. Examples of visual detection results for the GDA improvements are shown in Figure 2.12 and Figure 2.13.

Figure 2.12: Visual detection results using: (Left) State-of-the-art (SOTA) Domain Adaptive Faster R-CNN [42] that failed in detecting pedestrians in these examples, and (Right) The proposed GDA framework that successfully detected the same pedestrians (red bounding boxes).

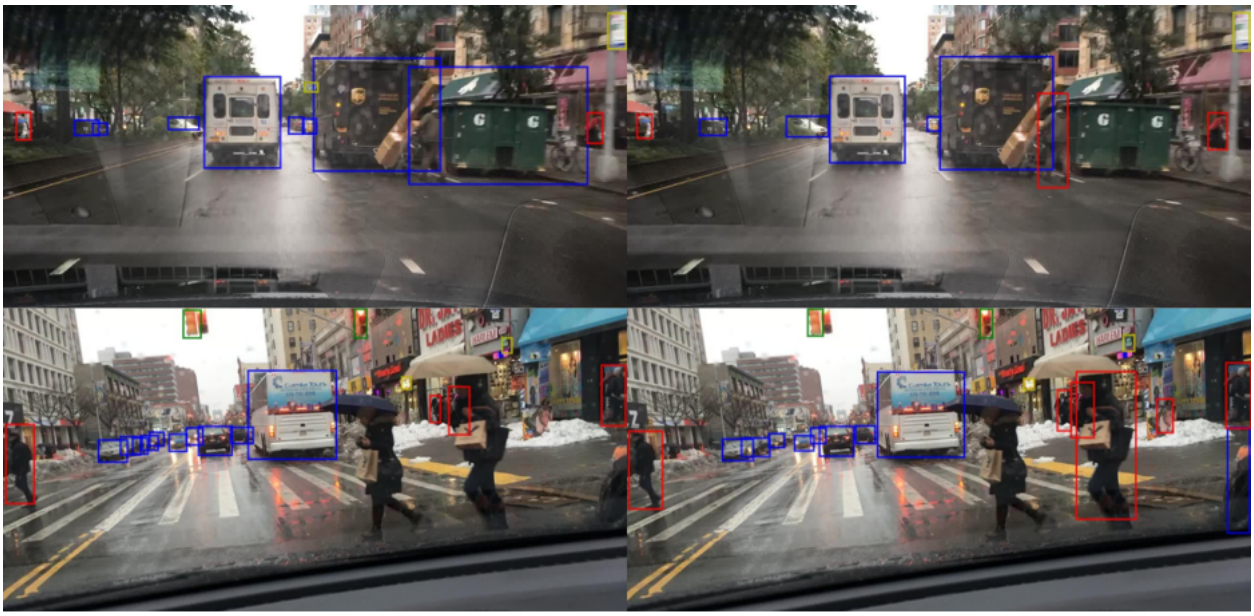


Figure 2.13: Visual detection results using Left: original Faster R-CNN object detector trained using generated images by image-to-image translation (I2IT) UNIT model [78], which failed in detecting some pedestrians, and Right: the proposed GDA framework, which successfully detected them (red bounding boxes).

Chapter 3

Multiscale Domain Adaptive YOLO for Cross-Domain Object Detection

Most previous approaches for domain adaptation object detection, used Faster R-CNN as the base detector. Despite its popularity, Faster R-CNN suffers from a long inference time to detect objects. As a result, it is arguably not the optimal choice for time-critical, real-time applications such as autonomous driving. On the other hand, one-stage object detectors, and in particular YOLO, can operate quite fast, even much faster than real-time, and this makes them invaluable for autonomous driving and similar time-critical applications. Furthermore, domain adaptation for the family of YOLO architectures has received virtually no attention. Besides the computational advantage of YOLO, the latest version, YOLOv4, has many salient improvements, and its object detection performance has improved rather significantly relative to prior YOLO architectures and more important in comparison to Faster R-CNN. All of these factors motivated our focus on the development of a new domain adaptation framework for YOLOv4.

In this chapter, we propose novel domain adaptation architectures for the YOLOv4 object detector. In particular, we introduce four new *MultiScale Domain Adaptive YOLO* (MS-DAYOLO) architectures that promote multiscale domain adaptation for the feature extraction stage and progressive channel reduction strategies for one or more domain classifiers. The proposed MS-DAYOLO framework achieves significant improvements over YOLOv4 as shown in the examples

of Figure 3.1 (c). To the best of our knowledge, this is the first proposed work that improves the performance of YOLO for cross-domain object detection [93].

3.1 Proposed MultiScale Domain Adaptive YOLO

YOLOv4 [13] has been released recently as the latest version of the family of the YOLO object detectors. Relative to its predecessor, YOLOv4 has incorporated many new revisions and novel techniques to improve the overall detection accuracy. YOLOv4 has three main parts: backbone, neck, and head as shown in Figure 3.2. The backbone is responsible for extracting multiple layers of features at different scales. The neck collects these features from three different scales of the backbone using upsampling layers and feed them to the head. Finally, the head predicts bounding boxes surrounding objects as well as class probabilities associated with each bounding box.

The backbone (*i.e.* feature extractor) represents a major module of the YOLOv4 architecture, and we believe that it makes a significant impact on the overall performance of the detector. In addition to many convolutional layers, it has 23 residual blocks [79], and five downsampling layers to extract critical layers of features that are used by the subsequent detection stages. Here, we concentrate on the features (F1, F2, and F3 in Figures 3.2) because they are fed to the next stage (neck module). In particular, our goal is to apply domain adaptation to these three features to make them robust against domain shifts, and hence, have them converge toward domain invariance during domain adaptation based training. Equally important, these three stages of features have different dimensions due to the successive downsampling layers that progressively reduce the width and height of features by half while doubling the number of channels. If d is the width of the feature at the first scale (F1), then the dimensions of the three stages of features are: F1: $d \times d \times 256$, F2: $\frac{d}{2} \times \frac{d}{2} \times 512$, and F3: $\frac{d}{4} \times \frac{d}{4} \times 1024$.

(a) YOLOv4 (b) YOLOv4 (c) Our MS-DAYOLO

Figure 3.1: Visual detection examples using the original YOLOv4 method on: (a) clear images and (b) foggy images. (c) Our proposed MS-DAYOLO applied onto foggy images. The images are from the Cityscapes [70] and Foggy Cityscapes [94] datasets.

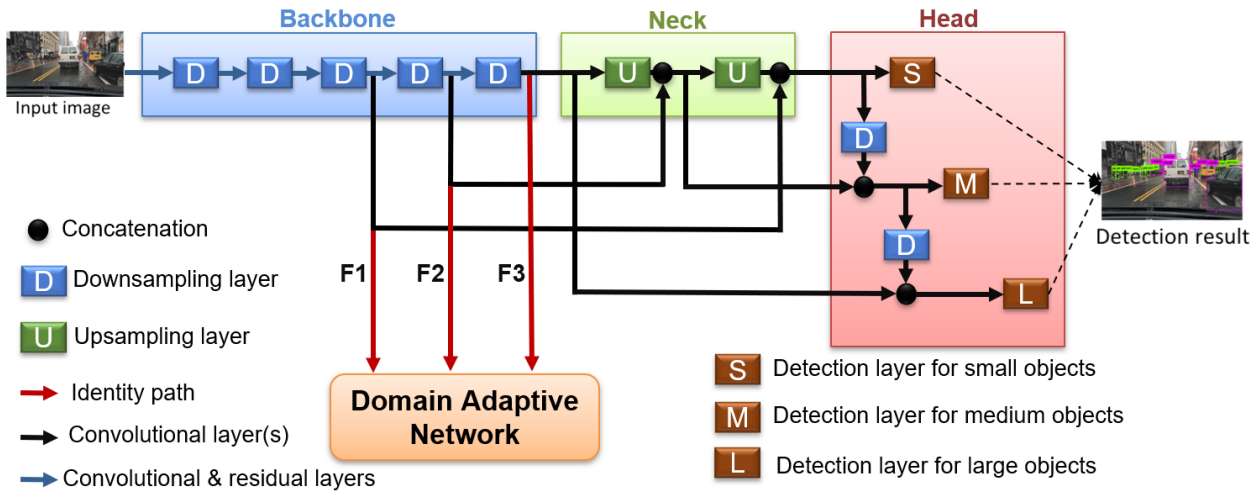


Figure 3.2: Architecture of YOLOv4 with domain adaptation network (DAN) to develop domain adaptive YOLO. The details architectures of DAN are shown in Figure 3.3.

3.1.1 Domain Adaptive Network for YOLO

The proposed Domain Adaptive Network (DAN) is attached to the YOLOv4 object detector only during training in order to learn domain invariant features. Indeed, YOLOv4 and DAN are trained in an end-to-end fashion. For inference, and during testing, domain-adaptive trained weights are used in the original YOLOv4 architecture (without the DAN network). Therefore, our proposed framework will not increase the underlying detector complexity during inference, which is an essential factor for many real-time applications such as autonomous driving.

DAN uses the three distinct scale features of the backbone that are fed to the neck as inputs. It has several convolutional layers to predict the domain class (either source or target). Then, domain classification loss ($\mathcal{L}_{dc}$) is computed via binary cross entropy as follows:

$$\mathcal{L}_{dc} = -\frac{1}{N}\sum_{i,x,y}[t_i \ln p_i^{(x,y)} + (1-t_i)\ln(1-p_i^{(x,y)})] \tag{3.1}$$

Here, t_i is the ground truth domain label for the i-th training image, with $t_i = 1$ for source

domain and $t_i = 0$ for target domain. $p_i^{(x,y)}$ is predicted domain class probabilities for i-th training image at location (x, y) of the feature map. N represents the total number of images in a batch multiplied by the total number of elements in the feature map.

DAN is optimized to differentiate between the source and target domains by minimizing this loss. On the other hand, the backbone is optimized to maximize the loss to learn domain invariant features. Thus, features of the backbone should be indistinguishable for the two domains. Consequently, this should improve the performance of object detection for the target domain. To solve the joint minimization and maximization problem, we employ the adversarial learning strategy [41]. In particular, we achieve this contradictory objective by using a Gradient Reversal Layer (GRL) [34, 31] between the backbone and the DAN network.

To compute the detection loss ($\mathcal{L}_{det}$) [13], only source images are used because they are annotated with ground-truth objects. Consequently, all three parts of YOLOv4 (*i.e.* backbone, neck and head) are optimized via minimizing $\mathcal{L}_{det}$. On the other hand, both source labeled images and target unlabeled images are used to compute the domain classification loss ($\mathcal{L}_{dc}$) which is used to optimize DAN via minimizing it, and the backbone via maximizing it. As a result, both $\mathcal{L}_{det}$ and $\mathcal{L}_{dc}$ are used to optimize the backbone. In other words, the backbone is optimized by minimizing the following total loss:

$$\mathcal{L}_t = \mathcal{L}_{det} + \lambda \mathcal{L}_{dc} \tag{3.2}$$

where λ is a negative scalar of GRL that balances a trade-off between the detection loss and domain classification loss. In fact, λ controls the impact of DAN on the backbone.

3.1.2 DAN Architectures

We developed various architectures for the Domain Adaptive Network (DAN) as shown in Figure 3.3 to explore and gain insight into the impact of different components on achieving improved performance for the target domain. Under all of our architectures, we employ a multiscale strategy that connects the three features F1, F2, and F3 of the backbone to the DAN through three corresponding GRLs. Other than this common multiscale strategy, the proposed DAN architectures differ from each other as explained below.

a- Multiscale Baseline : Instead of applying domain adaptation for only the final scale of the feature extractor as done in the Domain Adaptive Faster R-CNN architecture [42], we develop domain adaptation for three scales separately. In other words, applying domain adaptation only to the final scale (F3) does not make a significant impact on the previous scales (F1 and F2). As a result, we apply domain adaptation to all scales as shown in Figure 3.3 (a). For each scale, there are two convolutional layers after GRL, the first one reduces the feature channels by half, and the second one predicts the domain class probabilities. Finally, a domain classifier layer is used to compute the domain classification loss.

b- Progressive Feature Reduction (PFR): As shown in Figure 3.3(a), the baseline architecture reduces the feature vector size resulting from the YOLOv4 backbone into a single-value feature (scalar) rather abruptly through two stages of neural networks. This simple two-stage DAN aims at generating a single feature value that serves as an input into the domain classifier. The fact that the domain classifier requires a single feature value is inherent in the binary nature of the classifier that simply needs to classify the image data into either source domain or target domain. Meanwhile, and due to the adversarial strategy used here, the above Baseline domain adaptation network is competing with the significantly more complex network of the backbone as shown in Figure 3.2.

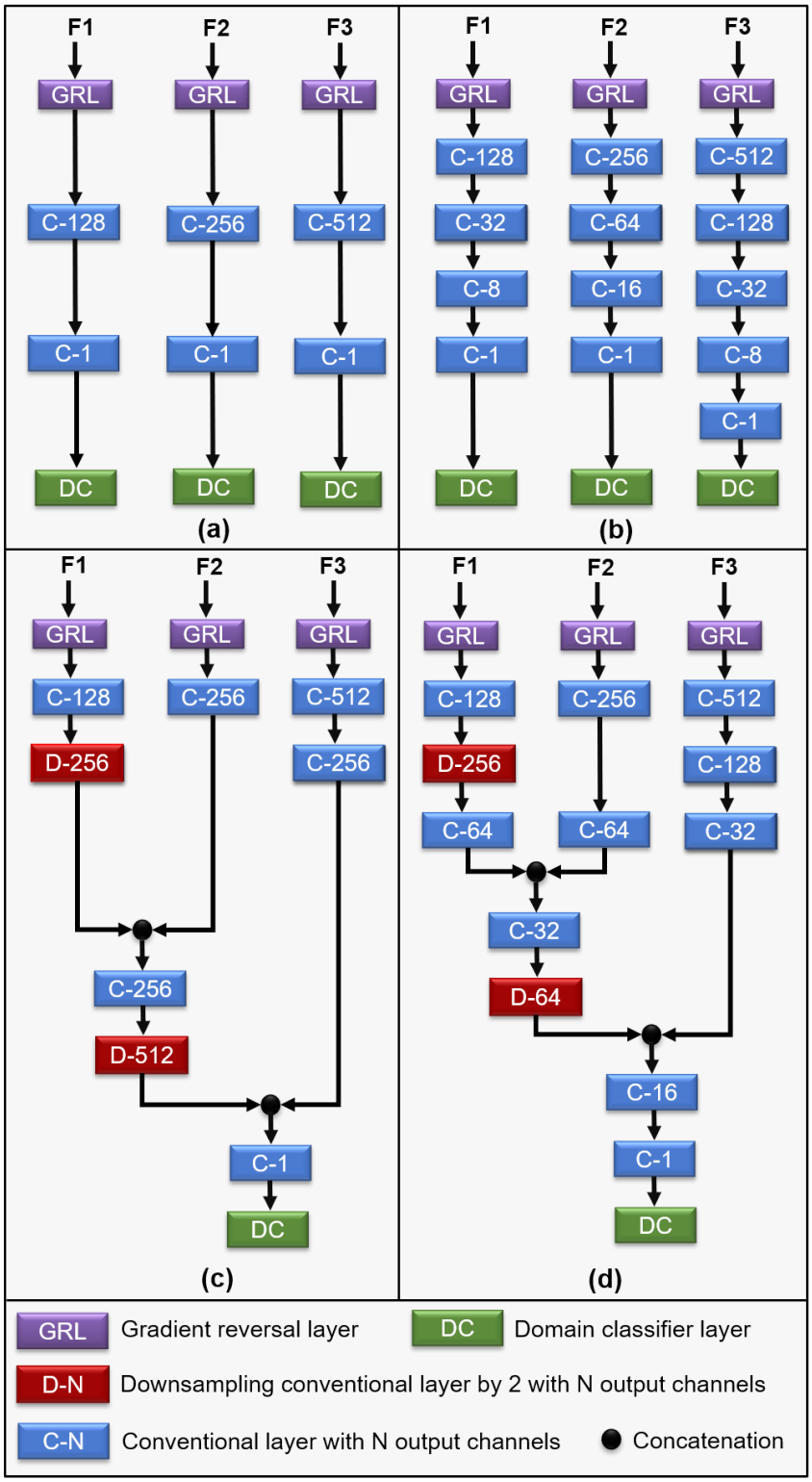


Figure 3.3: Proposed architectures for the Domain Adaptive Network (DAN): (a) Baseline, (b) Progressive Feature Reduction (PFR), (c) Unified Domain Classifier (UDC), (d) Integrated. F1, F2, and F3 are features of the backbone network of YOLO that are fed to the neck.

We observed that this mismatch between the simplistic baseline DAN architecture and the complex backbone network could compromise the domain adaptation performance. Thus, the DAN network may not be sufficiently powerful to distinguish between the source and target domains since the complex backbone network can easily confuse (and trick) the DAN network. To mitigate this mismatch, we increase the number of convolutional layers for each scale by progressively reducing the feature channels as shown in Figure 3.3(b). This progressive reduction of feature channels helps the DAN network to compete more efficiently against the more complex backbone. As a result, the extracted features by the backbone network will be more domain invariant.

Therefore, while the baseline architecture reduces the number of the feature channels using two stages of neural networks, our proposed progressive-feature-reduction employs four or five stages depending on the original feature size. In particular, for feature vectors F1 and F2, we employ four stages of neural networks that progressively reduce the feature vector size from 128 and 256, respectively, toward a single-feature scalar value, which is all that is required as an input for the binary domain classifier. For feature vector F3, we employ a five-stage neural network DAN to progressively reduce the backbone feature vector toward the scalar feature value. It is important to highlight the following regarding the proposed progressive-feature-reduction architecture. It is possible to employ a larger number of stages of progressive reduction than the number of stages we employed in our architecture shown in Figure 3.3(b). However, based on our experience, increasing the number of stages beyond four or five stages does not necessarily improve the overall performance.

c- Unified Domain Classifier (UDC): Under the multiscale baseline and Progressive Feature Reduction architectures, each scale has its own distinct domain classifier. This multi-classifier strategy may lead to inconsistency among scales. For example, a domain classifier at one scale may

classify an image patch as a source data, while a domain classifier at another scale may classify the same image patch as originating from the target domain. (Examples of this inconsistency are shown later in the Experiments section.) To address this potential inconsistency, we propose to use a single (unified) domain classifier that combines the feature vectors from all scales as shown in Figure 3.3(c). It is important to highlight the following about the proposed Unified Domain Classifier (UDC) domain adaptive network:

1. We use downsampling convolutional layers to match the size of features at different scales. For example, in order to combine the feature vectors resulting from the F1 and F2 scales of the backbone, we add a downsampling stage to the F1 scale and concatenate the resulting vector with the feature vector from F2. This strategy maintains the multiscale attribute of our domain adaptive network while targeting a unified domain classifier architecture.

2. Furthermore, we concatenate features at different scales in a way to make each scale contributes equally in terms of the number of feature channels. In other words, each feature scale equally contributes to the prediction of the domain class probabilities.

d- Integrated: It is important to note that the above two improvements, progressive feature reduction (PFR) and unified domain classifier (UDC), have been applied directly and separately to the multiscale baseline architecture. Consequently, to gain the benefits of both the progressive feature reduction and unified domain classifier strategies, we integrate them in one network as shown in Figure 3.3(d). In principle, we have developed the network by complementing the unified-classifier architecture (Figure 3.3(c)) with additional stages of convolutional layers to achieve a more progressive reduction in feature channel sizes. This is evident by comparing the two architectures shown in Figures 3.3(c) and 3.3(d).

3.2 Experiments

In this section, we evaluate our proposed domain adaptive YOLO framework and the proposed MS-DAYOLO architectures. We modified the official source code of YOLOv4 that is based on the darknet platform[1], and developed a new code to implement our proposed methods[2].

3.2.1 Setup

For training, we used the default settings and hyper-parameters that were used in the original YOLOv4[13]. The network is initialized using the pre-trained weights file. The training data includes two sets: source data that has images and their annotations (bounding boxes and object classes), and target data without annotation. Each batch has 64 images, 32 from the source domain and 32 from the target domain. Based on prior works [42, 43, 46] and our experience based on trial and error technique, we set $\lambda = 0.1$ for all experiments.

For evaluation, we report Average Precision (AP) for each class as well as mean average precision (mAP) with a threshold of 0.5 [77] using testing data that has labeled images of the target domain. We have followed other prior domain adaptive object-detection works that use the same threshold value of 0.5. We compare our proposed method with the original YOLOv4 and other state-of-the-art domain adaptation approaches that are based on Faster R-CNN object detector [6], all applied to the same target domain validation set.

[1] https://github.com/AlexeyAB/darknet
[2] https://github.com/Mazin-Hnewa/MS-DAYOLO

3.2.2 Results

3.2.2.1 Adverse Weather Adaptation

Domain shift due to changes in weather conditions is one of the most prominent reasons for the discrepancy between the source and target domains. Reliable object detection systems in different weather conditions are essential for many critical applications such as autonomous driving. As a result, we focus on presenting the evaluation results of our proposed MS-DAYOLO framework by studying domain shifts under changing weather conditions for autonomous driving. To achieve this, we use three different driving datasets: Cityscapes [70], Foggy Cityscapes [94], Waymo [95].

Clear $\rightarrow$ Foggy: We discuss the ability of our proposed method to adapt from clear to foggy weather using driving datasets: Cityscapes [70] and Foggy Cityscapes [94] as has been done by many recent works in this area [42, 47, 44, 46, 96, 97, 43, 45, 98]. The Cityscapes training set has 2975 labeled images that are used as source domain. Similarly, the Foggy Cityscapes training set also has 2975 images, but without annotations, and is used as the target domain. Original YOLOv4 is trained using the source domain data only. In contrast, MS-DAYOLO is trained using both source and target domain data. The Foggy Cityscapes validation set has 500 labeled images that are used for testing and evaluation. Because the Foggy Cityscapes training set is annotated, we are able to train the original YOLOv4 with this set to show the ideal performance (oracle).

Table 3.1 summarizes the performance results. Based on these results, all of the architectures of our proposed framework outperform the original YOLOv4 approach by a significant margin. Moreover, the proposed integrated architecture achieves the best overall performance in terms of mAP. Although the GPA method has slightly better results than the proposed integrated architecture in terms of AP for some classes, MS-DAYOLO achieves the best overall performance in term of mAP, and it is significantly faster than GPA in inference time. It is worth noting that the proposed

Table 3.1: Quantitative results of domain adaptation for the clear → foggy experiment of the Cityscapes dataset. The MS-DAYOLO uses YOLOv4 object detector [13], while the other methods use Faster R-CNN object detector [6]. *The results are reported from [97] and [99]. The inference time is measured in Frame Per Second (FPS) using NVIDIA GeForceGTX 1080 Ti GPU.

Method	Backbone	Person	Rider	Car	Truck	Bus	Train	Mcycle	Bicycle	mAP	FPS
DAF[42]		25.0	31.0	40.5	22.1	35.3	20.2	20.0	27.1	27.6	
MAF[46]		28.2	39.5	43.9	23.8	39.9	33.3	29.2	33.9	34.0	
iFAN[48]	VGG16	32.6	40.0	48.5	27.9	45.5	31.7	22.8	33.0	35.3	6.2
CT[49]		32.7	44.4	50.1	21.7	45.6	25.4	30.1	36.8	35.9	
PDA[50]		36.0	45.5	54.4	24.3	44.1	25.8	29.1	35.9	36.9	
DAF[42]*		29.2	40.4	43.4	19.7	38.3	28.5	23.7	32.7	32.0	
MTOR[97]	ResNet50	30.6	41.4	44.0	21.9	38.6	40.6	28.3	35.6	35.1	3.7
GPA[99]		32.9	**46.7**	54.1	24.7	45.7	41.1	**32.4**	**38.7**	39.5	
YOLOv4		31.6	38.3	46.9	23.9	39.9	20.1	16.8	30.3	31.0	
	Baseline	38.6	45.5	55.9	22.8	45.6	32.5	28.8	36.5	38.3	
MS-	PFR	38.5	46.5	56.5	27.6	48.7	38.5	26.4	38.4	40.1	
DAYOLO	UC	39.3	45.0	**57.0**	**29.9**	48.0	36.6	30.2	36.4	40.3	**48.2**
	Integrated	**39.6**	46.5	56.5	28.9	**51.0**	**45.9**	27.5	36.0	**41.5**	
Oracle		42.4	49.5	63.6	37.6	59.8	47.1	31.1	39.9	46.3	

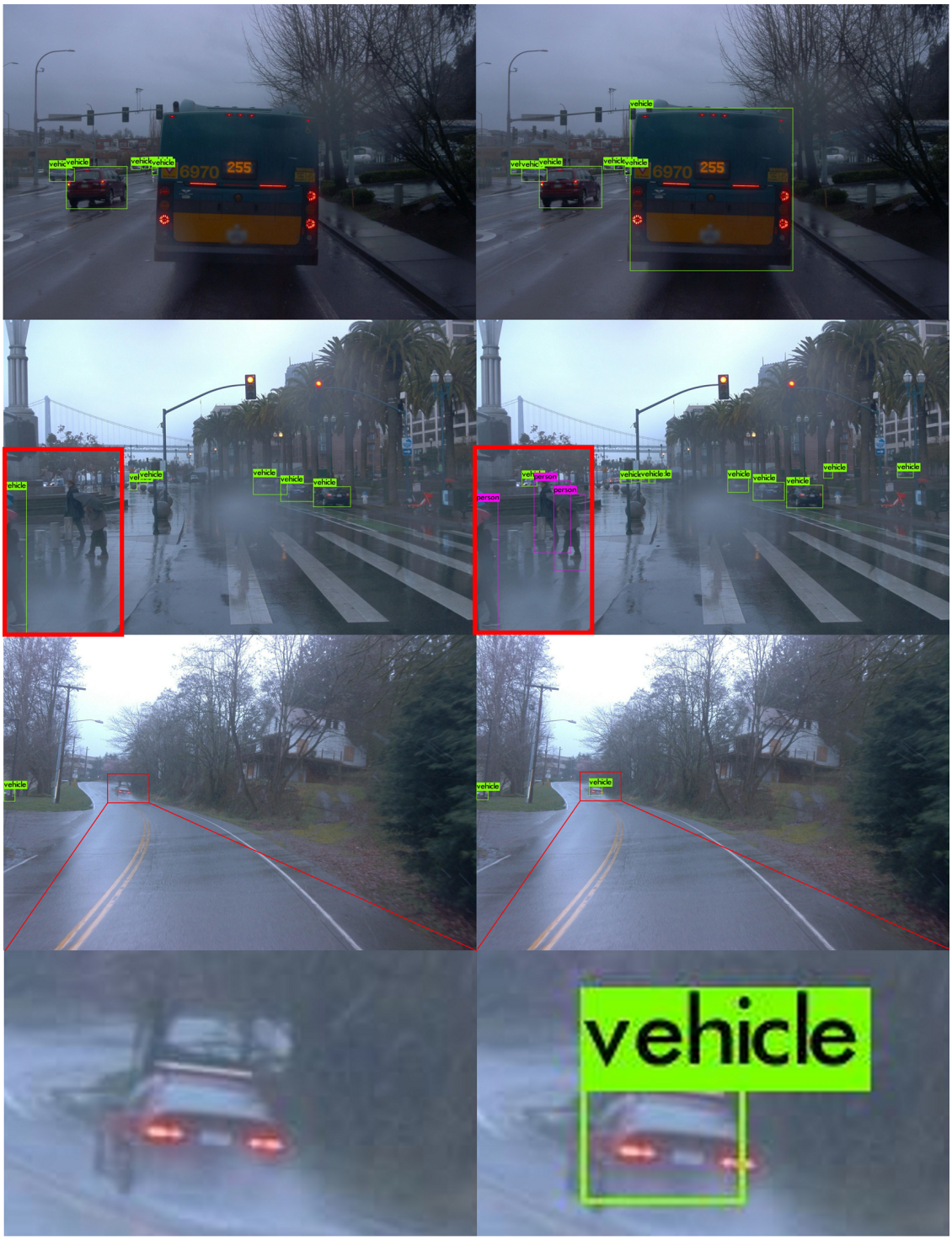
integrated architecture achieves significant improvements relative to the original YOLOv4, and it almost reaches the performance of the ideal (oracle) scenario, especially for some object classes in terms of average precision. Figure 3.1 shows examples of detection results of the proposed method as compared to the original YOLOv4.

Sunny → Rainy: we present results for applying YOLOv4 and our MS-DAYOLO framework on the Waymo dataset [95], which includes two sets of visual data that are designated as ”sunny” and ”rainy”. We extracted 14319 ”sunny weather” labeled images for the source data, and 13004 ”rainy weather” unlabeled images to represent the target data. As before, the original YOLOv4 is trained using only source data (*i.e.* labeled sunny images). Meanwhile, our proposed MS-DAYOLO is trained using both source and target data (*i.e.* labeled sunny images and unlabeled rainy images). In addition, we extracted 1676 labeled images from the rainy-weather data for testing and evaluation. It is important to note the following key observations regarding the Waymo

Figure 3.4: Examples of training images of Waymo dataset [95]. Images in the top row are tagged as being captured in sunny weather, while images in the bottom row are tagged as being captured in rainy weather. It is obvious that the domain shift between sunny and rainy images is not very significant.

datasets: (a) The designations ”sunny” and ”rainy” images have been determined by the providers of the Waymo dataset. (b) From our extensive experience in working with this data, the distinction between sunny and rainy image samples is quite subjective, and in many cases, one can argue that a ”rainy” image sample should be labeled as ”sunny” or vice versa. Consequently, the domain shift between the two domains, which are designated as ”sunny” and ”rainy”, is not very significant as shown in the examples of Figure 3.4. This is crucial since training the original YOLOv4 using the Waymo ”sunny” dataset effectively covers a large number of ”rainy” testing samples that fall within the source ”sunny” domain. Nevertheless, we opted to follow the dataset designations with the aim of evaluating any potential improvements that the proposed MS-DAYOLO framework may provide.

The results are summarized in Table 3.2. It is clear that the MS-DAYOLO framework still provided good improvements despite the fact that, in this case, the two domains, sunny and rainy,



(a) YOLOv4 (b) Our MS-DAYOLO

Figure 3.5: Visual detection examples of the sunny → rainy experiment using (a) the original YOLOv4, and (b) Our proposed Integrated MS-DAYOLO applied onto labeled rainy images extracted from the Waymo datasete [95].

Table 3.2: Quantitative results of domain adaptation for the sunny → rainy experiment of the Waymo dataset.

Method		Person	Vehicle	mAP
YOLOv4		38.56	55.41	46.99
MS-DAYOLO	Baseline	39.97	55.37	47.67
	PFR	39.75	56.34	48.05
	UC	39.42	56.66	48.04
	Integrated	**40.03**	**56.97**	**48.50**

have a significant overlap. This could explain why the improvements are not as salient as the improvements achieved when applying MS-DAYOLO on the Cityscapes data, which consisted of two clearly distinct domains as shown in the examples of Figure 3.1. Moreover, and similar to the clear → foggy experiment, we observe that the proposed Progressive Feature Reduction (PFR), Unified Domain Classifier (UDC), and Integrated architectures improve the detection performance relative to the baseline architectures when applied to the Waymo dataset. For this experiment, we do not report the performance of other domain adaptive object detection methods that are based on Faster R-CNN because none of these methods reported or used the Waymo dataset for the sunny → rainy domain-shift scenario. Figure 3.5 shows examples of detection results of the proposed Integrated MS-DAYOLO framework as compared to the original YOLOv4. In addition, Figure 3.6 shows examples of detection results that the integrated architecture succeeds in detecting objects while the baseline architecture fails to detect the same objects. Furthermore, Figure 3.7 shows examples where the baseline architecture suffers from false positive cases, while the integrated one eliminates these false positives, which contribute to its improved performance.

3.2.2.2 Cross Camera Adaptation

Domain shift can occur between different real visual datasets captured by different driving vehicles equipped with different cameras even if these visuals are taken under similar weather conditions.

(a) Baseline MS-DAYOLO (b) Integrated MS-DAYOLO

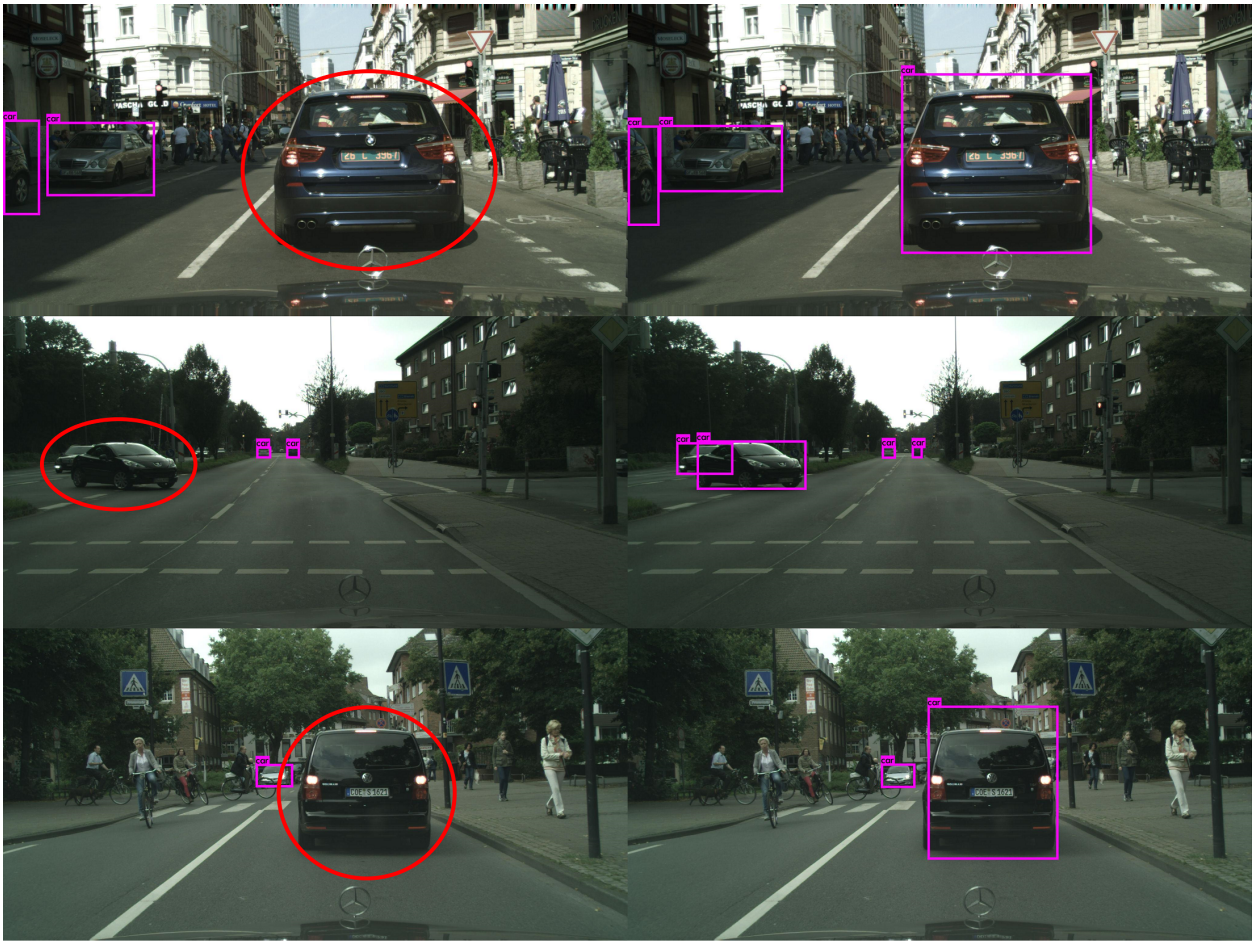
Figure 3.6: Visual detection examples of the sunny → rainy experiment using (a) the baseline architecture, and (b) the integrated architecture of MS-DAYOLO applied onto the rainy images extracted from the Waymo datasete [95]. The baseline MS-DAYOLO fails to detect people that cross the street in the top two images, and cars in the bottom two images, while the integrated MS-DAYOLO successfully detects them.

(a) Baseline MS-DAYOLO (b) Integrated MS-DAYOLO

Figure 3.7: Visual detection examples of the sunny → rainy experiment using (a) the baseline architecture, and (b) the integrated architecture of MS-DAYOLO applied onto the rainy images extracted from the Waymo datasete [95]. In these examples, the baseline MS-DAYOLO suffers from false positive problem, while the integrated MS-DAYOLO eliminates these false positives. **FP**: means false positive.

Such domain shift is usually driven by different camera setups leading to a shift in image quality and resolution. Moreover, such datasets are usually captured in various locations, which have different views and driving environments. All these factors lead to domain disparity between datasets. Under this experiment, we evaluate the performance of our MS-DAYOLO framework for domain adaptation between two real driving datasets: KITTI [69] and Cityscapes as has been done by many recent works in this area [42, 46, 96, 43, 50]. In particular, the KITTI training set which has 6000 labeled images, is utilized as source data. While the Cityscapes training set which has 2975 images, but without labels, is utilized as target data. The Cityscapes validation set which has 500 labeled images, is used for testing and evaluation.

Table 3.3 presents the performance results based on the car AP as has been reported by prior works [96, 42, 43, 46, 50] because it is the only common object class between the two datasets. A clear performance improvement is achieved by our method over the original YOLOv4. We also observe that the proposed Progressive Feature Reduction (PFR), Unified Domain Classifier (UDC), and Integrated architectures improve the detection performance relative to the baseline architecture. Although the GPA method outperforms Integrated MS-DAYOLO by a small margin (0.3%), our MS-DAYOLO runs in real-time, and it is significantly faster than GPA in terms of frames per second (FPS), which is essential for time-critical applications. Figure 3.8 shows visual examples for qualitative comparison of our method with the original YOLOv4. It is obvious from these examples that our approach successfully detects the vehicles in the scenes while the original YOLOv4 fails to detect the same vehicles.



(a) YOLOv4 (b) Our MS-DAYOLO

Figure 3.8: Visual detection examples of the KITTI → Cityscapes experiment for the car class using (a) the original YOLOv4, and (b) Our proposed integrated MS-DAYOLO applied onto the Cityscapes validation set. These examples show that the integrated MS-DAYOLO successfully detects the vehicles in the scenes while the original YOLOv4 fails to detect the same vehicles.

Table 3.3: Quantitative results of cross camera adaptation from KITTI to Cityscapes based on AP of the car class which is shared between the two datasets. The MS-DAYOLO uses YOLOv4 object detector [13], while the other methods use Faster R-CNN object detector [6]. *The results are reported from [99]. The inference time is measured in Frame Per Second (FPS) using NVIDIA GeForceGTX 1080 Ti GPU.

Method	Backbone	Car AP	FPS
DAF [42]	VGG16	38.5	6.2
MAF [46]		41.0	
CT [49]		43.6	
PDA [50]		43.9	
DAF [42]*	ResNet50	41.8	3.6
GPA [99]		**47.9**	
YOLOv4		44.5	**48.2**
MS-DAYOLO	Baseline	45.5	
	PFR	46.8	
	UC	47.3	
	Integrated	47.6	

3.2.3 Ablation Study

To show the importance of applying domain adaptation to three distinct scales of the backbone network, we conducted an ablation study for the clear $\rightarrow$ foggy experiment. First, we applied domain adaptation, separately, to each of the three scales of features that are fed into the neck of the YOLOv4 architecture. Also, we applied domain adaptation to different combinations of two scales at a time. Finally, we compared the results with the performance of applying these combinations of the study with the performance of applying our baseline MS-DAYOLO to all three scales as explained in section 3.1.2. Another important aspect of this ablation study is that we wanted to consider objects that have statistically significant numbers of sample data. In that context, because the number of ground-truth objects for some classes (truck, bus, and train) is small (*i.e.* less than 500 in the training set, and 100 in the testing set), the performance measure will be inaccurate for these classes. As a result, we exclude them in this ablation study and compute mAP based on the remaining classes.

Table 3.4: Ablation Study, ✓means that domain adaptation is applied to the feature scale(s) using our baseline MS-DAYOLO for the clear → foggy experiment.

F1	F2	F3	Person	Rider	Car	Mcycle	Bicycle	mAP
			31.57	38.27	46.93	16.75	30.32	32.77
		✓	36.84	42.84	53.69	24.77	32.35	38.09
	✓		37.08	41.49	54.49	26.22	32.43	38.34
✓			36.28	44.22	53.10	25.81	35.87	39.06
	✓	✓	36.62	42.68	55.70	26.09	33.52	38.92
✓	✓		37.50	42.48	54.53	27.84	34.75	39.43
✓		✓	36.41	**46.06**	52.19	22.48	34.99	38.43
✓	✓	✓	**38.62**	45.52	**55.85**	**28.82**	**36.46**	**41.05**

Table 3.4 summarizes results of the ablation study. It is clear that based on these results, we can conclude that applying domain adaptation to all three feature scales improves the detection performance on the target domain, and achieves the best result.

3.2.4 Analysis

In order to show the benefit of using a unified domain classifier instead of three different domain classifiers, we recorded the domain classifier loss of Equation 3.1 over training iterations for the KITTI → Cityscapes experiment. Figure 3.9 shows the losses of the three domain classifiers, corresponding to features F1, F2, and F3 of the baseline architecture over the first 2500 iterations of training. We can see that the losses are dissimilar after 1K iterations. This implies inconsistency among the classifiers' performance, which leads to a drop in performance. This motivated our objective in employing a unified domain classifier for all three scales. In turn, this led to the UC architecture for improving the detection performance when applied to the target domain data as shown in Tables 3.1, 3.2 and 3.3.

Moreover, to study the relationship between the domain classification loss of DAN and the detection performance, we conducted a time analysis during training. We plot in Figure 3.10 the

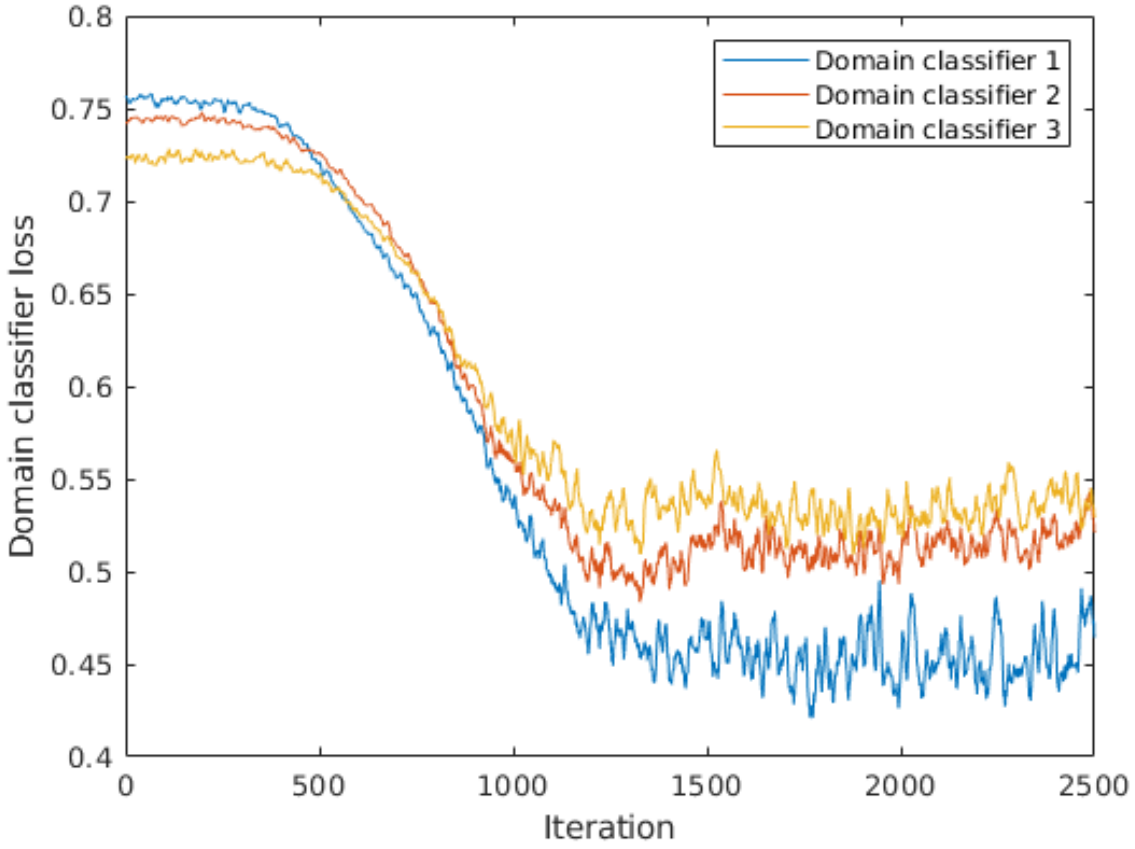


Figure 3.9: Losses of three domain classifiers of the baseline architecture over the first 2500 iterations of training for the KITTI → Cityscapes experiment.

domain classifier (DC) loss of the integrated architecture, and the detection performance in term of mAP for the KITTI $\rightarrow$ Cityscapes experiment. We normalize mAP by 100 to plot it at the same scale with the DC loss. At the beginning of the training, we found the DC loss starts at its highest values that are around 0.745. Then, and as training progresses, the DAN is optimized by minimizing the loss while the YOLO backbone is optimized by maximizing the loss. In other words, the DAN and the YOLO backbone compete against each other. From the figure, we observe that the detection performance continues to improve until the loss approximately reaches around 0.6. After that, the performance almost remains the same because the impact of the DAN on the backbone will not be significant as the DC loss becomes small.

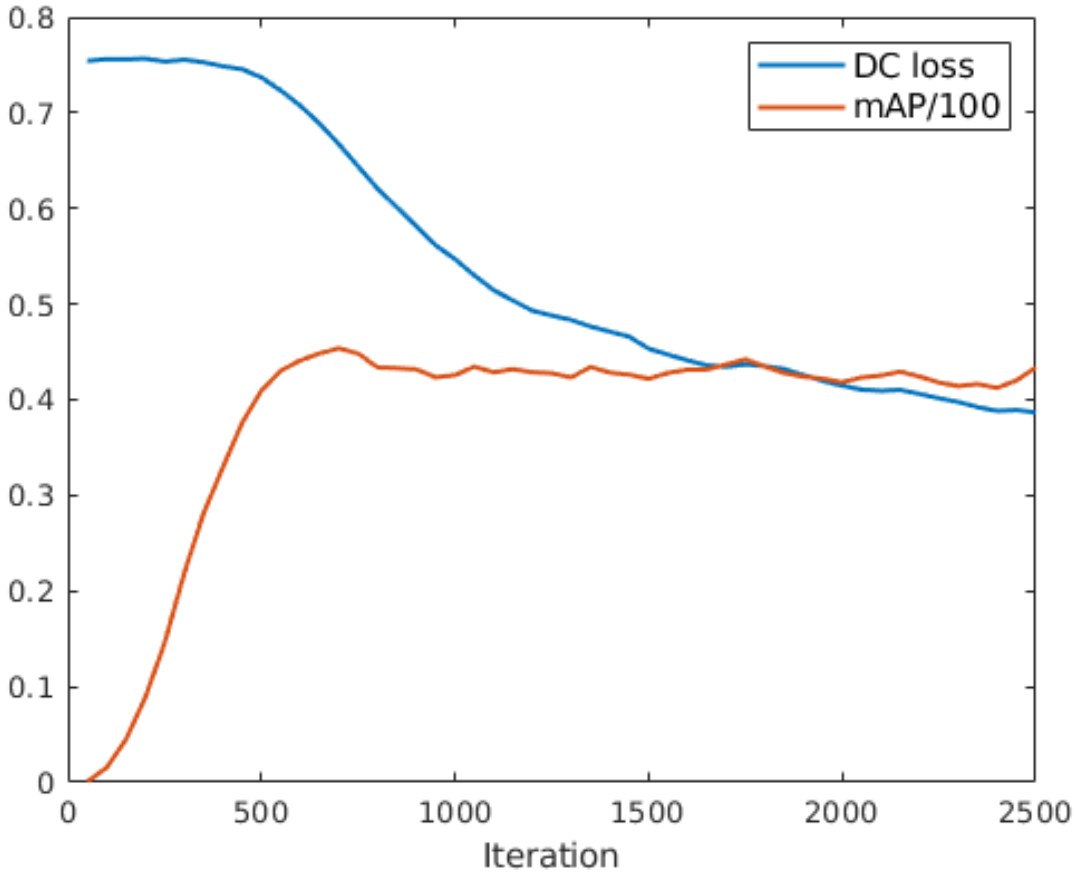


Figure 3.10: Domain classifier (DC) loss of the integrated architecture, and the detection performance in term of normalized mAP over the first 2500 iterations of training for the KITTI $\rightarrow$ Cityscapes experiment.

Chapter 4

Cross Modality Knowledge Distillation for Robust Pedestrian Detection

Detecting pedestrians in low light and adverse weather conditions would be challenging if only RGB imaging is used as input. This is because the quality of images captured under poor lighting and/or adverse weather conditions can degrade rather significantly. This is due to the fact that low lighting can adversely impact the dynamic range of RGB images while decreasing signal-to-noise ratio (SNR) levels, thereby affecting the performance of deep learning and related computer vision algorithms. Consequently, the performance of object detection methods can significantly drop. One solution is incorporating other sensing modalities such as thermal and gated imaging [100, 27, 22]. However, such sensor modalities are expensive, and incorporating them into ADAS and autonomous vehicle platforms can significantly increase production cost as well. Moreover, additional sensors add complexity to the design and manufacturing processes. For instance, extra laser illuminators need to be installed on the car for gated imaging [101]. In addition, using several modalities increases the inference time of a detection model due to sensing time and the computation overhead of a fusion technique for combining different modalities. Naturally, increasing inference time is undesirable for many real-time applications such as ADAS and autonomous driving.

Knowledge Distillation (KD) is one of the most effective techniques to transfer knowledge

between different models. It was initially proposed for model compression to transfer the information for a large complex teacher model to a smaller and simpler student model without a substantial drop in accuracy [102]. Typically, KD methods rely on a teacher model that supervises a student model during training to improve the performance of the student model [103, 104, 105]. KD methods were originally proposed for classification tasks [104, 106, 107, 108, 109]. Subsequently, KD has been adapted to object detection, which is inherently more challenging than classification [51, 52, 53, 54, 55, 56]. These previous KD approaches for object detection transfer the knowledge from a large model to a smaller one (*i.e.* model compression). However, we propose that KD can also be used in an alternate way where we can transfer multi-modal information. Specifically, we refer to the case where a single modality student model learns from a multi-modal teacher model.

In this chapter, we propose a novel framework that is based on Cross Modality Knowledge Distillation (CMKD) to improve the performance of RGB-based pedestrian detection in low light and adverse weather conditions. We achieve this by transferring the knowledge of a teacher detector that is trained using both RGB and gated images to a student detector, which is trained using RGB images only as shown in Figure 4.1. The proposed CMKD framework makes the student model generate features that are similar to the features of the teacher model. To accomplish this, we develop two methods within the proposed CMKD framework. The first one is based on using a KD loss, while the second one incorporates adversarial training with knowledge distillation. Based on experimental results, we show that both of our proposed CMKD methods significantly improve the detection performance relative to a baseline RGB detector, and they reduce the performance gap between teacher and baseline models by a considerable margin. To the best of our knowledge, this is the first proposed work that uses CMKD to improve the performance of pedestrian detection in low light and adverse weather conditions.

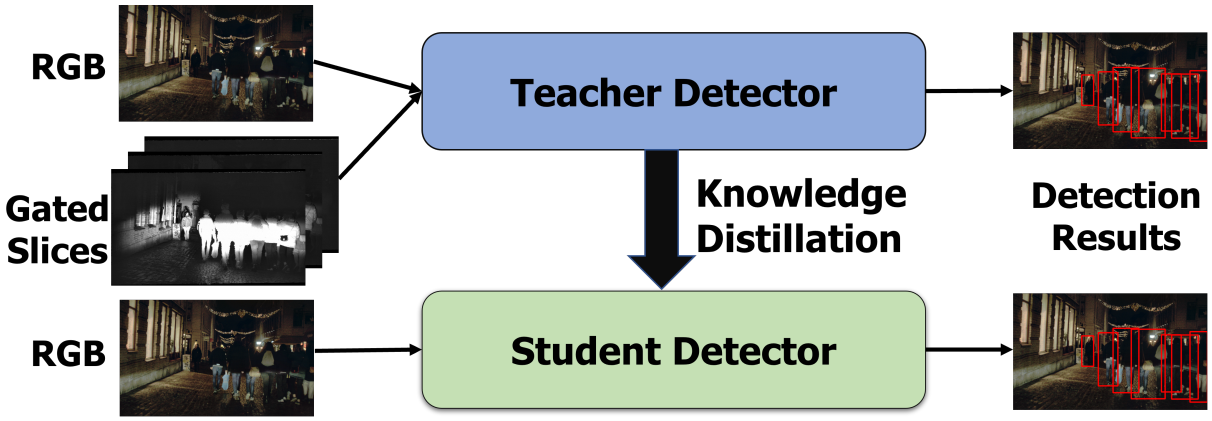


Figure 4.1: The proposed Cross Modality Knowledge Distillation (CMKD) framework to improve RGB-based pedestrian detection in low light and adverse weather conditions.

4.1 Proposed Framework

Most object detectors have two primary parts: a backbone to extract meaningful features, and a head that uses these features to detect objects. These features have a significant impact on the overall performance of the object detector. As a result, we focus on these features, and our goal is to make the student model generate features that are similar to the features of the teacher model. To achieve this objective, we first train the teacher detector using multiple modalities. Then, we freeze the teacher detector and begin training the student detector using KD loss in addition to the ground truth loss as shown in Figure 4.2. The KD loss makes the student backbone generate features that are similar to features generated by the teacher backbone. To compute the KD loss ($\mathcal{L}_{KD}$), we have attempted different forms of losses such as Mean Squared Error (MSE), Mean Absolute Error (MAE), Smooth $\mathcal{L}_1$ loss [5], and Kullback-Leibler (KL) divergence. Based on our experiments, we find out that using MSE as $\mathcal{L}_{KD}$ achieves the best results. Specifically, we compute $\mathcal{L}_{KD}$ using MSE in (4.1) to measure the Euclidean distance between features of the teacher and student models. Therefore, we refer to this proposed method as CMKD-MSE henceforth.

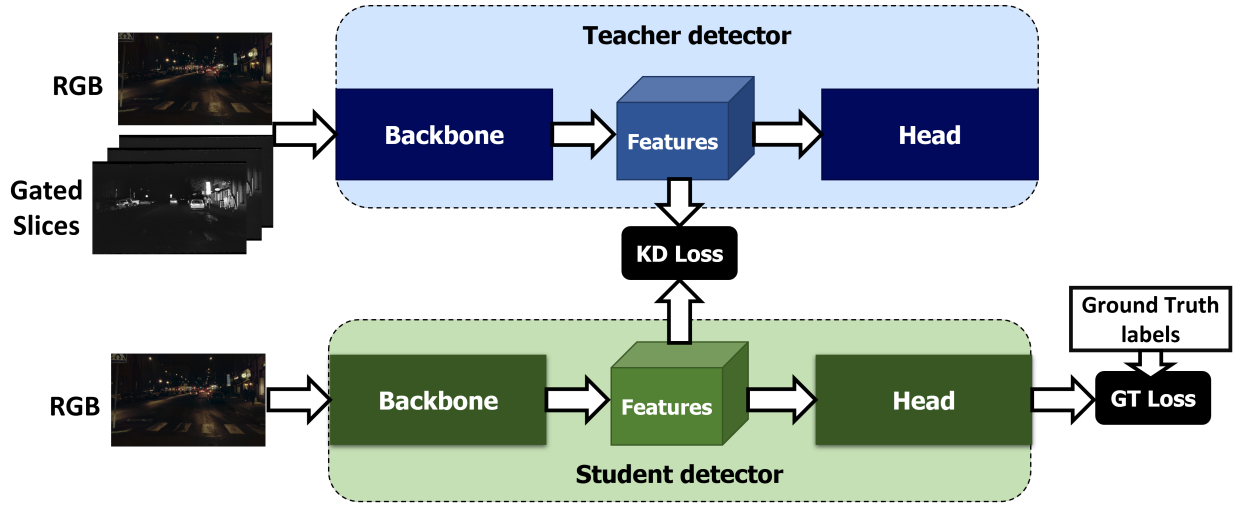


Figure 4.2: An overview of the first proposed method (CMKD-MSE) that utilizes Knowledge Distillation (KD) loss to make the student backbone produce features that are similar to the features of the teacher model. First, The teacher detector is trained using multiple modalities. Then, the trained teacher detection is frozen and the student detector is trained using KD loss in addition to the Ground Truth (GT) loss.

$$\mathcal{L}_{KD} = \frac{1}{NA} \sum_{i}^{N} \sum_{j}^{A} (F_i^S(j) - F_i^T(j))^2 \tag{4.1}$$

Where F_i^S and F_i^T are features of the i-th input image in the current batch for student and teacher models, respectively, N is the batch size, and A is the total number of activations in the feature tensor. During training, the student backbone is optimized to reduce the KD loss. Therefore, the total training loss for the student detector can be written as follows:

$$\mathcal{L} = \mathcal{L}_{GT} + \alpha \mathcal{L}_{KD} \tag{4.2}$$

where α is a weight parameter that balances a trade-off between the ground-truth detection loss ($\mathcal{L}_{GT}$) and the KD loss ($\mathcal{L}_{KD}$). Hence, the weight parameter controls the impact of knowledge distillation on the student backbone.

4.1.1 Adversarial Training

To further improve the performance, we propose another CMKD method that is based on adversarial training [41]. We build a Binary Classifier Network (BCN) to classify the features into teacher features and student features. For its architecture, we use two convolutional layers to reduce feature channels to a single one, followed by two fully connected layers to predict a final binary class probability (*i.e.* 0 for student and 1 for teacher). During training, we feed the extracted features from the backbones of both teacher and student models to the BCN. Then, BCN is optimized to differentiate between teacher features and student features by minimizing binary cross entropy loss ($\mathcal{L}_{BCE}$) in (4.3). On the other hand, the student backbone is optimized to maximize the loss in (4.3) to confuse the BCN. This makes the student backbone generate features that are similar to the teacher features. Consequently, the performance of the student detector will be significantly improved.

$$\mathcal{L}_{BCE} = -\frac{1}{N}\sum_{i}^{N}[t_i \ln p_i + (1 - t_i)\ln(1 - p_i)] \tag{4.3}$$

Here, t_i is the ground truth label of the binary classifier for the i-th input image in the current batch, with $t_i = 1$ for the teacher features and $t_i = 0$ for the student features. p_i is the predicted class probability of BCN for features of i-th training image, and N is the batch size.

To simultaneously solve the minimization and maximization problems, we adopt adversarial learning. In particular, we use a Gradient Reversal Layer (GRL) [34, 31] between the student backbone and BCN to achieve this contradictory objective. It should be noted that GRL is a bidirectional operator that is used to realize two different optimization objectives. In fact, GRL acts as an identity operator in the feed-forward direction. This leads to the standard objective of

minimizing the classification error when performing local backpropagation within the BCN. In contrast, GRL becomes a negative scalar (λ) for backpropagation toward the student backbone. Consequently, it leads to maximizing the classification error; and this maximization makes the student backbone generate features that are similar to the teacher features.

The total training loss for the student detector can be written as follows:

$$\mathcal{L} = \mathcal{L}_{GT} + \lambda \mathcal{L}_{BCE} \tag{4.4}$$

where λ is a negative scalar of GRL that balances a trade-off between the ground-truth detection loss($\mathcal{L}_{GT}$) and the binary cross entropy loss ($\mathcal{L}_{BCE}$). In that context, λ controls the impact of BCN on the student backbone. The overview of the proposed adversarial learning method is shown in Figure 4.3. Since this proposed method depends on adversarial training, we refer to it as CMKD-Adv henceforth.

For inference, the trained weights of the student detector in both proposed methods are used in the original detector without the need for the teacher detector or the binary classifier. Hence, our proposed framework will not increase the detector complexity during testing, which is an essential factor for many real-time applications such as ADAS and autonomous driving.

4.2 Experiments

4.2.1 Setup

We use the state-of-the-art Faster R-CNN [6] and SSD [9] as base detectors in our experiments. For backbone networks, we use ResNet50 [6] with Faster R-CNN, and VGG16 [67] with SSD.

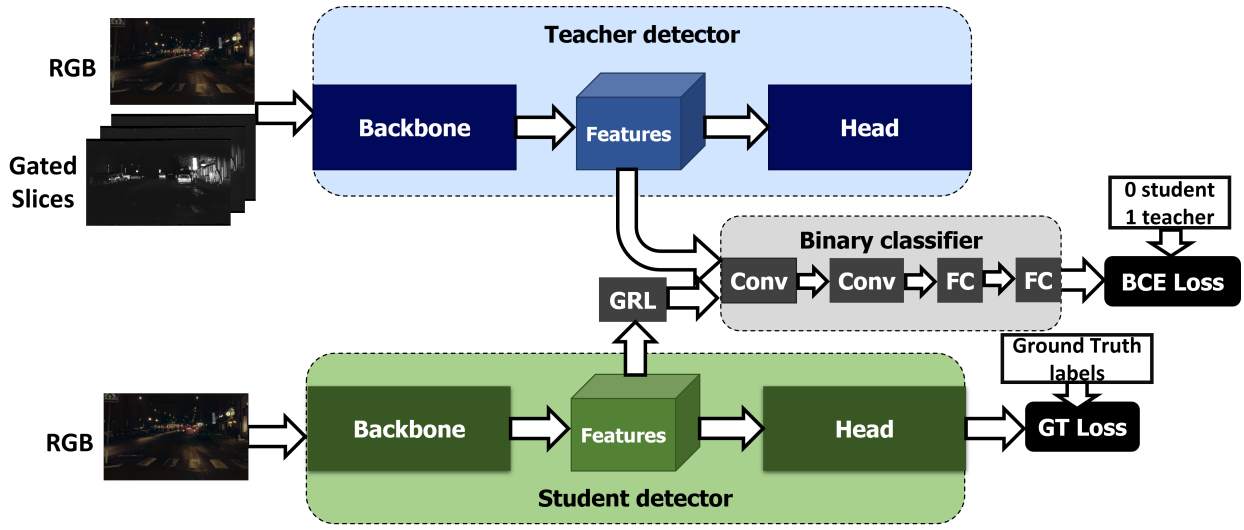


Figure 4.3: An overview of the second proposed method (CMKD-Adv) that utilizes adversarial learning to make student backbone produces features that are similar to teacher features. First, The teacher detector is trained using multiple modalities. Then, the trained teacher detector is frozen and the student detector is trained using adversarial learning in addition to the Ground Truth (GT) loss. The binary classifier is optimized to differentiate between teacher features and student features by minimizing Binary Cross Entropy (BCE) loss, while the student backbone is optimized to increase BCE loss due to Gradient Reversal Layer (GRL) to generate features that are similar to the teacher features. Conv: Convolutional layer, FC: Fully Connected layer.

It is worth noting that these backbones generate several scales of features that are used by the detector head to predict location and class of objects. Therefore, we apply our framework to all scales of features. Specifically, we compute KD loss in (4.1) and BCE loss in (4.3) for each scale individually, and then we take the average loss among the scales. This average loss is used to optimize the student backbone during training to acquire knowledge from the teacher detector.

There are some limitations in the availability of public datasets that provide matched pairs of cross-domain data with pedestrian annotations for driving scenarios under challenging conditions. Only the Seeing Through Fog dataset [27] publicly provides in addition to RGB images, corresponding images of other modalities such as thermal and gated images. In fact, these modalities substantially help to improve the detection performance in low light and adverse weather conditions. Therefore, in the experiments, we only use this dataset. Since only the gated images in the dataset contain both projections to the RGB frame and object annotations, we only add the gated images to the RGB images to train the teacher detector. To do so, we first project three slices of gated images from the dataset onto the image plane of the RGB camera. Examples of projected slices of gated images and corresponding RGB ones are shown in Figure 4.4. Then, we perform an early fusion to fuse the projected gated images with the corresponding RGB images by concatenating them in the channel dimension. Hence, the teacher detector uses a 6 channel tensor as input. By comparison, the student detector uses only the 3 channel RGB images as input.

In this work, we focus on detecting pedestrians in low light and adverse weather conditions. As a result, we only consider the pedestrian class and select the images at night in various weather conditions. We split the selected images into three sets: train, validation, and test. The number of images and annotated pedestrians for each set is provided in Table 4.1. We use the train set to optimize models during training. After each epoch of training, we evaluate the current performance

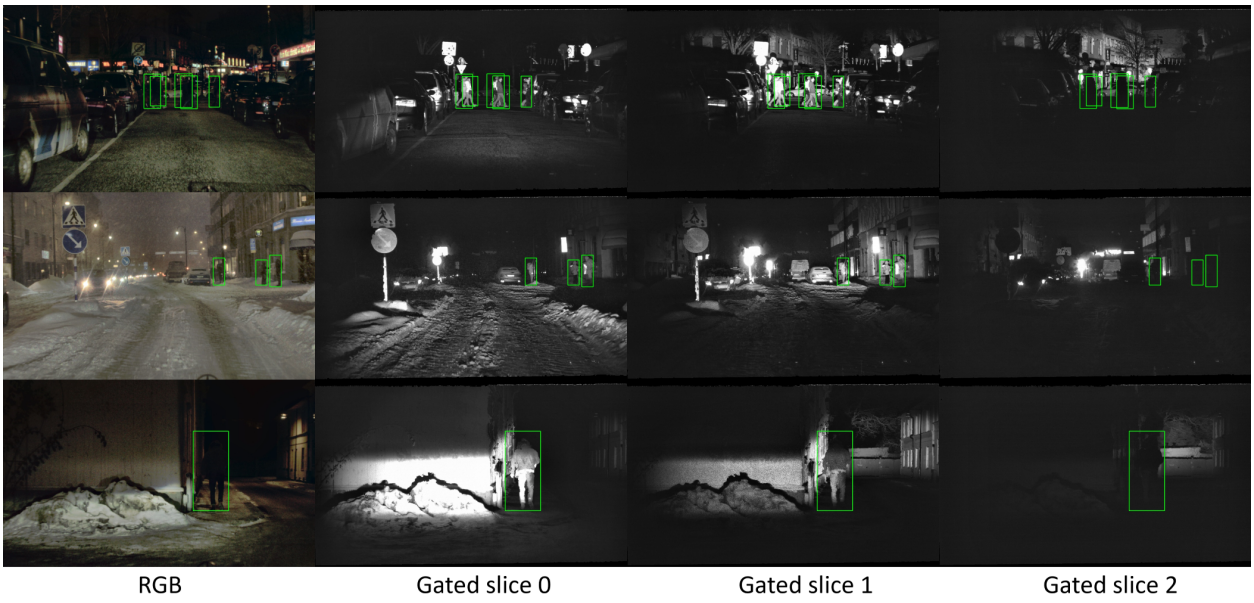


Figure 4.4: Examples of projected slices of gated images and corresponding RGB ones that are used in our experiments with ground-truth annotations for pedestrian class. The original images are from the Seeing Through Fog dataset [27].

Table 4.1: Number of images and annotated pedestrians in each set that is used in our experiments.

Set	Images	Annotated pedestrians
Train	3500	10377
Val	1167	3290
Test	1167	3211
Total	5834	16878

of the model using the validation set. Once the performance does not increase for 5 consecutive epochs, we stop the training. Eventually, we calculate the final performance of the trained model using the test set images that are unseen during training. This allows achieving generalized performance results. For an evaluation metric, we adopt the popular COCO Average Precision (AP) [110].

We implement the proposed framework using PyTorch [111]. All models are initialized using pre-trained weights from ImageNet. We use the SGD optimizer to fine-tune the model for our

detection task. All used images are resized to have a height of 500 pixels. The initial learning rate is set to 0.005, and is scaled by 0.1 every 5 epochs. Furthermore, we set the batch size to 4, the momentum to 0.9, and weight decay to 0.0005. For the hyper-parameters α and λ in our proposed methods, we select them based on the trial and error technique.

4.2.2 Results and Discussion

The performance results of the proposed methods as compared to the baseline and teacher models are summarized in Table 4.2. Only RGB images in the test set are used to calculate the performance, except for the teacher model which requires both RGB and gated images for training and testing. The baseline model is the original Faster R-CNN and SSD detectors that are trained using only RGB images without any KD technique for comparison. Based on the results, both of our methods significantly improve the detection performance relative to the baseline model. Equally important, they minimize a performance gap between the teacher and baseline models. For example with Faster R-CNN detector, CMKD-MSE reduces the gap in AP by 36%, while CMKD-Adv reduces it by 55% as shown in Figure 4.5. For all metrics, CMKD-Adv achieves the best results relative to the baseline.

It is worth noting that we do not report the performance of previous methods [51, 52, 53, 54, 55, 56] that used KD for object detection. This is because none of these methods used CMKD to improve object detection in low light and adverse weather conditions. In fact, all these methods use KD for model compression (*i.e.* they transfer the knowledge from a large complex model to a small simple one) which is different from the objective of this work.

To further demonstrate the improvements from our proposed framework, we show in Figure 4.6 visual detection examples of our framework as compared to the baseline. These examples

Table 4.2: Performance results of the proposed framework based on the metrics in COCO dataset evaluator [110]. Both our methods (CMKD-MSE and CMKD-Adv) improve the detection performance relative to the baseline model. For AP_{75} metric, they achieve better performance than the teacher model. For all metrics, CMKD-Adv obtains the best results.

	Faster R-CNN - ResNet50						SSD - VGG16					
Model	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Teacher	27.5	63.0	18.6	8.5	27.5	43.6	16.3	44.2	7.4	3.7	16.8	28.2
Baseline	24.2	55.5	17.3	4.6	23.0	45.0	12.5	34.7	6.8	2.5	11.2	27.0
CMKD-MSE	25.4	56.4	19.0	5.1	24.9	45.2	13.8	36.4	6.7	2.5	12.9	28.0
CMKD-Adv	**26.0**	**56.6**	**19.7**	**5.5**	**25.6**	**45.8**	**14.5**	**39.2**	**7.8**	**2.9**	**13.8**	**28.9**

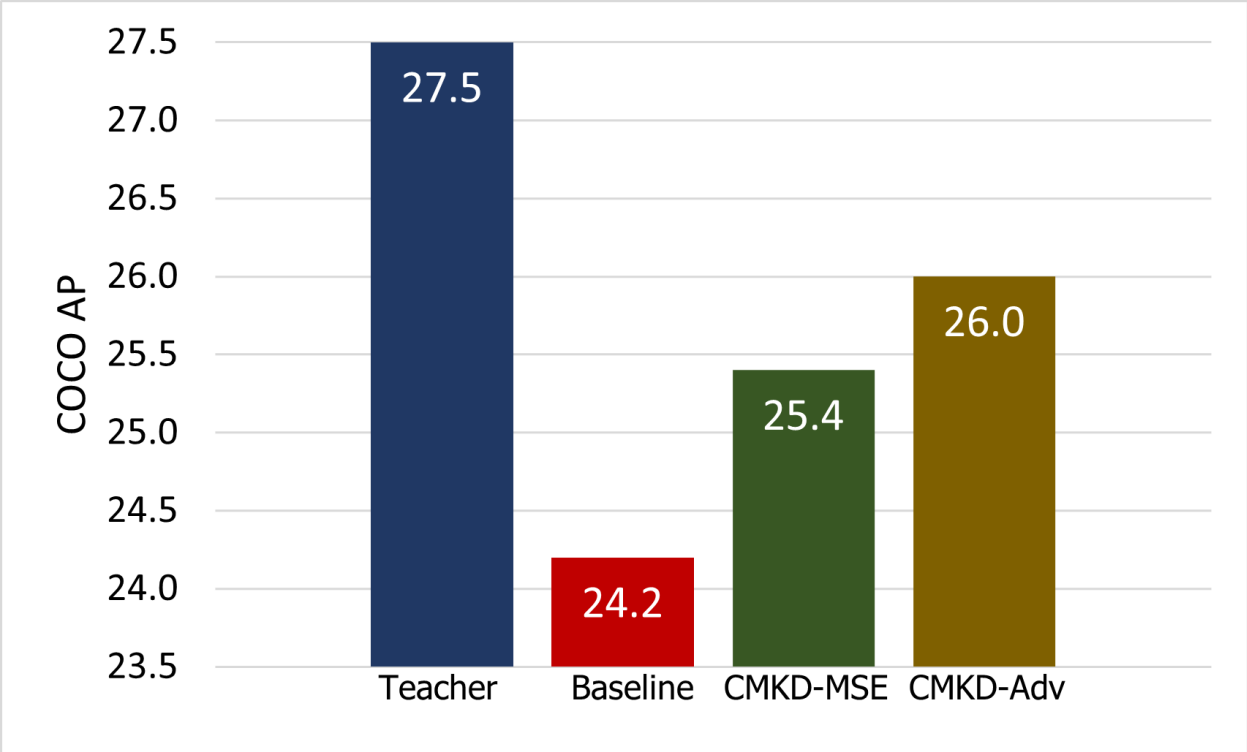


Figure 4.5: The performance of the proposed framework based on COCO AP [110] using only RGB images in the test set except for **Teacher** model which is trained and tested using both RGB and gated images. **CMKD-MSE** is the first proposed method that uses MSE in (4.1) to transfer knowledge, while **CMKD-Adv** is the second proposed method that transfers knowledge using adversarial training. **Baseline** is the original Faster R-CNN detector that is trained using only RGB images.

show that our methods (CMKD-MSE and CMKD-Adv) successfully detect pedestrians that the baseline fails to detect in snowy weather and low light conditions. Moreover, Figure 4.7 shows examples where the baseline suffers from false positive cases, while our methods eliminate these false positives which contribute to improve the performance.

It is worth noting that fundamental limitations to CMKD can exist if the information is simply missing from the single modality data. Our framework does not reach to the full performance of the teacher model because many annotated pedestrians in the dataset are totally dark in RGB images. In other words, there are not enough pixels in the RGB images to represent many pedestrians. For this reason, our framework that only uses RGB images as inputs cannot detect these specific pedestrians as the examples shown in Figure 4.8. By comparison, these pedestrians are clear and easy to detect in gated images. Therefore, the teacher model successfully detects them because it uses both RGB and gated images as inputs.

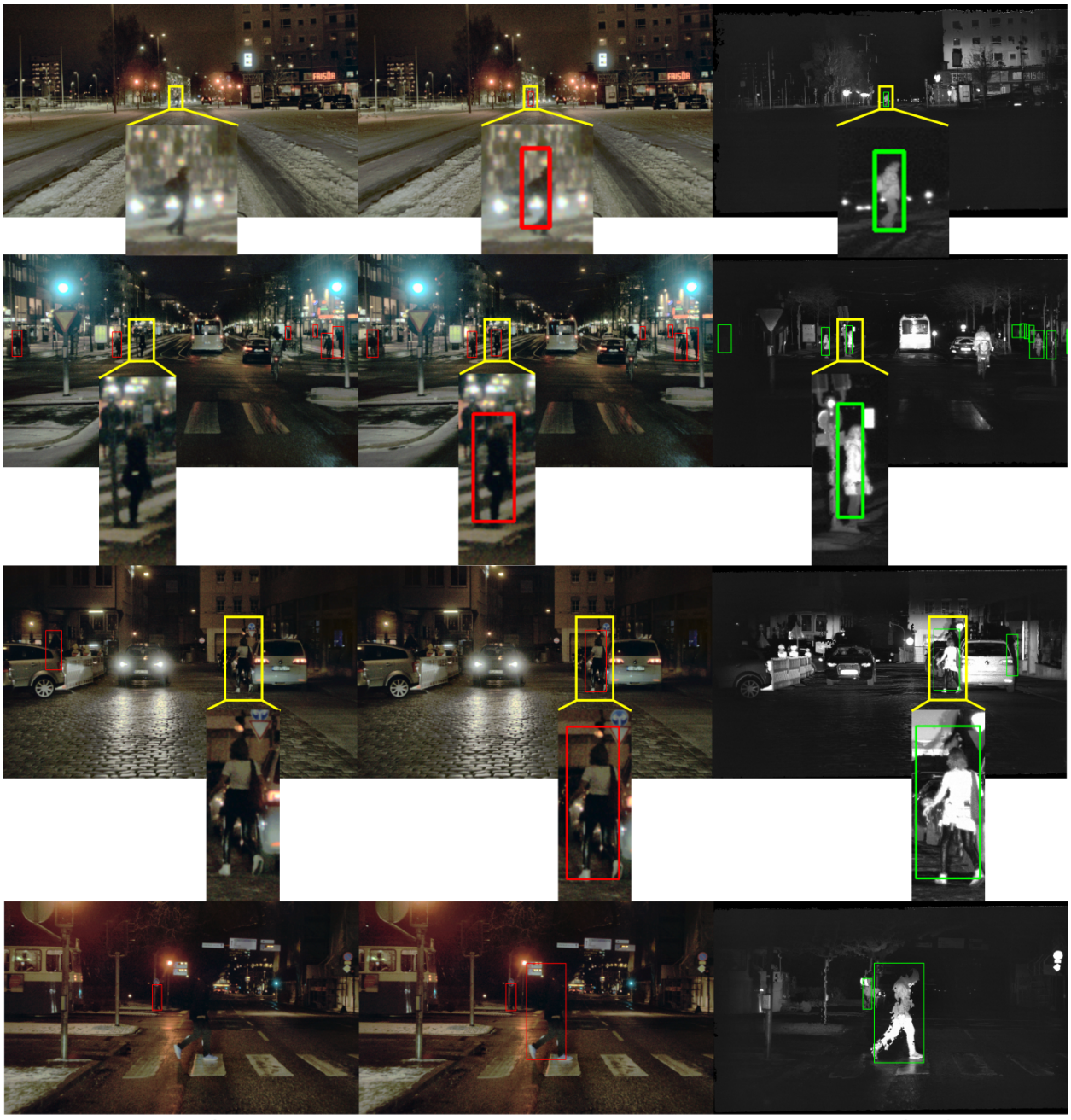


Figure 4.6: Visual detection examples of our framework as compared to the baseline. The examples show that CMKD-MSE (top row) and CMKD-Adv (bottom rows) successfully detect pedestrians that the baseline fails to detect in snowy weather and low light conditions.

Figure 4.7: Visual detection examples of our framework as compared to the baseline. In these examples, the baseline suffers from instances of false positives, while our methods: CMKD-MSE (top middle image) and CMKD-Adv (bottom middle image) eliminate these false positives.

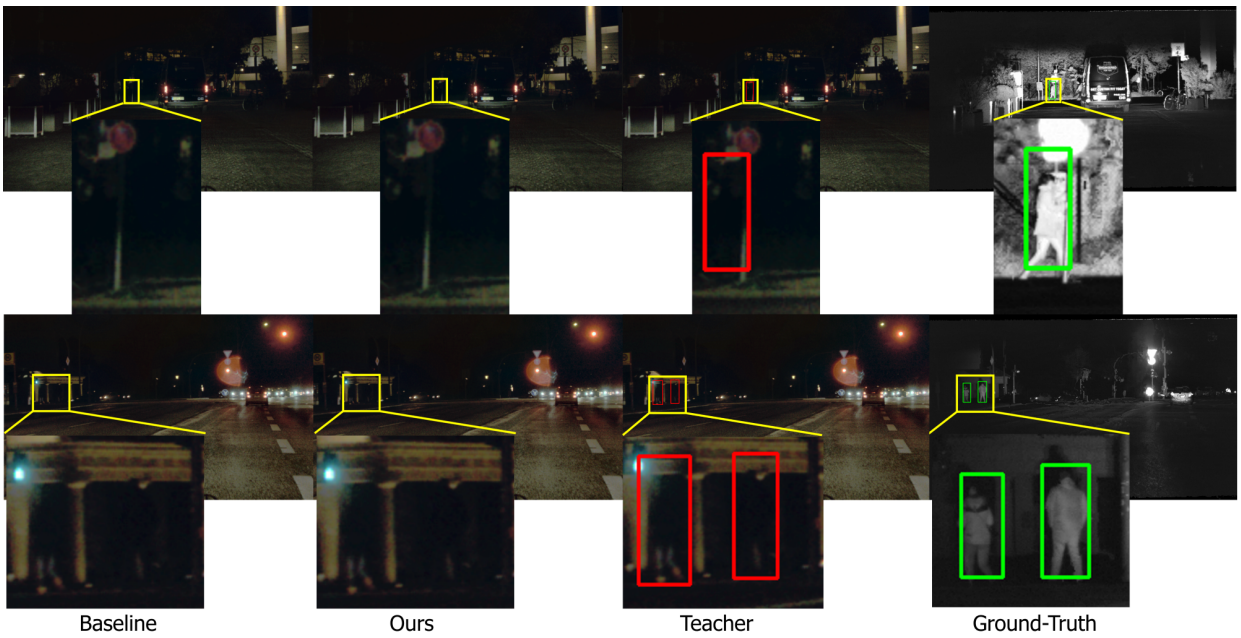


Figure 4.8: Visual detection examples of our framework as compared to the baseline and teacher models. The examples show that pedestrians are totally dark in RGB images, and there are not enough pixels to represent them. Therefore, it is challenging for the baseline and our framework which only use RGB images as inputs to detect them. While the teacher model successfully detects them since it uses both RGB and gated images as inputs.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

In this dissertation, we first outlined state-of-the-art frameworks for object detection, deraining, image-to-image translation, and domain adaptation. Moreover, we highlighted crucial results regarding current methods in terms of their performance under rainy weather conditions. In particular, there is an overarching consistent message regarding the limitations of these techniques in handling and mitigating the impact of rain for visuals captured by moving vehicles. However, we believe that generative models and domain adaptation could still play a crucial role in training object detection methods to be more robust and resilient under challenging conditions. As a result, we proposed the GDA framework that integrates generative model-based image translation with domain adaptation object detection to improve the detection performance of objects in a driving environment under realistic rainy conditions. In particular, we generated visuals that are representatives of a target challenging domain by utilizing unsupervised Image-to-Image Translation (I2IT). Then, these generated visuals and unlabeled target domain data were used to train a domain adaptive object detection method. Our results showed that the proposed GDA achieved significant improvements when tested under real rainy weather conditions in comparison with state-of-the-art Domain Adaptive Faster R-CNN or the baseline generative model-based I2IT training method.

In addition, we proposed a multiscale domain adaptation framework for the popular state-of-the-art real-time object detector YOLO. Specifically, under our MS-DAYOLO architecture, we

applied domain adaptation to three different scale features within the YOLO feature extractor that are fed to the next stage. In addition to the baseline architecture of a multiscale domain adaptive network, we developed three various deep learning architectures to produce more robust domain invariant features that reduce the impact of domain shift. The proposed architectures include progressive feature reduction, unified domain classifier, and the integrated architecture that combines the benefits of progressive feature reduction and unified domain classifier strategies for improving the overall detection performance under the target domain. Based on various experimental results, our proposed MS-DAYOLO framework can successfully adapt YOLO to target domains without the need to annotate the objects found in the data. Furthermore, the proposed MS-DAYOLO architectures outperformed state-of-the-art YOLOv4 and other exciting approaches that are based on Faster R-CNN object detector under diverse testing scenarios for autonomous driving applications.

Furthermore, we proposed a novel framework that improved the performance of RGB-only pedestrian detection in low light and adverse weather conditions by employing cross modality knowledge distillation. In particular, we transferred the knowledge of a teacher detector that uses both RGB and gated images to a student RGB-only detector. Specifically, we developed two methods within the proposed framework that forced the student model to generate features that are similar to features of the teacher model. Our experiments show that incorporating adversarial training with knowledge distillation improves the detection performance by a significant margin, and reduces the performance gap between teacher and baseline models by 55%. Equally important, the proposed framework can be applied to other state-of-the-art detectors.

Overall, we believe that the proposed frameworks in this dissertation will increase the efficiency and safety of autonomous systems via improving the detection performance of objects under challenging conditions.

5.2 Future Work

Object-based Domain adaptation: Domain adaptation of MS-DAYOLO in Chapter 3 is an image-level adaptation because it is applied to all features of the input image. This image-level strategy helps to reduce the domain shift due to global image differences such as image scale, image style, and illumination. However, local instance differences such as object appearance, size, and viewpoint are not addressed in MS-DAYOLO. Therefore, we believe applying domain adaptation to local features of objects in the YOLO framework will further improve the detection performance. Specifically, domain adaptation could be applied to the features that are associated with each object in an input image. This instance-level adaptation should help to reduce the domain shift that is due to the local instance differences.

After the input image features are generated by the YOLO backbone network, we plan to extract the features that correspond to each object in the image as shown in Figure 5.1. Then, we can apply domain adaptation to the extracted features by feeding them to an *Object-Based Domain Adaptive Network* (OB-DAN).

In domain adaptation, ground-truth bounding boxes of target data are generally not available. And hence, it is not clear which features from within the image should be used for instance-level domain adaptation. To solve this problem, an already trained YOLO can be used to detect objects, and these objects can be used to extract the instance-level features. However, these detections do not represent the ”ground-truth”, and therefore such detections may be highly inaccurate, which could lead to a degradation in performance. Alternatively, we propose to perform the following steps:

- Train MS-DAYOLO, which represents the global image-level domain adaptation for the YOLO object detector.

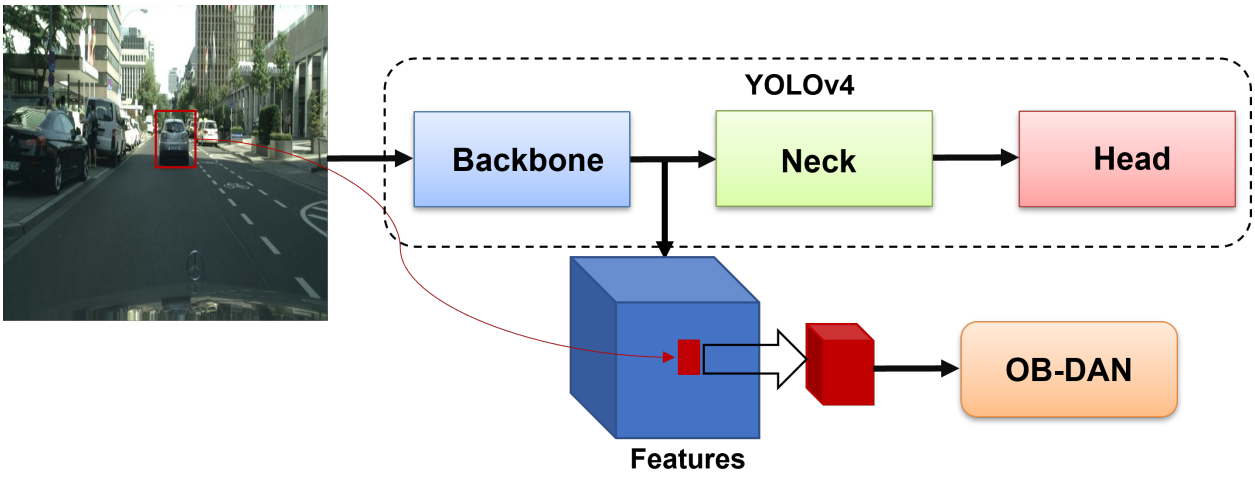


Figure 5.1: Example of extracting features that are corresponding to the car in the middle of the image, the extracted features are fed to Object-Based Domain Adaptive Network (OB-DAN).

- Use the trained MS-DAYOLO to predict bounding boxes for target data.
- Filter the predicted bounding boxes based on their confidence scores. In particular, bounding boxes with confidence scores larger than a specific threshold can be selected. This technique is usually called *pseudo labeling*.
- Train the object-based domain adaptive YOLO, which is initialized with the trained weights of MS-DAYOLO, using ground truth bounding boxes for source data and the filtered predicted bounding boxes for target data.

The proposed framework is summarized in Figure 5.2. We plan to conduct several experiments for this framework with different datasets and scenarios to validate its performance.

Class-Wise Domain Adaptation: Instead of applying domain adaptation to all objects independent of their classes as explained in Object-based Domain adaptation, we believe that applying domain adaptation separately to each class can provide further improvements. Specifically, we in-

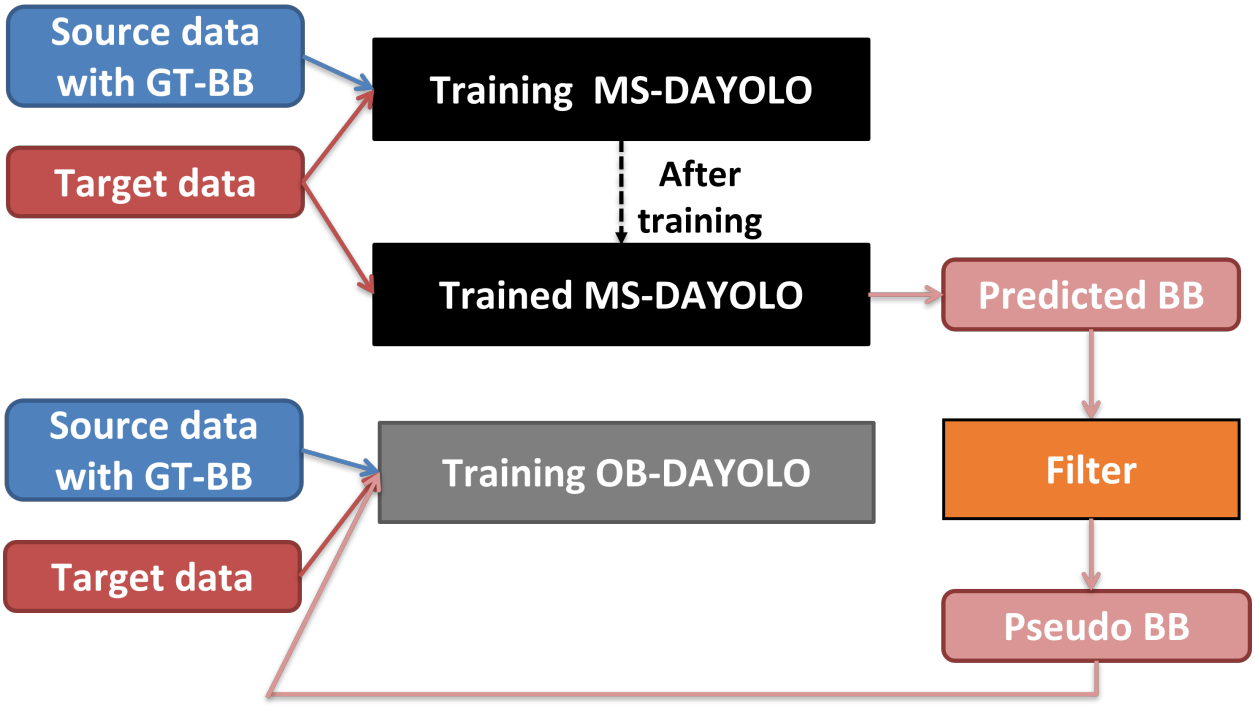


Figure 5.2: The proposed framework for Object-Based Domain Adaptive YOLO (OB-DAYOLO). GT: Ground Truth, BB: Bounding Boxes.

tend to develop an individual object-based Domain Adaptive Network (DAN) that is optimized for each class as shown in Figure 5.3. This will prevent aligning distributions of object features from different classes jointly. For example, aligning the distribution of features for the car class with the distribution of features for the pedestrian class in a different domain, may drop the detection performance. Therefore, aligning the distributions of local instance features within each class can improve the detection performance, especially with a dataset that has many classes.

Extension of Cross Modality Knowledge Distillation (CMKD): The CMKD framework in Chapter 4 can be extended in several directions. In addition to RGB camera, other sensor modalities (*e.g.* thermal imaging, lidar, and radar) that improve the detection performance in low light and adverse weather conditions, can be used in the teacher model. In the CMKD framework, we used

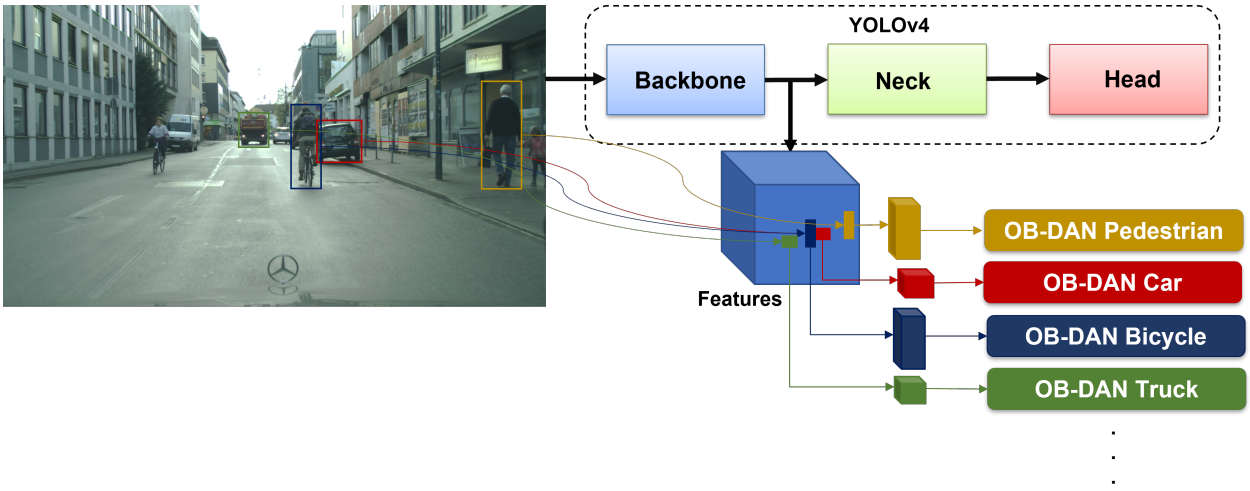


Figure 5.3: The proposed Class-Wise Domain Adaptation for YOLO object detector. Object-Based Domain Adaptive Network (OB-DAN) is developed for each class in the used dataset.

early (data-level) fusion to fuse RGB and gated images. On the hand, there are other techniques such as late (decision-level) fusion and intermediate (features-level) fusion that can be employed to fuse multiple modalities in the teacher model. These techniques can be used to improve the performance of the teacher model, which subsequently should improve the student model.

New performance metric: It is important to emphasize that the current metrics for detection performance (Pascal mAP and COCO AP) may not reflect the true effectiveness of the proposed frameworks. In driving environments, we observe that detecting certain objects are more crucial than others. For instance, detecting pedestrians that are crossing the road or walking nearby the vehicle is more critical than detecting pedestrians that are walking at safe distances on well-protected sidewalks. To illustrate this observation, there are two ground-truth annotated pedestrians in the scene of the bottom row of Figure 4.6 that are used to compute COCO AP. However, for this particular image-frame, the pedestrian that crosses the road is more important to detect than the pedestrian standing back on the sidewalk. To address this issue, we believe it is essential for future

research to develop a new performance metric rather than COCO mAP or Pascal mAP to evaluate the effectiveness of detecting critical objects instead of all objects in a scene.

BIBLIOGRAPHY

[1] C. Chen, A. Seff, A. Kornhauser, and J. Xiao, "Deepdriving: Learning affordance for direct perception in autonomous driving," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 2722–2730.

[2] B. Wu, F. Iandola, P. H. Jin, and K. Keutzer, "Squeezedet: Unified, small, low power fully convolutional neural networks for real-time object detection for autonomous driving," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2017, pp. 129–137.

[3] M. Teichmann, M. Weber, M. Zoellner, R. Cipolla, and R. Urtasun, "Multinet: Real-time joint semantic reasoning for autonomous driving," in *IEEE Intelligent Vehicles Symposium (IV)*, 2018, pp. 1013–1020.

[4] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 580–587.

[5] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440–1448.

[6] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, pp. 1137–1149, June 2017.

[7] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar, "Focal loss for dense object detection," in *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[8] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016, pp. 779–788.

[9] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *European conference on computer vision*. Springer, 2016, pp. 21–37.

[10] J. Dai, Y. Li, K. He, and J. Sun, "R-fcn: Object detection via region-based fully convolutional networks," in *Advances in neural information processing systems*, 2016, pp. 379–387.

[11] J. Redmon and A. Farhadi, "Yolo9000: better, faster, stronger," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 7263–7271.

[12] ——, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[13] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "Yolov4: Optimal speed and accuracy of object detection," *arXiv preprint arXiv:2004.10934*, 2020.

[14] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2961–2969.

[15] K. He, J. Sun, and X. Tang, "Single image haze removal using dark channel prior," *IEEE transactions on pattern analysis and machine intelligence*, vol. 33, no. 12, pp. 2341–2353, 2010.

[16] X. Liu, Y. Ma, Z. Shi, and J. Chen, "Griddehazenet: Attention-based multi-scale network for image dehazing," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 7314–7323.

[17] C. Guo, C. Li, J. Guo, C. C. Loy, J. Hou, S. Kwong, and R. Cong, "Zero-reference deep curve estimation for low-light image enhancement," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 1780–1789.

[18] H. Dong, J. Pan, L. Xiang, Z. Hu, X. Zhang, F. Wang, and M.-H. Yang, "Multi-scale boosted dehazing network with dense feature fusion," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 2157–2167.

[19] X. Qin, Z. Wang, Y. Bai, X. Xie, and H. Jia, "Ffa-net: Feature fusion attention network for single image dehazing," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 07, 2020, pp. 11 908–11 915.

[20] S. Li, I. B. Araujo, W. Ren, Z. Wang, E. K. Tokuda, R. H. Junior, R. Cesar-Junior, J. Zhang, X. Guo, and X. Cao, "Single image deraining: A comprehensive benchmark analysis," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 3838–3847.

[21] M. Hnewa and H. Radha, "Object detection under rainy conditions for autonomous vehicles: A review of state-of-the-art and emerging techniques," *IEEE Signal Processing Magazine*, vol. 38, no. 1, pp. 53–67, 2021.

[22] M. Krišto, M. Ivasic-Kos, and M. Pobar, "Thermal object detection in difficult weather conditions using yolo," *IEEE access*, vol. 8, pp. 125 459–125 476, 2020.

[23] R. Blin, S. Ainouz, S. Canu, and F. Meriaudeau, "Road scenes analysis in adverse weather conditions by polarization-encoded images and adapted deep learning," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. IEEE, 2019, pp. 27–32.

[24] P. Tumas, A. Nowosielski, and A. Serackis, "Pedestrian detection in severe weather conditions," *IEEE Access*, vol. 8, pp. 62 775–62 784, 2020.

[25] G. Liao, W. Gao, G. Li, J. Wang, and S. Kwong, "Cross-collaborative fusion-encoder network for robust rgb-thermal salient object detection," *IEEE Transactions on Circuits and Systems for Video Technology*, 2022.

[26] F. Julca-Aguilar, J. Taylor, M. Bijelic, F. Mannan, E. Tseng, and F. Heide, "Gated3d: Monocular 3d object detection from temporal illumination cues," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 2938–2948.

[27] M. Bijelic, T. Gruber, F. Mannan, F. Kraus, W. Ritter, K. Dietmayer, and F. Heide, "Seeing through fog without seeing fog: Deep multimodal sensor fusion in unseen adverse weather," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 11 679–11 689.

[28] S. S. Chaturvedi, L. Zhang, and X. Yuan, "Pay "attention" to adverse weather: Weather-aware attention-based object detection," *arXiv preprint arXiv:2204.10803*, 2022.

[29] L. Duan, I. W. Tsang, and D. Xu, "Domain transfer multiple kernel learning," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 3, pp. 465–479, 2012.

[30] B. Kulis, K. Saenko, and T. Darrell, "What you saw is not what you get: Domain adaptation using asymmetric kernel transforms," in *CVPR 2011*. IEEE, 2011, pp. 1785–1792.

[31] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky, "Domain-adversarial training of neural networks," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 2096–2030, 2016.

[32] E. Tzeng, J. Hoffman, K. Saenko, and T. Darrell, "Adversarial discriminative domain adaptation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 7167–7176.

[33] M. Long, H. Zhu, J. Wang, and M. I. Jordan, "Unsupervised domain adaptation with residual transfer networks," in *Advances in neural information processing systems*, 2016, pp. 136–144.

[34] Y. Ganin and V. Lempitsky, "Unsupervised domain adaptation by backpropagation," in *Proceedings of the 32nd International Conference on International Conference on Machine Learning-Volume 37*. JMLR. org, 2015, pp. 1180–1189.

[35] W. Li, F. Li, Y. Luo, P. Wang *et al.*, "Deep domain adaptive object detection: A survey," in *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, 2020, pp. 1808–1813.

[36] V. F. Arruda, T. M. Paixão, R. F. Berriel, A. F. De Souza, C. Badue, N. Sebe, and T. Oliveira-Santos, "Cross-domain car detection using unsupervised image-to-image translation: From day to night," in *2019 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2019, pp. 1–8.

[37] C.-T. Lin, "Cross domain adaptation for on-road object detection using multimodal structure-consistent image-to-image translation," in *2019 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2019, pp. 3029–3030.

[38] T. Guo, C. P. Huynh, and M. Solh, "Domain-adaptive pedestrian detection in thermal images," in *2019 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2019, pp. 1660–1664.

[39] C. Devaguptapu, N. Akolekar, M. M Sharma, and V. N Balasubramanian, "Borrow from anywhere: Pseudo multi-modal object detection in thermal imagery," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2019.

[40] N. Inoue, R. Furuta, T. Yamasaki, and K. Aizawa, "Cross-domain weakly-supervised object detection through progressive domain adaptation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 5001–5009.

[41] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[42] Y. Chen, W. Li, C. Sakaridis, D. Dai, and L. Van Gool, "Domain adaptive faster r-cnn for object detection in the wild," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 3339–3348.

[43] X. Zhu, J. Pang, C. Yang, J. Shi, and D. Lin, "Adapting object detectors via selective cross-domain alignment," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 687–696.

[44] T. Wang, X. Zhang, L. Yuan, and J. Feng, "Few-shot adaptive faster r-cnn," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 7173–7182.

[45] K. Saito, Y. Ushiku, T. Harada, and K. Saenko, "Strong-weak distribution alignment for adaptive object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 6956–6965.

[46] Z. He and L. Zhang, "Multi-adversarial faster-rcnn for unrestricted object detection," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 6668–6677.

[47] V. A. Sindagi, P. Oza, R. Yasarla, and V. M. Patel, "Prior-based domain adaptive object detection for hazy and rainy conditions," in *European Conference on Computer Vision*. Springer, 2020, pp. 763–780.

[48] C. Zhuang, X. Han, W. Huang, and M. Scott, "ifan: Image-instance full alignment networks for adaptive object detection," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 07, 2020, pp. 13 122–13 129.

[49] G. Zhao, G. Li, R. Xu, and L. Lin, "Collaborative training between region proposal localization and classification for domain adaptive object detection," in *European Conference on Computer Vision*. Springer, 2020, pp. 86–102.

[50] H.-K. Hsu, C.-H. Yao, Y.-H. Tsai, W.-C. Hung, H.-Y. Tseng, M. Singh, and M.-H. Yang, "Progressive domain adaptation for object detection," in *The IEEE Winter Conference on Applications of Computer Vision*, 2020, pp. 749–757.

[51] G. Chen, W. Choi, X. Yu, T. Han, and M. Chandraker, "Learning efficient object detection models with knowledge distillation," *Advances in neural information processing systems*, vol. 30, 2017.

[52] Q. Li, S. Jin, and J. Yan, "Mimicking very efficient network for object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 6356–6364.

[53] T. Wang, L. Yuan, X. Zhang, and J. Feng, "Distilling object detectors with fine-grained feature imitation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4933–4942.

[54] J. Guo, K. Han, Y. Wang, H. Wu, X. Chen, C. Xu, and C. Xu, "Distilling object detectors via decoupled features," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 2154–2164.

[55] X. Dai, Z. Jiang, Z. Wu, Y. Bao, Z. Wang, S. Liu, and E. Zhou, "General instance distillation for object detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 7842–7851.

[56] G. Li, X. Li, Y. Wang, S. Zhang, Y. Wu, and D. Liang, "Knowledge distillation for object detection via rank mimicking and prediction-guided feature imitation," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 2, 2022, pp. 1306–1313.

[57] K. Garg and S. K. Nayar, "Detection and removal of rain from videos," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 1, 2004, pp. I–I.

[58] L. Kang, C. Lin, and Y. Fu, "Automatic single-image-based rain streaks removal via image decomposition," *IEEE Transactions on Image Processing*, vol. 21, no. 4, pp. 1742–1755, April 2012.

[59] W. Yang, R. T. Tan, J. Feng, J. Liu, Z. Guo, and S. Yan, "Deep joint rain detection and removal from a single image," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 1357–1366.

[60] X. Fu, J. Huang, D. Zeng, Y. Huang, X. Ding, and J. Paisley, "Removing rain from single images via a deep detail network," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 3855–3863.

[61] R. Qian, R. T. Tan, W. Yang, J. Su, and J. Liu, "Attentive generative adversarial network for raindrop removal from a single image," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2482–2491.

[62] D. Ren, W. Zuo, Q. Hu, P. Zhu, and D. Meng, "Progressive image deraining networks: a better and simpler baseline," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 3937–3946.

[63] T. Wang, X. Yang, K. Xu, S. Chen, Q. Zhang, and R. W. Lau, "Spatial attentive single-image deraining with a high quality real rain dataset," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 12 270–12 279.

[64] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE transactions on image processing*, vol. 13, no. 4, pp. 600–612, 2004.

[65] M. Hnewa and H. Radha, "Integrated generative-model domain-adaptation for object detection under challenging conditions," in *2022 IEEE 95th Vehicular Technology Conference:(VTC2022-Spring)*, pp. 1–5.

[66] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *European conference on computer vision*. Springer, 2014, pp. 818–833.

[67] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *3rd International Conference on Learning Representations, ICLR*, 2015.

[68] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models," *IEEE transactions on pattern analysis and machine intelligence*, vol. 32, no. 9, pp. 1627–1645, 2009.

[69] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[70] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[71] G. Neuhold, T. Ollmann, S. Rota Bulò, and P. Kontschieder, "The mapillary vistas dataset for semantic understanding of street scenes," in *International Conference on Computer Vision (ICCV)*, 2017. [Online]. Available: https://www.mapillary.com/dataset/vistas

[72] F. Yu, W. Xian, Y. Chen, F. Liu, M. Liao, V. Madhavan, and T. Darrell, "Bdd100k: A diverse driving video database with scalable annotation tooling," *arXiv preprint arXiv:1805.04687*, 2018.

[73] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, "nuscenes: A multimodal dataset for autonomous driving," *arXiv preprint arXiv:1903.11027*, 2019.

[74] P. Rousseau, V. Jolivet, and D. Ghazanfarpour, "Realistic real-time rain rendering," *Computers & Graphics*, vol. 30, no. 4, pp. 507–518, 2006.

[75] K. Garg and S. K. Nayar, "Photorealistic rendering of rain streaks," *ACM Transactions on Graphics (TOG)*, vol. 25, no. 3, pp. 996–1002, 2006.

[76] *Cycore Rainfall simulation, Adobe After Effects CC 2019*, Adobe Inc., Sab Jose, CA, USA, 2019. [Online]. Available: https://www.adobe.com/products/aftereffects.html

[77] M. Everingham and J. Winn, "The pascal visual object classes challenge 2012 development kit," *Pattern Analysis, Statistical Modelling and Computational Learning, Tech. Rep*, 2011.

[78] M.-Y. Liu, T. Breuel, and J. Kautz, "Unsupervised image-to-image translation networks," in *Advances in neural information processing systems*, 2017, pp. 700–708.

[79] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[80] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[81] S. Xingjian, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, and W.-c. Woo, "Convolutional lstm network: A machine learning approach for precipitation nowcasting," in *Advances in neural information processing systems*, 2015, pp. 802–810.

[82] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1125–1134.

[83] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2223–2232.

[84] J.-Y. Zhu, R. Zhang, D. Pathak, T. Darrell, A. A. Efros, O. Wang, and E. Shechtman, "Toward multimodal image-to-image translation," in *Advances in Neural Information Processing Systems*, 2017, pp. 465–476.

[85] X. Huang, M.-Y. Liu, S. Belongie, and J. Kautz, "Multimodal unsupervised image-to-image translation," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 172–189.

[86] Z. Yi, H. Zhang, P. Tan, and M. Gong, "Dualgan: Unsupervised dual learning for image-to-image translation," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2849–2857.

[87] Y. Choi, M. Choi, M. Kim, J.-W. Ha, S. Kim, and J. Choo, "Stargan: Unified generative adversarial networks for multi-domain image-to-image translation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 8789–8797.

[88] C. Ledig, L. Theis, F. Huszar, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi, "Photo-realistic single image super-resolution using a generative adversarial network," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[89] E. Tzeng, J. Hoffman, N. Zhang, K. Saenko, and T. Darrell, "Deep domain confusion: Maximizing for domain invariance," *arXiv preprint arXiv:1412.3474*, 2014.

[90] M. Long, Y. Cao, J. Wang, and M. Jordan, "Learning transferable features with deep adaptation networks," in *International Conference on Machine Learning*, 2015, pp. 97–105.

[91] R. Xie, F. Yu, J. Wang, Y. Wang, and L. Zhang, "Multi-level domain adaptive learning for cross-domain detection," in *The IEEE International Conference on Computer Vision (ICCV) Workshops*, Oct 2019.

[92] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky, "Domain-adversarial training of neural networks," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 2096–2030, 2016.

[93] M. Hnewa and H. Radha, "Multiscale domain adaptive yolo for cross-domain object detection," in *IEEE International Conference on Image Processing (ICIP)*, 2021, pp. 3323–3327.

[94] C. Sakaridis, D. Dai, and L. Van Gool, "Semantic foggy scene understanding with synthetic data," *International Journal of Computer Vision*, vol. 126, no. 9, pp. 973–992, 2018.

[95] P. Sun, H. Kretzschmar, X. Dotiwalla, A. Chouard, V. Patnaik, P. Tsui, J. Guo, Y. Zhou, Y. Chai, B. Caine *et al.*, "Scalability in perception for autonomous driving: Waymo open dataset," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 2446–2454.

[96] M. Khodabandeh, A. Vahdat, M. Ranjbar, and W. G. Macready, "A robust learning approach to domain adaptive object detection," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 480–490.

[97] Q. Cai, Y. Pan, C.-W. Ngo, X. Tian, L. Duan, and T. Yao, "Exploring object relation in mean teacher for cross-domain detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 11 457–11 466.

[98] T. Kim, M. Jeong, S. Kim, S. Choi, and C. Kim, "Diversify and match: A domain adaptive representation learning paradigm for object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 12 456–12 465.

[99] M. Xu, H. Wang, B. Ni, Q. Tian, and W. Zhang, "Cross-domain detection via graph-induced prototype alignment," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 12 355–12 364.

[100] M. Bijelic, T. Gruber, and W. Ritter, "Benchmarking image sensors under adverse weather conditions for autonomous driving," in *2018 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2018, pp. 1773–1779.

[101] T. Gruber, F. Julca-Aguilar, M. Bijelic, and F. Heide, "Gated2depth: Real-time dense lidar from gated images," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 1506–1516.

[102] C. Buciluǎ, R. Caruana, and A. Niculescu-Mizil, "Model compression," in *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2006, pp. 535–541.

[103] J. Ba and R. Caruana, "Do deep nets really need to be deep?" *Advances in neural information processing systems*, vol. 27, 2014.

[104] G. Hinton, O. Vinyals, J. Dean *et al.*, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, vol. 2, no. 7, 2015.

[105] G. Urban, K. J. Geras, S. E. Kahou, O. Aslan, S. Wang, R. Caruana, A. Mohamed, M. Philipose, and M. Richardson, "Do deep convolutional nets really need to be deep and convolutional?" in *ICLR workshop*, 2016.

[106] J. Kim, S. Park, and N. Kwak, "Paraphrasing complex network: Network compression via factor transfer," *Advances in neural information processing systems*, vol. 31, 2018.

[107] S. I. Mirzadeh, M. Farajtabar, A. Li, N. Levine, A. Matsukawa, and H. Ghasemzadeh, "Improved knowledge distillation via teacher assistant," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 34, no. 04, 2020, pp. 5191–5198.

[108] S. Ahn, S. X. Hu, A. Damianou, N. D. Lawrence, and Z. Dai, "Variational information distillation for knowledge transfer," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 9163–9171.

[109] B. Heo, M. Lee, S. Yun, and J. Y. Choi, "Knowledge transfer via distillation of activation boundaries formed by hidden neurons," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, 2019, pp. 3779–3787.

[110] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European conference on computer vision*. Springer, 2014, pp. 740–755.

[111] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," 2017.