Domain Adaptive Object Detection using Semi-Supervised Learning and Pseudo labeling based Self-Training

Tilahun Gedefaw Belay

A Thesis Submitted to the Department of Computer Science and Engineering School of Electrical Engineering and Computing

Presented in Partial Fulfillment of the Requirement for the Degree of Master's in Computer Science and Engineering

Office of Graduate Studies

Adama Science and Technology University

June,2023

Adama, Ethiopia

Domain Adaptive Object Detection using Semi-Supervised Learning and Pseudo labeling based Self-Training

Tilahun Gedefaw Belay

Advisor: Worku Jifara Sori (Ph.D)

A Thesis Submitted to the Department of Computer Science and Engineering School of Electrical Engineering and Computing

Presented in Partial Fulfillment of the Requirement for the Degree of Master's in Computer Science and Engineering

Office of Graduate Studies

Adama Science and Technology University

June, 2023

Adama, Ethiopia

DECLARATION

I hereby declare that this Master Thesis entitled "**Domain Adaptive Object Detection using Semi-Supervised Learning and Pseudo labeling based Self-Training"** is my original work. That is, it has not been submitted for the award of any academic degree, diploma, or certificate in any other university. All sources of materials that are used for this thesis have been duly acknowledged through citation

Tilahun Gedefaw Belay

_____	_____	_____
Name	Signature	Date

RECOMMENDATION

I, the advisor of this thesis, hereby certify that I have read the revised version of the thesis entitled "**Domain Adaptive Object Detection using Semi-Supervised Learning and Pseudo labeling based Self-Training**" prepared under my guidance by Tilahun Gedefaw Belay submitted in partial fulfillment of the requirements for the degree of Mater's of Science computer science and engineering. Therefore, I recommend the submission of a revised version of the thesis to the department following the applicable procedures.

Worku Jifara Sori (Ph.D)

Major Advisor Signature Date

_____ _____ _____

Co-Advisor Signature Date

APPROVAL PAGE

I/we, the advisors of the thesis entitled "**Domain Adaptive Object Detection using Semi-Supervised Learning and Pseudo labeling based Self-Training**" and developed by Tilahun Gedefaw Belay, hereby certify that the recommendation and suggestions made by the board of examiners are appropriately incorporated into the final version of the thesis.

Major Supervisor	Signature	Date
Co-Supervisor	Signature	Date

We, the undersigned, members of the Board of Examiners of the thesis by ---------- (Name of student) have read and evaluated the thesis entitled "---------- ---------- ---------- ----------" and examined the candidate during the open defense. This is, therefore, to certify that the thesis is accepted for partial fulfillment of the requirement of the degree of Master of Science in ----------.

Chair Person	Signature	Date
Internal Examiner	Signature	Date
External Examiner	Signature	Date

Finally, approval and acceptance of the thesis is contingent upon submission of its final copy to the Office of Postgraduate Studies (OPGS) through the Department Graduate Council (DGC) and School Graduate Committee (SGC).

Department Head	Signature	Date
School Dean	Signature	Date
Office of Postgraduate Studies, Dean	Signature	Date

ACKNOWLEDGMENT

I am immensely grateful to Almighty God for His boundless blessings, unwavering guidance, and continuous support throughout my academic journey. His divine presence has provided me with the strength, wisdom, and inspiration needed to undertake this thesis and overcome every obstacle along the way.

I would like to express my deepest gratitude to my thesis advisor **Dr.-Worku Jifara**, for his invaluable guidance, support, and suggestion throughout the thesis work from starting up to the end. I would also like to extend my heartfelt appreciation to Prof. Yun Koo Chung head of the Computer Vision and Robotics SIG.

I would like to acknowledge my dear family and friends, whose unwavering love, support, and encouragement have been my pillars of strength throughout this academic journey. Their belief in me and their constant words of encouragement have propelled me forward even during the most challenging times. I am truly blessed to have them by my side.

Lastly, I would like to express my gratitude to all the individuals who have contributed to this research in various ways, whether through their collaboration, insightful discussions, or technical assistance. Your contributions have been invaluable, and I am deeply grateful for your support.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBRIVATIONS AND ACRONYMS

AI	Artificial intelligence
ML	Machine learning
CV	Computer vision
DL	Deep Learning
DBN	Deep Belief Network
SIFT	Scale invariant feature transform
HOG	Histogram of Oriented gradient
SURF	Speeded up robust features
GAN	Generative Adversarial Networks
CNN	Convolutional neural networks
FCN	Fully convolutional network
DCNN	Deep Convolutional Neural networks
RCNN	Region based Convolutional neural networks
YOLO	You only look once
SSD	Single-shot detection
FPN	Feature pyramid networks
RPN	Regional proposal network
ROI	Region of Interest
VGG	Visual geometry Group
ReLU	Rectified Linear unit
TL	Transfer Learning
DA	Domain Adaptation
UDA	Unsupervised Domain adaptation
DAOD	Domain adaptive Object detection
GRL	Gradient reversal Layer
EMA	Exponentially moving Average
UMT	Unbiased Mean Teacher

MT	Mean Teacher
ST	Self-Training
TDD	Target-perceived Dual-branch Distillation
AT	Adaptive Teacher
DIR	Domain Invariant Representation
NIR	Near-infrared Imaging
GPS	Global Positioning System
BDD	Berkley Deep Drive
KITTI	Karlsruhe institute of technology and Toyota technological Institute
UML	Unified Modelling Language
IDE	Integrated Development Environment
ResNet	Residual Network
COCO	Common Objects in Context
VOC	Visual Object classes
mAP	Mean absolute precision
IOU	Intersection over Union
NMS	Non-maximal Suppression
GPU	Graphical Processing Unit
RAM	Random Access Memory
BCEL	Byte code Engineering Library
GTX	Giga Texel Shader extreme
GTA	Grand auto Theft
CDATOD	Cross-Domain Adaptive Teacher for Object Detection
LDAOPT	Learning Domain Adaptive Object Detection with Probabilistic Teacher

LIST OF NOTATIONS AND SYMBOLS

$\mathcal{D}_S$	Source domain
$\mathcal{T}_S$	Target Task
$\mathcal{D}_T$	Target Domain
$\mathcal{T}_T$	Target Task
θ_t'	Teacher model
α	Smoothing coefficient hyperparameter
x_s	Labelled source data
x_u	Unlabeled target data
$\hat{y}_u$	Pseudo-label vectors
p_i	Prediction probability of classification
$\mathcal{L}_{cls}$	Classification loss
$\mathcal{L}_{\text{reg}}$	Box regression loss
y_i^u	Pseudo label annotated by the teacher model
L_U	Unsupervised loss
$\mathcal{L}^{rpn}$	RPN loss
$\mathcal{L}^{roi}$	ROI loss
$p_i^{(u,v)}$	Domain classifier

ABSTRACT

In the area of computer vision object detection is concerned with detecting and localizing objects within images. Accurate localization of objects in an image plays a vital role in various computer vision tasks, such as object detection, tracking, and segmentation. However, when it comes to training machine learning models for object detection, a significant amount of annotated data is typically required for supervised learning. A common challenge arises when these models, trained on one data domain, encounter a drop-in performance when tested on data from an unseen domain. The existence of domain discrepancies or shifts poses significant hurdles to the generalization ability of the deep learning models. Even though augmenting the training data with samples from the new domain can improve performance, the process of collecting annotations for this data is often time-consuming and labor-intensive. To address this issue, researchers have created deep transfer learning approaches, particularly deep domain adaptation techniques. These approaches aim to adapt the model's learned representations to effectively handle domain discrepancies, enabling the network to generalize well across different domains. This study presents a novel Deep Domain adaptation approach using semi-supervised learning approach with pseudo labeling based self-training to solve object detection problem. It presents an approach to improve domain adaptive object detection through the integration of several techniques such as progressive confidence thresholding, non-maximal suppression, consistency regularization, and image-level and instance-level alignment are applied to enhance the performance of object detection models when dealing with domain shifts. The proposed method increases the detection accuracy and robustness by effectively addressing the challenges posed by domain adaptation. The experimental result indicates a significant improvement in performance of the initial Faster R-CNN model, which was exclusively trained on the source domain, when evaluated across diverse domains. As a result, the proposed approach improves the overall performance of deep neural networks by enhancing their ability to generalize across domains.

Keywords: Object detection, Domain adaptation, semi-supervised learning, pseudo-labeling Source domain, Target domain, self-training

CHAPTER ONE

INTRODUCTION

1.1. Background of the Study

The emergence of deep learning has led to remarkable advancements in a wide range of fields, including computer vision, robotics, and natural language processing. In the field of computer vision, significant progress has been made, particularly with the development of convolutional neural networks (CNNs) (LeCun et al., 2015). CNN models have demonstrated exceptional performance in various tasks such as object detection, semantic segmentation, object classification, thanks to their exceptional learning capabilities. Consequently, CNNs have surpassed traditional computer vision algorithms and are now extensively employed in real-world applications (LeCun et al., 2015) (M. Wang & Deng, 2018).

The development of object detection techniques such as You Only Look Once (YOLO), Faster RCNN, and Single Shot Multi-Box Detector (SSD) has greatly improved the speed and performance of object detection (Li et al., 2020).While most deep convolutional neural network models rely on supervised training, which requires large datasets with thousands of labeled images, they often struggle to generalize well when presented with visually different images compared to the training set (L. Liu et al., 2020). Domain adaptation is a field within machine learning that focuses on addressing the problem of domain shift. It is a crucial area of study that holds potential benefits for both business and academia. (S. Zhao et al., 2022). It has received a lot of interest and has undergone substantial research because of its usefulness in reducing the expense of time-consuming human annotation.(M. Wang & Deng, 2018).

The performance of object detection models can suffer when applied to images from different domains than the one they were trained on. Such models often experience a significant drop in performance when tested on various datasets (Xie et al., 2019) (W. Li et al., 2020). Traditionally, addressing this issue necessitates human annotators manually labeling the position and size of each object in the image, which is a tedious and time-consuming task (Hsu et al., 2020). To alleviate the requirement for manual annotation in every new target domain, domain adaptation (DA) methods have been developed. (Y. Chen, Wang, et al., 2021; Rodriguez & Mikolajczyk, 2020). In

this study, I propose a technique for multi-target-domain adaptation in domain invariant object detection. The approach combines semi-supervised learning and self-training, building upon the widely adopted Faster RCNN model as a baseline. By leveraging these methods, the goal is to enhance the adaptability of the deep learning model across multiple target domains, without the need for extensive manual annotation efforts.

1.2. Motivation for the Study

Although standard machine learning technology has proven to be highly effective and successfully employed in numerous practical applications, it does have notable limitations when it comes to certain real-world scenarios. Optimal performance of machine learning is achieved when there is an abundance of labeled training examples that closely align with the distributions of test data (L. Liu et al., 2020). Nevertheless, gathering enough training data is sometimes a costly, time-consuming, or even unachievable endeavor in many cases. Recent research demonstrates that dataset bias or domain shift in context of object classification or detection is a difficult problem that requires attention (Xie et al., 2019) (W. Li et al., 2020). The majority of object detection models are now evaluated and trained using similar image distribution, but in real-world applications, visual domains are continually changing. Visual domains might often contrast in a variety of (sometimes hidden) determinants, such as scene, resolution, intra-category variation, view angle, motion blur, object placement and pose, scene lighting, camera features, backdrop clutter, and so on. Studies have shown that the performance of cutting-edge image classifiers has drastically declined due to data bias caused by a switch from commercial to consumer video, pose shifts], and, more importantly, the training of datasets that are biased in the manner in which they were gathered.

The inability of deep learning models to generalize to visually diverse images is considered a form of model bias because the model is biased towards the specific patterns and features present in the training data (Z. He et al., 2022). This bias can lead to poor performance and inaccuracies when the model encounters new or visually diverse images. To mitigate this bias and improve the generalization capabilities of deep learning models, researchers and practitioners employ various techniques such as data augmentation, transfer learning, domain adaptation, or using more diverse training datasets that better represent the target data.

.

One potential approach in machine learning to address this challenge is transfer learning. This concept draws inspiration from human learning, where individuals can transfer information across different domains. For example, someone who has learned to play the violin may be able to learn the piano more quickly due to shared knowledge between musical instruments. Transfer learning in machine learning is designed to utilize the knowledge gained from the source domain to improve learning in the target domain. This approach has the potential to decrease the reliance on labeled instances. (W. Li et al., 2020; Rodriguez & Mikolajczyk, 2020; M. Wang & Deng, 2018) Nevertheless, it is crucial to recognize that not all knowledge from the source domain can be advantageous for tackling new tasks in the new target domain.

Domain adaptation is a form of transfer learning that assumes the source and target domains share the same feature space but have distinct distributions. Many companies have spent significant resources collecting and annotating data, but there will never be enough data for every task or domain. This means that learning to transfer knowledge between domains is critical, as it can reduce the cost of acquiring labeled training data from previously unknown domains (Rodriguez & Mikolajczyk, 2020). For example, a huge online retail corporation may have images of millions of products and want to develop a product classification model, but only certain domains of product images have labels.

1.3. Statement of the Problem

The influence of domain shift on object detection has been largely ignored, even though there is significant progress in the development of various techniques in object detection particularly in the context outdoor environments. Under adverse conditions such as rain, fog, haze, and variable illumination conditions, the accuracy and precision of object detection algorithms might decrease dramatically (Yu et al. 2020). In bad weather, like when it's raining, snowing, or foggy, the quality of the visual signals that self-driving cars get from other cars might be messed up and changed (Wulfmeier et al., 2018). As a result of these circumstances, scene contrast and visibility are reduced, and the ability of object detection algorithms to identify crucial items in the area will not be accurate. Consequently, one of the most difficult issues for object detection algorithms in outdoor environment will be to mitigate the effects of weather and light on visual data (Sindagi et

al., 2020). From shallow models to deep models, a lot of effort has been made in the past to construct robust classifiers that can learn how to do a visual recognition task from data.

Although enormous amount of unlabeled dataset is produced in different fields, labeling these dataset remains significantly costive. Alternative approaches have been put forward in the literature to reduce the load of annotating by using unlabeled data, data, or models from related fields (referred to as transfer learning)(Zheng et al., 2020). A specific use of transfer learning called domain adaptation (DA) uses labeled dataset from source domains to train a model for unlabeled data for the target domain. The work is often believed to be the same, hence class labels are used across domains (Z. He & Zhang, 2020).

The objective of domain adaptive object detection (DAOD), as outlined by Sindagi et al. in 2020, is to address the disparity between training and test domains, specifically in scenarios involving different weather conditions. The primary focus of DAOD is to bridge the domain gap and enable effective object detection across diverse environmental settings. Various methods have been investigated to solve DAOD in order to close the gap by using feature alignment from source and target domain, obtaining domain-invariant representations (DIR), which serve as a bridge to lessen the impact of domain-shift and can (X. Yang et al., 2020)aid in the extraction of domain-invariant object features, is crucial for DAOD.

In domain adaptation, the goal is to learn a predictive function Ft that utilizes knowledge gained from a source domain Ds and task Ts to improve performance on a target domain Dt and task Tt (X. Liu et al., 2022). Essentially, Ft adapts to the domain divergence between Ds and Dt. However, it is important to exercise caution when conducting transfer learning in scenarios where $Ds \neq Dt$ and $Ts \neq Tt$ (X. Liu et al., 2022).

In real-world scenarios, distribution differences between domains, also known as domain shifts, are more common in visual applications. The severity of domain mismatch increases when the source and target domains contain diverse types of images, such as regular images, NIR images, paintings, or drawings. These differences can arise from various factors like changes in backdrop, location, or variations in posture (X. Liu et al., 2022; Nguyen et al., 2020; M. Wang & Deng, 2018). Service-oriented companies, in particular, face concerns as the data distribution may significantly differ among customers, even for the same service or job. To address this, effective

adaptation mechanisms are required for machine learning components in service solutions when transitioning from one client or location to another, accommodating the changing circumstances (Zhang et al., 2022; S. Zhao et al., 2022). In the fields of surveillance and urban traffic comprehension, adjusting models trained on previous sites for optimal performance in new contexts can be achieved through data labeling or fine-tuning. However, data labeling is costly and time-consuming. Alternatively, models can be constructed by combining labeled source data with new unlabeled target data or by adapting pre-trained models using available unlabeled (and labeled) target datasets. This approach leverages existing knowledge while utilizing unlabeled data to improve model performance in the new domain. In many practical scenarios, obtaining a vast amount of annotated training data can be challenging. Fine-tuning becomes necessary in such cases (Y. Yu et al., 2019; S. Zhao et al., 2022; Zheng et al., 2020).

When there is a significant shift in lighting conditions, position, or image quality between two domains, the performance of models can be severely impacted.

In the proposed framework, a semi-supervised paradigm with self-training (Tarvainen & Valpola, 2017) is adopted. Semi-supervised learning can enhance classification accuracy and speed up the learning process of trained networks.

Specifically, the initial Faster R-CNN model is used to get pseudo annotations for unlabeled images, and high-quality pseudo labels are filtered out. These pseudo labels are then utilized by the second model during the semi-supervised training phase, contributing to the overall improvement in performance.

1.4. Research Questions

Research Question 1: What is the effect of using pseudo-labeling based self-training regarding making object detectors domain adaptive?

Research Question 2: What is the effective mechanism to improve the quality of pseudo labels in semi supervised learning for domain adaptation?

Research Question 3: How does the integration of consistency regularization, image-level and instance-level alignment techniques contribute to improved domain adaptation in object detection tasks using semi-supervised learning and pseudo labeling?

1.5. Objective of the Study

1.5.1 General Objective

The main goal of the study is to develop Domain Adaptive Faster RCNN framework using semi-supervised learning with self-training for improving performance of cross domain object detection.

1.5.2 Specific Objective

In order to accomplish the general objective of this thesis, the following specific objectives will be accomplished.

- To introduce iterative pseudo labeling approach in the semi-supervised model by using progressive confidence thresholding
- To introduce adaptive feature alignment technique on to reduce the distributional discrepancy between the two faster RCNN models.
- To apply class-wise non-maximum suppression (NMS) after pseudo labeling to eliminate the redundant box predictions.
- To propose a consistency regularization on aligning the target domain to the source domain and optimizing them using faster RCNN model
- Comparative analysis is performed using benchmark datasets that include images captured in both sunny and adverse weather conditions. These datasets should cover a wide range of outdoor scenarios and encompass various challenges, including variations in lighting, weather phenomena, and other environmental factors.

1.6. Scope and Limitations

1.6.1. Scope of the Study

- Adapt the faster RCNN object detection framework to different domains
- Implement a Faster RCNN model with image-level and instance-level feature alignment

- Implement Faster-RCNN framework that generates Pseudo labels with pseudo label selection mechanism
- Apply iterative confidence thresholding and non-maximal Suppression to produce high quality pseudo labels on the first Faster-RCNN model
- Extensively experiment on domain shifts associated with cross dataset and adverse scenarios, and demonstrate results on several datasets such as cityscapes, foggy cityscapes, BDD10k, KITTI and Sim10k.

1.6.2. Limitation of the study

- **Source-free domain adaptation**: Describes a scenario in which samples from the source domain cannot be accessed during the training process of domain adaptation. This can occur due to various reasons such as privacy concerns, legal constraints, or slow data transfer.
- **Computationally expensive adaptation techniques**: Some domain adaptive object detection methods involve computationally expensive operations, such as adversarial training, multiple stages of adaptation, or fine-tuning on target domain data. These techniques can be time-consuming and computationally intensive, limiting their practical applicability in real-time or resource constrained environments
- The proposed approach is evaluated only on major kinds of domain shift scenarios not on all kinds of domain shift scenarios.

1.7. Beneficiaries of the Study

The proposed thesis work has the potential to bring several benefits to various areas listed below.

- **Autonomous systems**: One significant advantage is addressing the limitations of object detection techniques used in autonomous vehicles. However, in reality, there can be a domain gap, causing an object detection model developed in normal weather conditions to underperform in different weather conditions.
- **Surveillance and security cameras:** disturbance of the images also degrades the perception quality of surveillance cameras which makes difficult to monitor the surrounding environment clearly.

- **Robotics:** Robots are increasingly seeing, analyzing, and making judgments like humans, due to breakthroughs in emerging technologies .. Image degradation caused by rain fog, haze and change in illumination condition of the surrounding environment it is difficult for robots to achieve high precision in object detection.
- **Drone Object Detection:** When a drone navigates a building in search of things, the drone wants to be able to see as much of its surroundings as possible. As drones are developed to navigate to different places, they also face challenges due to variations of domains when moving from one location to another locations.
- **Traffic Flow Monitoring and Analysis:** is One of the essential components of the smart traffic concept is vehicle recognition and monitoring.

1.8. Organization of the Thesis

The thesis is organized into the following sections:

Chapter Two: This chapter delves into the theoretical aspects pertaining to domain adaptation and object detection under challenging conditions. It conducts an extensive review of relevant literature, discussing the theoretical framework of deep learning, semi-supervised learning, and self-training. The goal is to establish a comprehensive understanding of the concepts and techniques that are essential for the proposed research.

Chapter Three: This chapter presents a detailed account of the research methodology employed in the study. It outlines the various methods and techniques utilized in developing the proposed solution and selecting the most appropriate approach.

Chapter Four: This chapter provides a comprehensive description of the architecture of the proposed model. It offers detailed explanations of the model's structure, pseudocode for implementation, and mathematical explanations of the associated learning functions. The emphasis is on presenting a clear and thorough understanding of the components of the proposed model.

Chapter Five: In this chapter, the code implementation of the proposed approach is presented, along with a detailed discussion on the setup of the working environment. Specifically, it focuses on the domain adaptive Faster R-CNN object detection algorithm and provides explanations of the training process using code snippets.

Chapter Six: This chapter presents the findings of the proposed solution, compares them with previous approaches, and provides an interpretation of the results obtained from the

experimentation. It highlights the achievements and implications of the study, demonstrating how the proposed solution outperforms existing methods.

Chapter Seven: This chapter summarizes the research and provides suggestions for potential future work in the field. It summarizes the key findings and implications of the study, offering insights into further avenues of exploration and development. It serves as a culmination of the research, tying together the main contributions and proposing directions for future research endeavors.

CHAPTER TWO

LITERATURE REVIEW AND RELATED WORKS

2.1. Introduction

Since the invention of the computer, many people have questioned that whether a machine might develop intelligence, whatever subjective standard of intelligence there may be. Nowadays Artificial intelligence is a growing area. It focuses on the study of intelligent agents, or systems that behave intelligently in order to make the best decisions possible given the conditions of their surroundings and a certain objective (Callier & Sandel, 2021). Different techniques to artificial intelligence have been used during the last few years. One such strategy is to hardcode general information in formal languages in order to create a knowledge base (Callier & Sandel, 2021; LeCun et al., 2015). However, human supervisors find it difficult to create formal rules that are intricate enough to capture reality, which has a significant influence on how well these systems can draw conclusions.

Researchers have developed learning algorithms known as machine learning that have the capacity to learn on its own by extracting patterns from unstructured data. It can also be described as a computer program that gains knowledge from experience E related to a specific task T and a measurement of performance P. In simpler terms, if the program's performance in task T gets better as it gains more experience E, as judged by measurement P.(Callier & Sandel, 2021; LeCun et al., 2015; L. Liu et al., 2020). The aim of machine learning is to solve a problem based on prior knowledge (data) without having to be manually programmed to do it. We can use software to find patterns and schemas from the data; this method is useful for solving issues that are too complex to be modelled manually (LeCun et al., 2015; M. Wang & Deng, 2018). Examples of such issues include object detection, autonomous driving, natural language processing, path finding, classification, regression, and clustering on data.

Deep learning is a subset of machine learning techniques and algorithms that are inspired by the functioning of the human brain. Deep learning emerged as a novel concept by LeCun et al. (2015) in their 2006 paper on Deep Belief Networks (DBNs). DBNs consist of learning modules that are subsets of Boltzmann machines (Callier & Sandel, 2021; W. Li et al., 2020). Deep Belief Networks

(DBNs) consist of a visible layer that represents the input data and multiple hidden layers, which are trained to capture complex patterns in the data. Artificial neural networks aim to automatically extract relevant features from data without explicitly specifying them, which sets them apart from traditional machine learning approaches where specific features of interest are defined in the data.

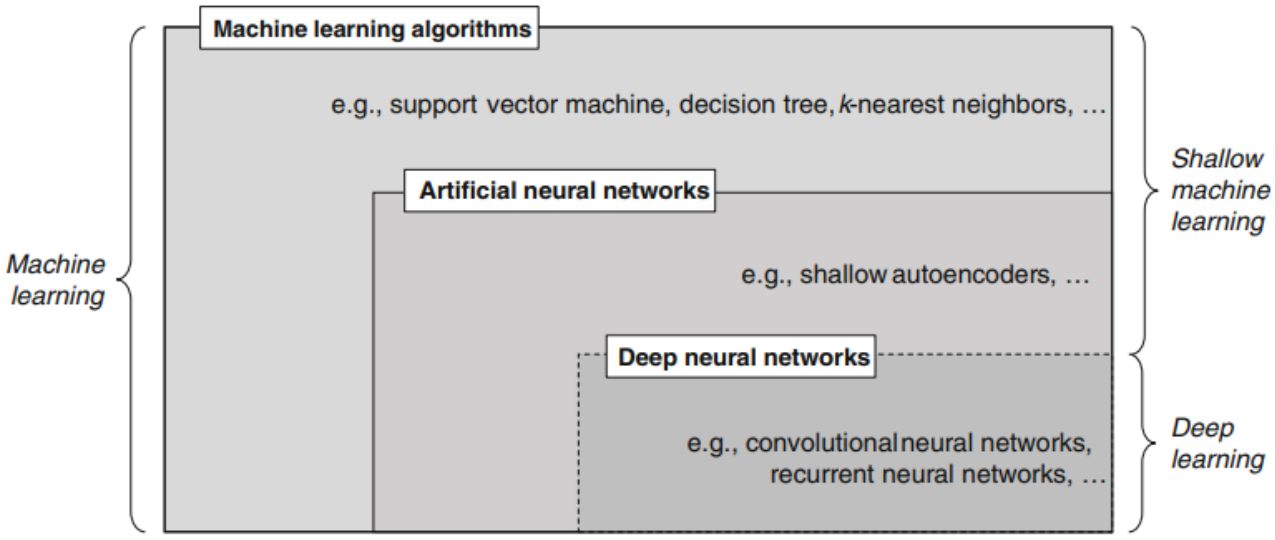


Figure 2.1: Traditional vs Deep learning approaches (MICHAEL COPELAND, 2016)

2.2. Convolutional Neural Networks

CNNs, known as convolutional neural networks, belong to the category of feedforward neural networks that are specifically used for processing data arranged in a spatial grid, such as 2D or 3D images. These networks draw inspiration from the visual perception system observed in living organisms, as mentioned in a study by W. Li et al. in 2020. Like any other artificial neural network (ANN), CNNs are comprised of neurons with adjustable weights and biases. Each neuron carries out a dot product operation on its inputs and can subsequently apply an activation function to introduce non-linear behavior into the network.(C. D. Xu et al., 2020).

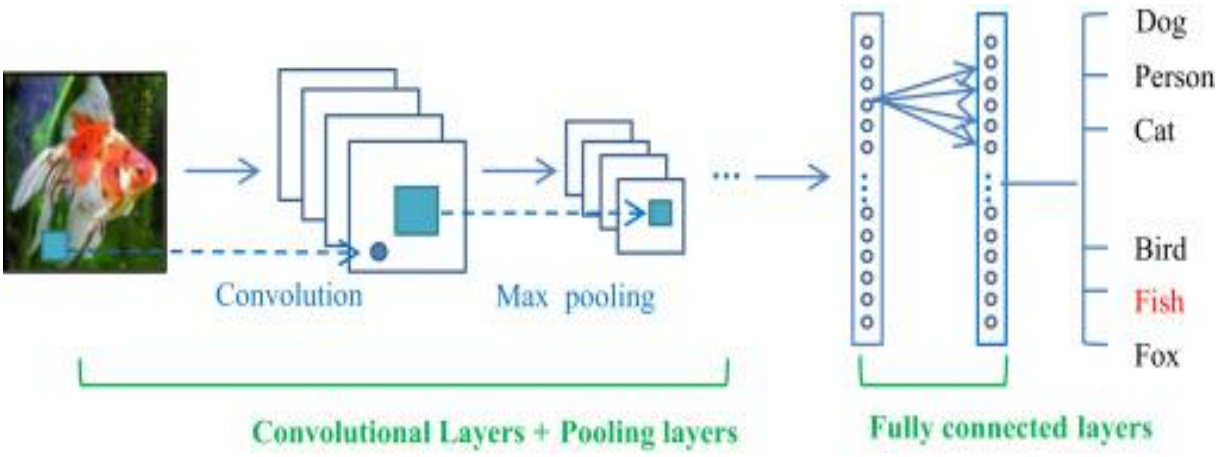


Figure 2.2: The general CNN architecture (Hassan et al., 2019)

The main objective of CNN network is to acquire a non-linear combination of high-level features extracted from the output of the convolutional layer, which is referred to as a feature map. This is achieved by flattening the feature map and passing it through a fully connected layer, which is responsible for the classification task.(Khodabandeh et al., 2019).

(1) A convolutional layer is composed of multiple trainable parameters, a Convolutional Neural Network (CNN) is constructed using a set of convolutional filters. Utilizing convolutional filters, a CNN accepts a four-dimensional tensor as input to execute a sliding convolutional operation to extract the associated feature map.

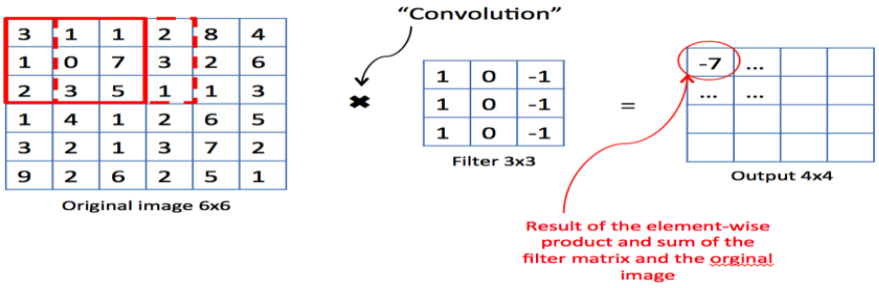


Figure 2.3: A Convolutional Layer (Ravindra Parmar, 2022.)

(2) A fully-connected (linear) also known as a linear layer, is utilized to produce the probabilities associated with each classification label by combining the extracted features. This layer consists of numerous trainable linear operation neurons. The CNN takes a flattened

input vector and applies the weights of all neurons to perform linear computations on each element.

(3) Activation layers, also referred to as activation functions, play a critical role in artificial neural networks (ANNs) by introducing nonlinearity to the network's computations. They are typically applied element-wise to the neurons within hidden layers.

The sigmoid function is one widely utilized activation function which maps input to a range of 0 to 1. It exhibits linearity near the origin but "squashes" large positive or negative inputs towards 1 or 0, respectively. Another frequently employed activation function is the hyperbolic tangent (tanh), The rectified linear unit (ReLU) is a highly popular activation function in deep neural networks. It is defined as the maximum between zero and the input value (x = max (x, 0)), resulting in a rectifying effect. ReLU effectively addresses the vanishing gradient problem and allows for faster convergence during training. Lastly, the softmax function is commonly used as an activation function for the output layer of a neural network. It performs a normalization operation, assigning probabilities to different classes and enabling the network to provide predictions in the form of class probabilities. Each activation function offers distinct properties that can impact the performance and learning dynamics of a neural network. The choice of activation function depends on the specific requirements of the task at hand and the characteristics of the data being processed.

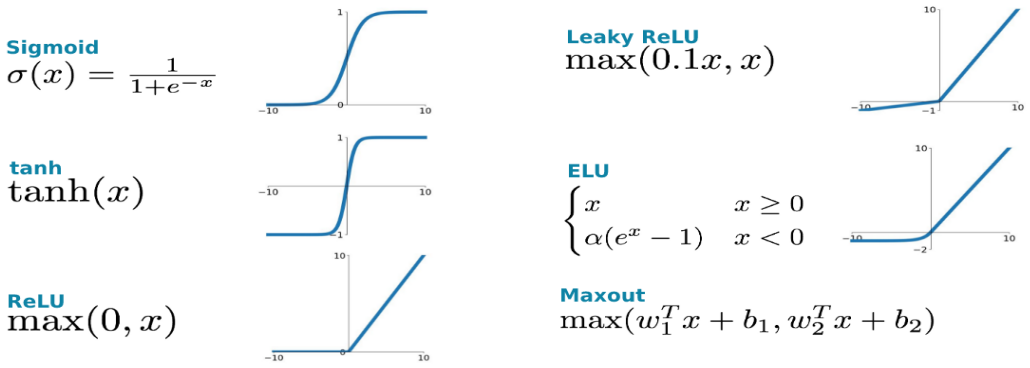


Figure 2.4: Activation Layers (Ravindra Parmar, 2022.)

(4) A normalization layer: The concept of normalization is often implemented using a normalization layer, which operates at the instance level, within a mini-batch, or for a

specific group of inputs. This layer typically includes adjustable parameters and can modify the normalization of the input data.

(5) A pooling layer: To improve the performance of a neural network, a pooling layer is utilized. It serves as a loss layer, responsible for evaluating and penalizing the deviation between the network's predictions, which are tailored to the given task, and the actual labels of the data. This loss layer plays a crucial role in guiding the training process by providing feedback to the network, motivating it to make more precise predictions and reduce the difference between its output and the true labels.

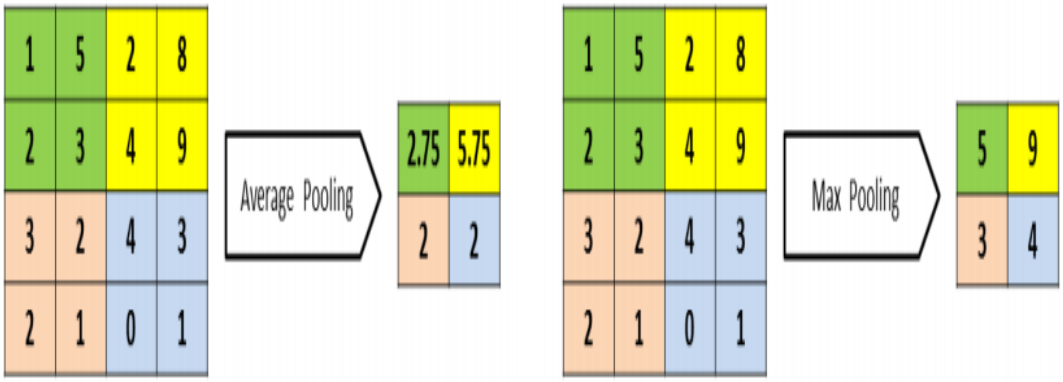


Figure 2.5: The pooling Layer (Ravindra Parmar, 2022.)

(6) To improve the neural network, **A loss layer** calculates the discrepancy between the outputs of the task-specific network and the true labels, imposing a penalty on the difference.

2.3. Object Detection

Object detection is widely regarded as one of challenging tasks in computer vision. It encompasses a diverse range of related tasks within the field, including semantic and instance segmentation, object tracking, image captioning, and more. These tasks all revolve around the identification, localization, and understanding of objects within images or videos, contributing to the broader objective of comprehending visual information in computer vision applications (L. Liu et al., 2020) . The overall process of object detection includes the localization and categorization of object occurrences.

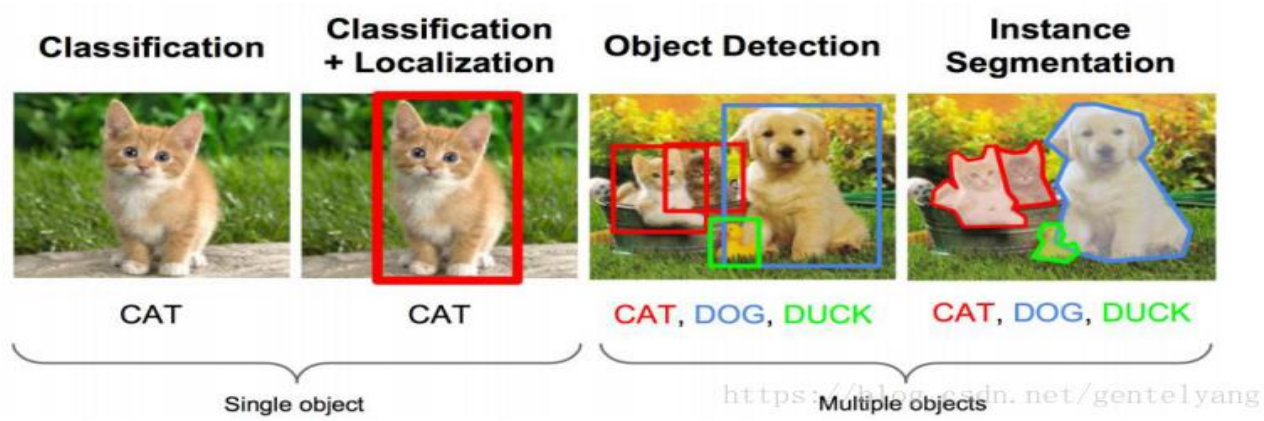


Figure 2.6: Object detection classification and Instance Segmentation (Mohan Dogra, 2020.)

Object detection involves two main steps: feature extraction and the subsequent stages of detection and classification. (Chaudhari et al., 2020). Deep learning-based object detection methods can be classified into two main groups: two-stage object detectors and one-stage object detectors, as discussed in a study by Zhu et al. in 2019. Two-stage object detection algorithms (s R-CNN, Fast R-CNN, and Faster R-CNN) decouple the localization of objects from the classification task, as explained by Ren et al. in 2017. These models first propose regions of interest and then classify those regions. On the other hand, one-stage object detection algorithms take a different approach by generating candidate regions and directly classifying them as objects or non-objects. For example, instead of predicting regions and subsequently categorizing them, one- stage detectors like YOLO and SSD utilize feature pyramid networks (FPNs) as a foundational framework to detect objects of various sizes in a single pass (Chaudhari et al., 2020; W. Liu et al., 2016). This allows for efficient and real-time object detection across different scales.

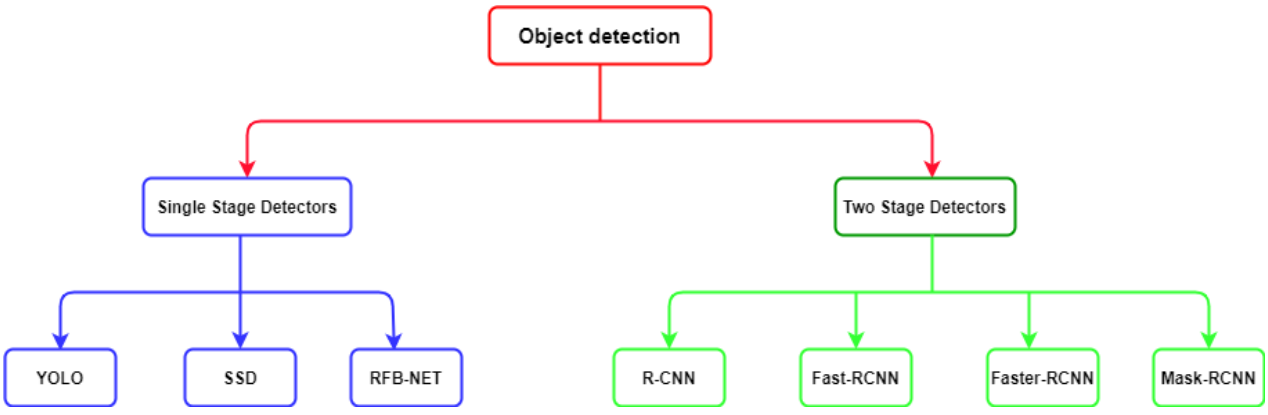


Figure 2.7: Types of Object detection algorithms

2.3.1. Single Stage Detectors

One-stage detectors, in contrast to their two-stage counterparts, directly predict potential object regions without relying on region proposal networks. These detectors are recognized for their efficiency and are widely employed in real-time object detection systems. Researchers like W. Liu et al. (2016) have proposed various methods that utilize one-stage detectors for object detection in challenging scenarios. In this section, I will organize these approaches based on the specific one-stage object detectors they utilize.

2.3.2. Two-Stage Detectors

Two-stage detectors, as highlighted by Girshick in 2015, are commonly used in object detection to address various challenges. They follow a two-stage approach, consisting of region proposal and object classification/refinement stages. Here are three well- known two-stage detector algorithms:

1. R-CNN (Regions with Convolutional Neural Networks): The R-CNN model, which was introduced by Girshick et al. in 2014, revolutionized object detection by introducing the idea of utilizing region proposals to concentrate on potential regions containing objects of interest. This approach improved the efficiency and accuracy of object detection by selectively examining specific regions rather than processing the entire image. It applies CNNs to extract features from these proposed regions and uses SVM classifiers to classify and localize objects.

2. Fast R-CNN (Fast Region-based Convolutional Networks): In 2015, Girshick presented the Fast R-CNN model, which improves upon the R-CNN approach by introducing an end-to-end trainable system. The R-CNN model, introduced by Girshick et al. in 2014, eliminates the requirement for distinct fine-tuning stages and substitutes the SVM classifier with fully connected layers to handle object classification and bounding box regression.
3. Faster R-CNN, introduced by Girshick in 2015, builds upon the advancements of two-stage detectors by introducing a Region Proposal Network (RPN). This network significantly improves the efficiency of object detection by generating region proposals directly from feature maps. As a result, the need for external proposal

methods is eliminated, leading to streamlined and more efficient object detection pipelines. The proposed regions are then classified and refined by the subsequent stages to detect objects accurately.

These two-stage detectors have proven effective in object detection tasks, providing accurate results by combining region proposal and classification/refinement stages.

2.3.2.1. R-CNN

From object detection models, RCNN (Regions with CNN features) was one of the initial convolutional neural networks (CNN)-based models proposed for this task. It utilized a module that eliminated object proposals before feeding them to a CNN. By applying a CNN, important features could be extracted, enabling classification and localization of the proposed regions. The selective search method was initially employed to generate approximately 2000 regions. AlexNet, a CNN architecture, was used as the backbone to extract feature vectors with 4096 dimensions. The features were then fed into binary classifiers tied to specific classes.

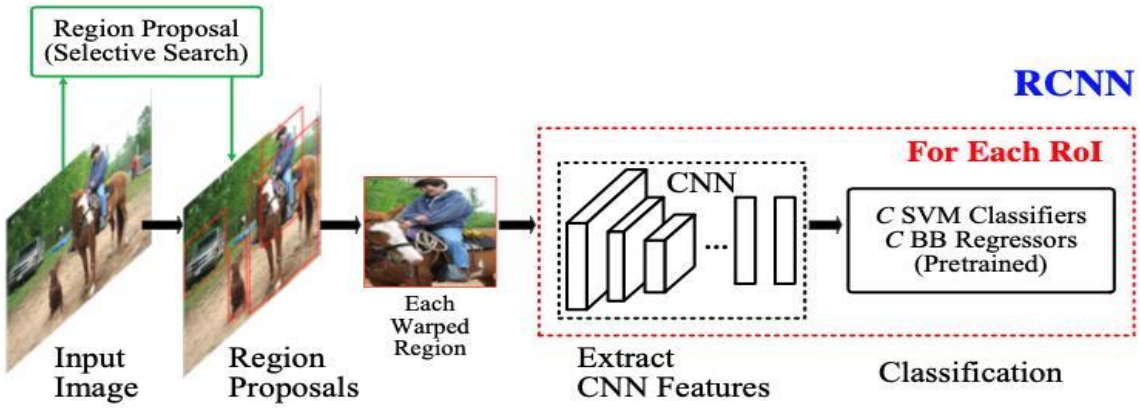


Figure 2.8: R-CNN overview (Girshick et al., 2014)

2.3.2.2. Fast-RCNN

The Fast R-CNN model, proposed by Girshick in 2015, builds upon the R-CNN framework introduced by the same authors in 2014. The Fast R-CNN model introduces various enhancements to improve the efficiency and effectiveness of object detection. Notably, it adopts an end-to-end trainable system, removing the requirement for separate fine-tuning stages. Additionally, instead of utilizing distinct regressors for bounding box coordinates,

the model incorporates a second FC layer that directly outputs the bounding box coordinates for the foreground classes. These enhancements contribute to faster and more accurate object detection in the Fast R-CNN model.

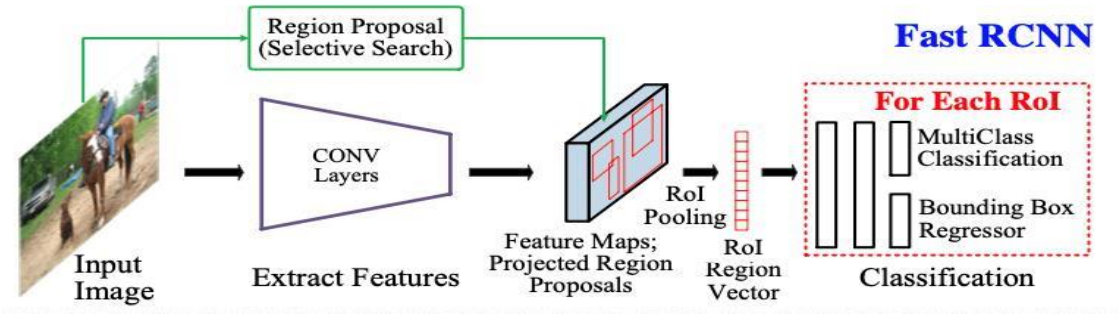


Figure 2.9: Fast-RCNN overview (Girshick et al., 2015)

2.3.2.3. Faster-RCNN

Faster R-CNN, introduced by Ren et al. in 2017, is a widely used framework in computer vision for detecting objects. It enhances previous techniques by combining a region proposal network with a convolutional neural network to achieve efficient and precise object detection. The main concept behind Faster R-CNN involves replacing the conventional selective search algorithm employed in traditional R-CNN with the RPN. This RPN generates region proposals directly from the convolutional feature maps of the input image, eliminating the need for external methods for region proposal and greatly accelerating the detection process.

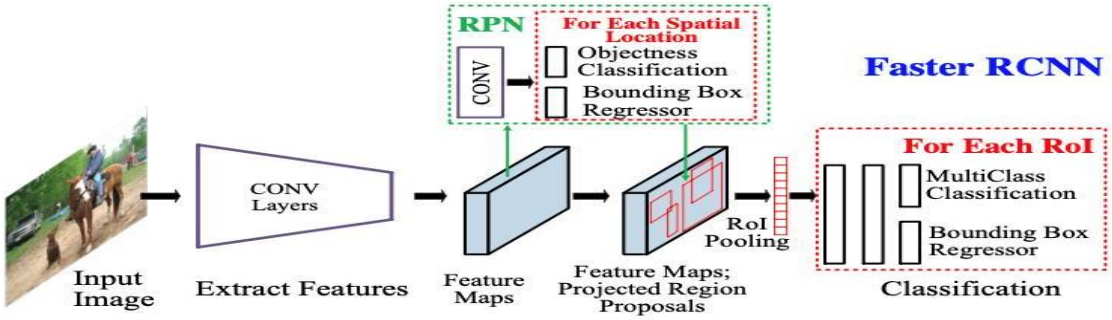


Figure 2.10: Faster-RCNN overview and its RPN module (Ren et al., 2017)

2.3.2.4. Mask R-CNN

Mask R-CNN, a region-based convolutional neural network introduced by K. He et al. in 2020, has become a prominent approach in modern object recognition and instance segmentation. It

extends the capabilities of Faster R-CNN by incorporating an additional branch responsible for generating masks corresponding to each identified object (K. He et al., 2020). The architecture of Mask R-CNN builds upon Faster R-CNN with two significant modifications: (1) replacing the RoI pooling with the Roi Align layer, and (2) introducing a segmentation branch alongside the existing regression and classification components (K. He et al., 2020). The mentioned figure (Figure 2.12) is reproduced from the work conducted by K. He et al. (2020).

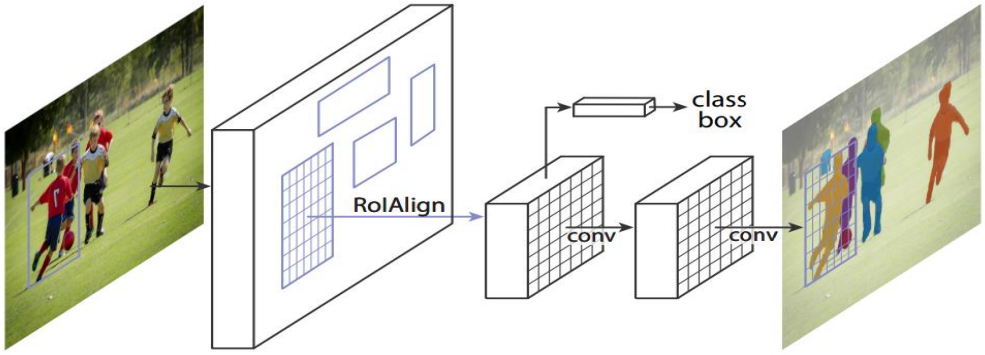


Figure 2.11: The architecture of Mask R-CNN (K. He et al., 2020).

2.4. Transfer Learning

Transfer learning is a significant area in machine learning that focuses on leveraging knowledge or models obtained in one domain to address similar problems in a different domain. (Weiss et al., 2016). The fundamental concept of transfer learning is to leverage previous experiences to address similar problems. For example, recognizing zebras can be facilitated by applying the understanding gained from identifying horses. The key idea is to learn a new representation that captures relevant information from previous tasks while disregarding irrelevant knowledge (M. Wang & Deng, 2018; Weiss et al., 2016). Transfer learning encompasses diverse concepts including inductive transfer, multi-task learning, information consolidation, meta-learning, and incremental learning. (W. Li et al., 2020; Weiss et al., 2016).

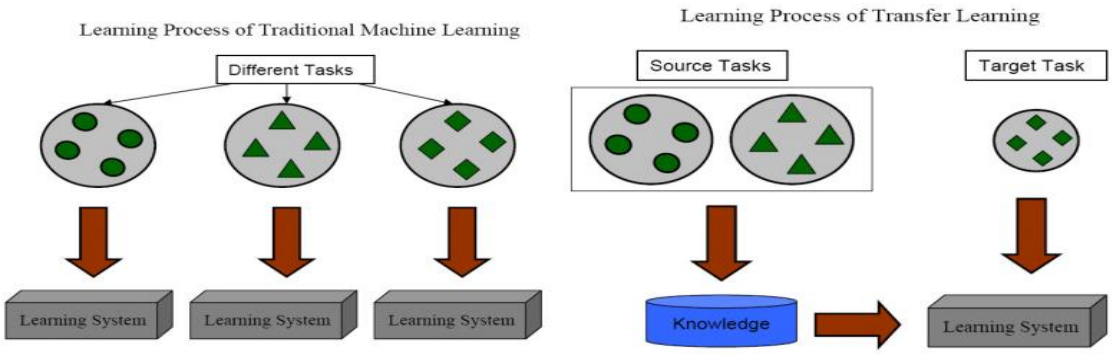


Figure 2.12: Depiction of Transfer learning (Weiss et al., 2016)

2.5. Domain Adaptation

Domain adaptation is a type of transfer learning, concentrates on transferring knowledge from a source to a target domain, aiming to overcome the challenge of domain shift between the two domains (Oza et al., 2021). This leads to a decline in performance due to variations in data distribution. The goal of domain adaptation is to improve the learning of the target predictive function within the target domain, enabling the model to adapt and achieve optimal performance despite disparities in data distribution.

Definition 1. Given a source domain $\mathcal{D}_S$ with a task$\mathcal{T}_S$, and a target domain $\mathcal{D}_T$ with a task$\mathcal{T}_T$, main goal of domain adaptation is to improve the learning of the target predictive function. $f_T(\cdot)$ in $\mathcal{D}_T$ using the knowledge in $\mathcal{D}_S$ and$\mathcal{T}_S$, where $\mathcal{D}_S \neq \mathcal{D}_T$ and $\mathcal{T}_S = \mathcal{T}_T$(Oza et al., 2021; M. Wang & Deng, 2018).

Traditional approaches to domain adaptation have thoroughly investigated the challenge of data distribution mismatch and have offered valuable insights for deep domain adaptation methods. These techniques were formulated prior to the prevalent utilization of deep neural networks in the field of machine learning and have laid the groundwork for subsequent advancements. The goal is to bridge the distribution gap between the two domains and leverage the labeled data in the source domain to improve learning and prediction performance in the target domain.(M. Wang & Deng, 2018; Weiss et al., 2016).

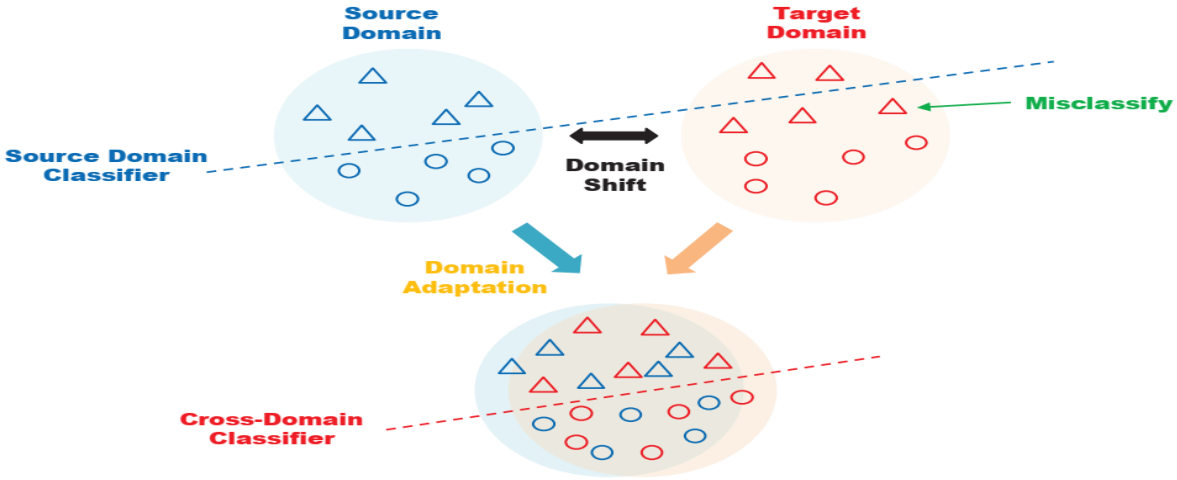


Figure 2.13: Visual representation of domain adaptation (Weiss et al., 2016)

Approaches used for Domain adaptive object detection are grouped into six Categories such as

- Pseudo-label based self-training
- Teacher-student training
- Adversarial feature learning
- Domain randomization
- Image-to-image translation
- Graph reasoning

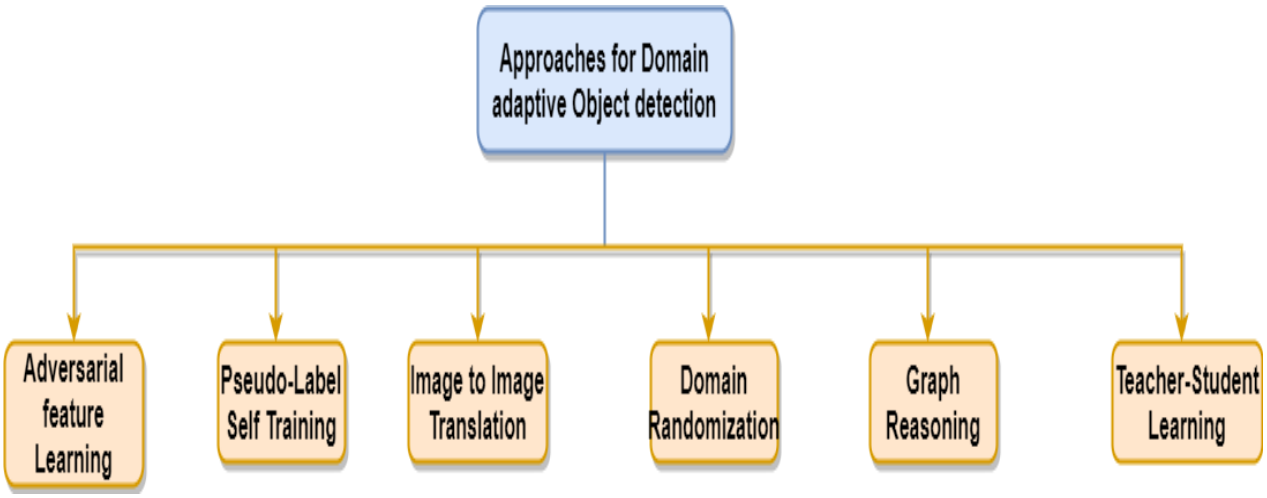


Figure 2.14: Categories of methods used in Domain Adaptive Object Detection

2.5.1. Adversarial Feature Learning through Gradient reversal

The gradient reversal layer (GRL) is a crucial component in a typical domain adaptation setup (Ganin & Lempitsky, 2015) The general approach implies reducing the upper limit by reducing the H-distance directly. Inversely correlated with distance is the domain classifier D's error rate. A GRL and a domain classifier are added as two more parts to the feature extractor's output. The GRL functions as a typical identity function during the forward pass of the DANN, during which the network tries to forecast the class and the domain labels (D'Innocente et al., 2020; Ganin & Lempitsky, 2015; Oza et al., 2021). During the back propagation, the GRL multiplies the gradient of the domain classifier by a predetermined negative weight constant.

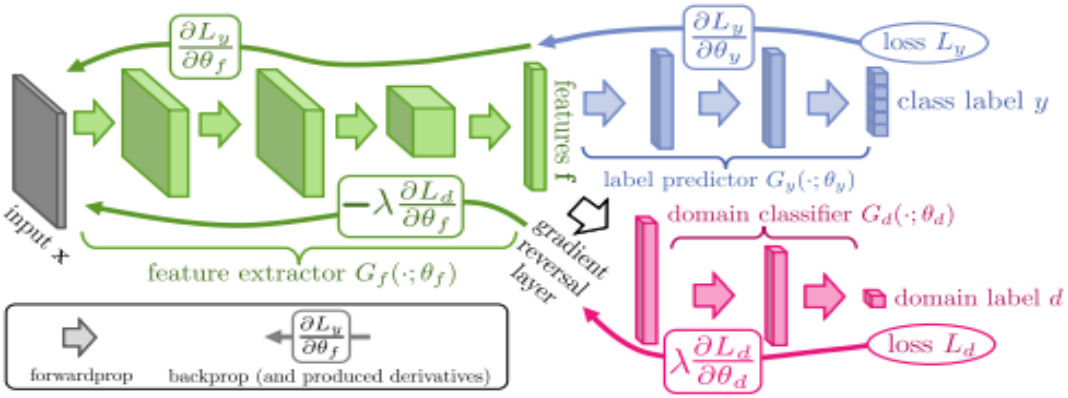


Figure 2.15: Domain-adversarial neural network and GRL Ganin & Lempitsky, 2015)

2.5.2. Pseudo-labeling-based Self-training Method

Using self-training based on pseudo-labeling is another reasonably simple approach to tackle an object identification difficulty. Pseudo-labeling is often done naively by first training the source dataset $\mathcal{D}_S = \{\mathcal{X}_S^i, \mathcal{Y}_S^i\}_{i=1}^{\mathcal{N}_S}$ and then using inference to generate labels for the target dataset $\mathcal{D}_\mathcal{T} = \{\mathcal{X}_\mathcal{T}^j\}_{j=1}^{\mathcal{N}_\mathcal{T}}$ images (Shubham.jain Jain, n.d.). The new dataset $\dot{\mathcal{D}}_\mathcal{T} = \{\mathcal{X}_\mathcal{T}^j, \mathcal{Y}_\mathcal{T}^j\}_{j=1}^{\mathcal{N}_\mathcal{T}}$ will then be formed using the labels that were produced. However, it is only reasonable that the outcomes would be noisy (Ganin & Lempitsky, 2015; Oza et al., 2021).

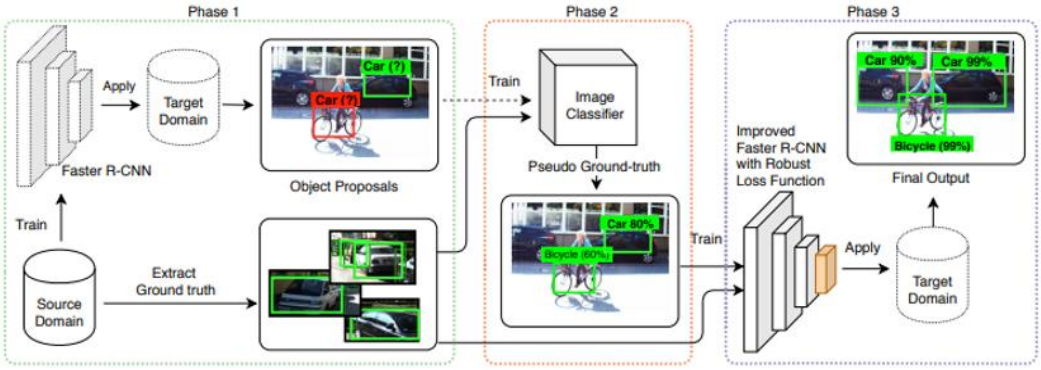


Figure 2.16: A Robust Learning Approach to Domain Adaptive Object Detection (Shubham.jain Jain, n.d.)

2.5.3. Teacher-student Learning method

Another often used method in domain adaptation and transfer learning in general is Teacher-student training (Gou et al., 2021; Tarvainen & Valpola, 2017). It comprises of two comparable models that were created utilizing two distinct training approaches. Two distinct Faster RCNN-based detector training sessions make up the training pipeline the technique of model distillation, often referred to as MT (Model Teacher). It aims to minimize the divergence between their probability predictions when subjected to various perturbations of the inputs.(Deng et al., 2021; Tarvainen & Valpola, 2017) The teacher model then receives weakly augmented data. However, the student uses images that have undergone significant editing to train their models. According to (Cai et al., 2019; Tarvainen & Valpola, 2017)weak augmentations are desired in order to create pseudo-labels of decent quality. However, stronger augmentations are necessary in the teacher model in order to improve performance and solve over-fitting. The generated pseudo-labels are then used by the student model to further address the problem of class imbalance, which often leads to the detector learning underrepresented classes poorly. Instead of a general cross-entropy loss that treats all samples equally, it was attempted to apply the multi-focal loss, which seeks to provide higher weight to the samples with lower confidence. This method, however, only addresses a semi-supervised object identification issue and overlooks the domain shift problem.

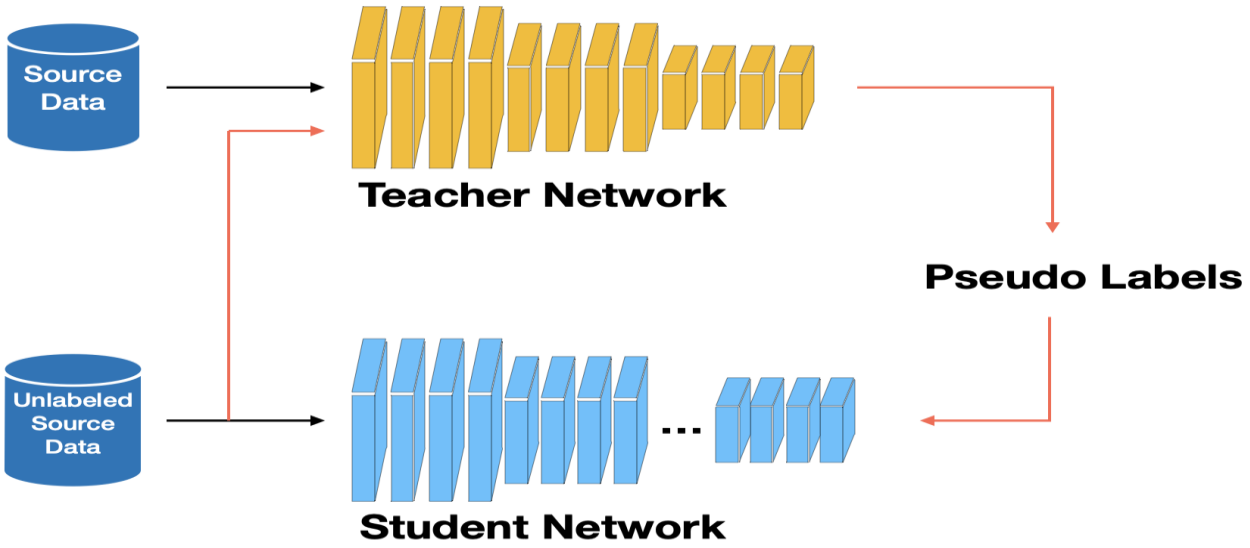


Figure 2.17: Teacher-Student Learning architecture (Tarvainen & Valpola, 2017)

2.6. Related Works

The research conducted by Y. Chen et al. (2018) made a significant breakthrough in addressing the challenge of adapting object detection to diverse domains. They proposed a pioneering approach that integrated gradient reversal training into the Faster-RCNN framework. The method involved applying adversarial training not only to the image-level features extracted from the backbone network but also to the instance-level features obtained from the RCNN network. This work provided valuable insights into leveraging adversarial training techniques for improving the adaptability of object detection models across different domains.

In contrast, (Saito et al., 2019) argued against the universal application of gradient reversal in the backbone network, highlighting that the shorter convolutional layers capture local information. They suggested that directly employing adversarial loss to learn domain-invariant local features across different domains would be more advantageous, which they termed "strong alignment.".

In their work, Hsu et al. (2020) present a progressive approach to address domain adaptation by gradually handling simpler adaptation subtasks and bridging the domain gap through an intermediate domain. They propose transforming the images from the source domain to resemble those in the target domain, creating this intermediate domain. Adversarial learning is employed to align the distributions at the feature level, effectively addressing the domain-shift problem. Furthermore, a weighted task loss is utilized to handle the imbalanced image quality within the

intermediate domain. This combined approach aims to improve the adaptation process and achieve better performance in domain adaptation tasks.

In their work, Soviany et al. (2021) propose a unique approach that combines self- paced learning and Cycle-GAN transformations for domain adaptation. Their algorithm follows a self-paced learning strategy, gradually learning from easy to hard samples, while incorporating Cycle-GAN transformations to bridge the domain gap. The authors' approach is efficient and straightforward, imposing minimal overhead during the inference stage. Ablation studies are also performed to analyze the contributions of each component in their architecture. The researchers investigate the compatibility of their framework with various object detectors. Furthermore, Soviany et al. compare their proposed difficulty measure with existing measures from the literature, demonstrating its effectiveness and its close association with the performance metric.

In their study, Wu et al. (2022) addressed the challenge of separating domain invariant and domain-specific features while obtaining instance-invariant characteristics in the context of domain adaptive object detection. They proposed a progressive disentangled framework that introduced two disentangled layers for feature decomposition. Through disentangled learning, they aimed to decompose the input features into domain-invariant and domain-specific components. The domain-invariant features were then utilized to extract instance-invariant characteristics. To enhance the disentanglement process, the authors developed a three-stage training method that incorporated different loss functions. These loss functions were designed to encourage the separation of domain-specific and domain-invariant information. The effectiveness of their approach was evaluated through experiments conducted on three different domain-shift scenarios. Overall, their progressive disentangled framework showed promising results in addressing the challenge of domain adaptation in object detection, providing a novel approach to effectively extract instance-invariant characteristics while disentangling domain-specific and domain-invariant features.

In their work, He et al. (2022) propose a novel framework called Target-perceived Dual-branch Distillation (TDD) to address the domain shift problem in object detection. The TDD framework aims to reduce the domain gap and provide accurate supervision by integrating detection branches from both the source and target domains into a unified teacher-student

learning system. This allows the framework to adaptively perceive objects in a target image using a source detector. to facilitate the learning process, the authors employ self-distillation in the two branches of the framework. This technique, known as Dual Branch Self Distillation, enables the models to gradually incorporate complementary object knowledge from multiple domains. The proposed TDD framework is extensively evaluated on various cross-domain object detection scenarios. Through these in-depth tests, the authors demonstrate the effectiveness of their approach in handling the challenges of cross-domain object identification.

Li et al. (2022) proposed a teacher-student system, called Adaptive Teacher (CDATOD), to address the domain gap in domain adaptive object detection. Their approach utilized domain adversarial learning and weak-strong data augmentation techniques. To align the distributions of features from the source and target domains, the student model underwent feature level adversarial training. This training aimed to make the student model learn domain-independent characteristics, enabling it to bridge the gap between the domains effectively. In addition, the researchers introduced weak-strong augmentation and reciprocal learning between the student model and the instructor model. The teacher model learned from the student model, acquiring information without being influenced by the domain of the source. This allowed the teacher model to adapt to the target domain more effectively.

Li et al. presented a comprehensive approach that successfully tackled the domain gap in domain adaptive object detection. Their study demonstrated promising results, showcasing the potential of the Adaptive Teacher system in improving the performance of object detection models across different domains.

2.7. Summary of Related Work

Table 2.1: Summary of Related works on domain adaptation for Object detection

Related Paper	Dataset used	Detection framework	Methodology	Limitation
Faster-RCNN in the wild (Y. Chen et al., 2018)	SIM 10k, Cityscapes, foggy Cityscapes, KITTI,	Faster-RCNN	Adversarial feature learning	✓ Evaluation is conducted on a relatively small number of datasets. ✓ Potential problem of mode collapsing
Progressive adaptation (Hsu et al., 2020)	SIM 10k, Cityscapes, Udacity, Foggy cityscape	Faster-RCNN	Adversarial feature learning, Image-to-image translation,	✓ The Synthetic Data produced by CycleGAN also introduces other kind of domain shift
Unbiased mean-teacher (Deng et al., 2021)	SIM 10k, Cityscapes Foggy Cityscapes, PASCAL, Watercolor	Faster-RCNN	Adversarial feature learning, Image-to-image, translation, Mean-teacher training,	✓ Performance drops due to not applying Pseudo labeling mechanism ✓ Problem of Scalability and Generalization to Large-scale Datasets ✓ Limited Evaluation on Diverse Domains
Instance-invariant progressive disentanglement	SIM 10k, Cityscapes , Foggy Cityscapes, KITTI,	Faster-RCNN	Adversarial feature learning, Graph-reasoning	✓ Limited Robustness to Outliers or Anomalous Data ✓ Difficulty in Handling Heterogeneous Data. ✓ Limited Adaptation to Dynamic Environments

(Wu et al., 2022)	PASCAL, Clipart, Watercolor			
Curriculum self-paced learning(Soviany et al., 2021)	SIM 10k, Cityscapes, Udacity, Foggy cityscape	Faster-RCNN	Pseudo-label based self-training, Image-to-image translation	✓ Performance drops for not applying data augmentation ✓ Problem of Handling Class Imbalance and Rare Classes ✓ Synthetic data produced introduced another Domain shift problem
Cross-Domain Adaptive Teacher (Y.-J. Li et al., 2022)	Cityscapes , Foggy Cityscapes, KITTI, PASCAL, Clipart, Watercolor	Faster-RCNN	Semi-supervised, Adversarial feature learning	✓ Performance problem due to generation of low-quality pseudo labels ✓ Problem of redundant bounding boxes ✓ Sensitivity to Parameter Initialization
Learning Domain Adaptive Object Detection with Probabilistic Teacher ((M. He et al., 2022.)	SIM 10k, Cityscapes, foggy Cityscapes, KITTI,	Faster-RCNN	Semi-supervised learning, Pseudo-labeling based self-training	✓ Performance problem due to generation of low-quality pseudo labels ✓ Problem of redundant bounding boxes ✓ Limited Generalization to Unseen Domains ✓ Sensitivity to Pseudo-labeling Thresholds

CHAPTER THREE

RESEARCH METHODOLOGY

3.1. Chapter Overview

This section presents methods resources used for this study, including the hardware, software, datasets, and algorithms. It includes Explanations of the datasets and methods used to evaluate the effectiveness of proposed approach. It outlines the methodology and scientific approach that are employed to solve the research questions. It provides as short explanation of the baseline research works that are related to the area and evaluation criteria that are used to measure the performance of the proposed approach.,

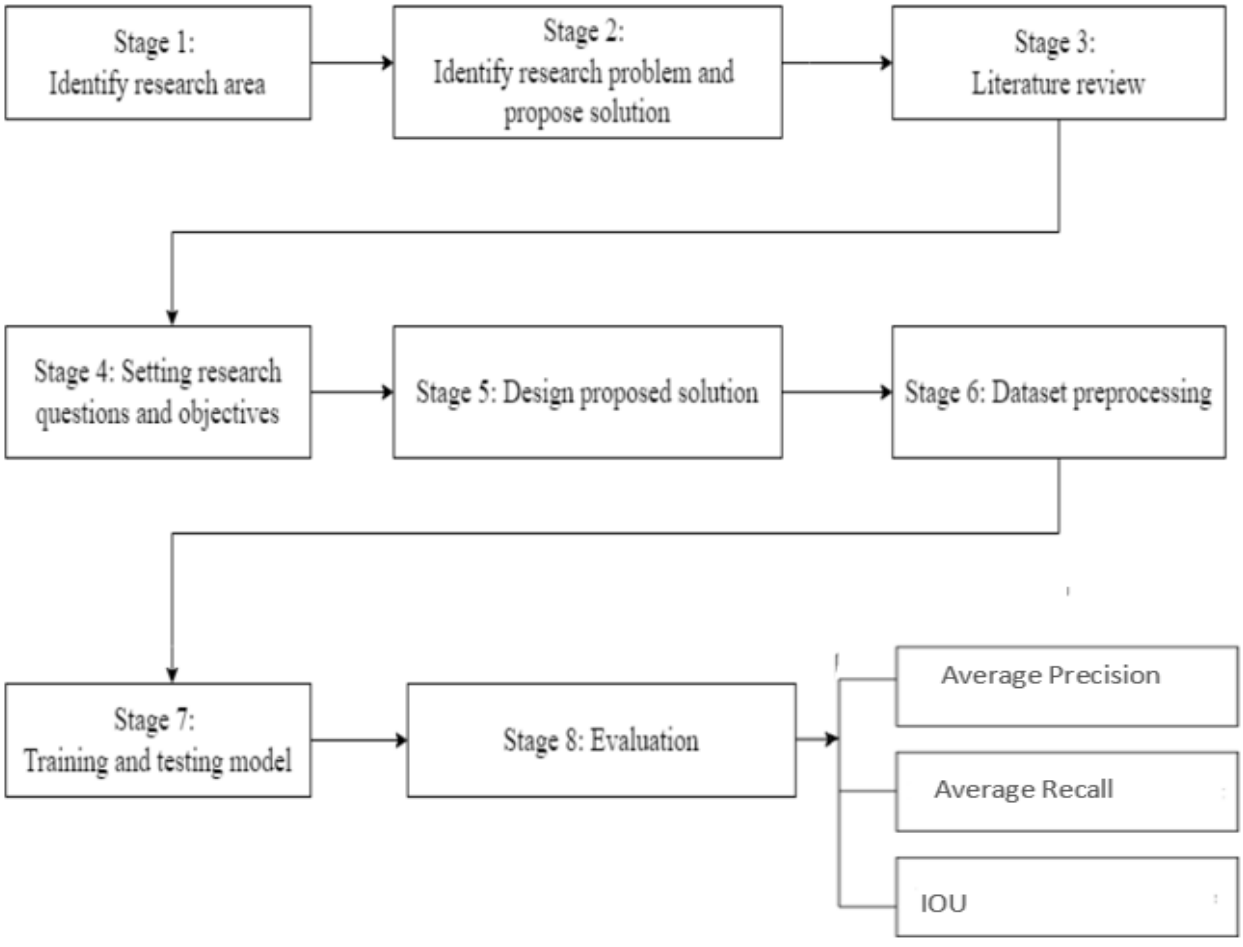


Figure 3.1: shows the workflow diagram of the domain adaptive Object detection approach

3.2. Datasets

The evaluation of the domain adaptive object detection techniques has been conducted using a range of datasets. Specifically, Cityscapes to Foggy Cityscapes, SIM10K to Cityscapes, and Cityscapes to KITTI. The evaluation is performed using both the target training data, which does not have annotation information, and the source training data, which includes annotation information such as bounding boxes and object categories.

3.2.1. Cityscapes

The Cityscapes dataset, introduced by Cordts et al. (2016), is a widely used real- world urban scene dataset primarily used for semantic segmentation tasks. It comprises a training set of 2,975 samples, a validation set of 500 samples (including fine pixel-level annotations), and an additional set of 20,000 images with coarse annotations. The target domain for the study is the unlabeled training set of 2,975 samples to ensure fair comparisons. In the evaluation protocol, the models are tested and performance results are reported by evaluating them on the 500-sample validation set. Cityscapes is renowned for its ground-truth semantic segmentation annotations, making it a standard benchmark for pixel-level annotation tasks. It is considered one of the largest datasets with such annotations. In the experiments conducted, the Cityscapes dataset serves as source domain dataset, while the target domain dataset is Foggy Cityscapes, introduced by Sakaridis et al. (2018). Foggy Cityscapes introduces challenging weather conditions, such as fog, to the urban scenes, adding complexity to the task. The dataset consists of various categories, including vehicles, trucks, motorcycles/bikes, trains, buses, riders, and humans. These categories are the object classes used for semantic segmentation in the experiments.

Figure 3.2: sample images from cityscape dataset(Cordts et al., 2016)

3.2.2. Foggy Cityscapes

Foggy Cityscapes (Sakaridis et al., 2018) is an extension of the original Cityscapes dataset that incorporates artificially synthesized fog. The annotation information in Foggy Cityscapes remains the same as the annotations in the original Cityscapes dataset. To create the Foggy Cityscapes dataset, a fog simulation pipeline was employed. The dataset consists of a total of 20,550 outdoor scene images, sourced from the Cityscapes dataset. By applying these fog conditions to the original Cityscapes images, the Foggy Cityscapes dataset was generated. The Foggy Cityscapes dataset enables researchers to evaluate and develop models that are robust to foggy conditions. It provides a valuable resource for studying object detection and other computer vision tasks in the presence of simulated fog.

3.2.3. The SIM10k Dataset

The dataset SIM10K dataset (Johnson-Roberson et al., 2017) is a synthetic dataset generated using the Grand Theft Auto (GTAV) engine. The objective is to evaluate the model's ability to adapt from synthetic visuals to real-world scenarios by performing domain adaptation from SIM10K to Cityscapes. It is important to note that only the automobile class is considered in this adaptation process, focusing on object detection specifically within this category.

3.2.4. The KITTI Dataset

The KITTI dataset is a used as benchmark dataset for autonomous driving and computer vision tasks. It consists of a total of 14,999 examples, which are divided into 7,481 training examples and 7,518 testing examples. Each example in the KITTI dataset contains two images captured by a forward-facing stereo camera rig, providing stereo vision. Additionally, a LiDAR (Light Detection and Ranging) point cloud captured by a Velodyne sensor is available for each example. The dataset covers various scenarios, including urban areas and highways in the German city of Karlsruhe. The images in the KITTI dataset were captured during daylight and clear weather conditions. A typical training example includes an image and the corresponding point cloud, as shown in Figure 2.1 of the reference. The bounding boxes are provided for cars, pedestrians, and cyclists that are visible in the images. The 2D bounding boxes are parameterized using pixel coordinates, represented as (umin, umax, vmin, vmax), where (umin, vmin) denotes the pixel coordinates of the top-left corner of the bounding box.

The KITTI dataset serves as a valuable resource for developing and evaluating algorithms related to object detection, tracking, and 3D scene understanding in the context of autonomous driving and computer vision.

3.2.5. BDD100K dataset

The BDD100K dataset, introduced by Yu et al. in 2020, is a large-scale dataset focused on collecting data through crowdsourcing. It comprises more than 100,000 RGB video sequences recorded at a resolution of 1280x720. The dataset encompasses various regions across the United States. Alongside the video sequences, GPS data is also provided. The annotated data in the BDD100K dataset includes various annotations such as bounding boxes, scene types, and instance segmentation. With its large-scale and diverse data, it enables researchers to tackle various challenges in computer vision and autonomous driving.

3.3. Pre-processing Techniques

Data preparation and analysis are crucial steps that should be performed before developing or selecting a neural network for training on a specific dataset. These stages help in understanding the dataset and making informed decisions about the neural network design and hyperparameters. Data preparation involves converting the dataset into a format that is compatible with the neural network model. This may include tasks such as data cleaning, normalization, scaling, and feature extraction. By conducting data preparation and analysis, researchers and practitioners can ensure that the dataset is suitable for training a neural network. It allows for a better understanding of the data's characteristics, potential challenges, and the necessary adjustments or preprocessing steps required to achieve optimal performance with the neural network model

- **Image resizing**: Since the original images have varying dimensions, they were resized to a fixed size of 256x256. Bounding boxes were employed to achieve this, with each image having its own bounding box. Now let's explore the normalization techniques.
- **Image normalization**: This method adjusts the pixel intensity values in an image. The image was scaled and adjusted to a specific range. The normalization formula used is as follows:

$$image = \frac{image - mean}{std}$$

Here, the mean and standard deviation (std) values were set to 0.5. To perform the normalization, the image is converted to a tensor in the expected range using PyTorch functions.

3.4. Development Tools

During the development phase, a range of tools were used, which encompassed prototype development tools, UML modeling tools, and other indispensable resources. The subsequent section presents a concise summary of these development tools, outlining their key features and functionalities.

3.4.1. Design Tools

Design tools encompass techniques utilized to generate, present, and comprehend design concepts. In this project, Microsoft Visio, a versatile software application, was utilized for this purpose. Microsoft Visio streamlines the process of creating visually appealing diagrams, including flowcharts, network diagrams, and flowchart diagrams, while ensuring a professional finish.

3.4.2. Hardware Tools used

The following hardware tools will be required to carry out this experiment.

Table 3.1: Hardware tools used for the study

No.	Tools	Used for
1.	GPU	To accelerate and optimize the processing of graphics and visual data
2.	Hard Disk	To store datasets, training data, and model weights
3.	RAM	To speed up the training process by facilitating faster data transfer between the CPU.

Various software and coding tools were used to implement the proposed approach such as:

- **TensorFlow**: A powerful, open-source numerical analysis and processing framework that enables developers to build machine learning applications using various tools and frameworks.

- **Keras**: is a Python-based interface that provides easy access to a machine learning framework that is freely available. It acts as a convenient interface for utilizing TensorFlow, allowing users to interact with and utilize the functionalities of TensorFlow in a simplified manner.
- **PyTorch** A popular Python-based machine learning framework that is widely used for deep learning research
- **Anaconda**: is a software distribution, to install the most recent version of Python that supports 64-bit systems. This installation includes not only Python but also various modules and integrated development environments (IDEs) that are commonly used in Python development.
- **Jupyter Notebook**: is a popular integrated development environment (IDE) that is widely used for Python programming. It is particularly favored by researchers working in the field of artificial intelligence.
- **Python**: is a high-level, interpreted and object-oriented programming language. It is widely utilized for a variety of purposes, including automating tasks and conducting data analysis.

3.5. Baseline Works

The base model used in this study is a conventional Faster R-CNN network (Ren et al., 2017) with a ResNet backbone that has been pre-trained on the ImageNet dataset. The training process exclusively involves the source domain, while the testing is performed on various target domains. To implement this, a publicly available PyTorch-based implementation of Faster R-CNN, developed by Facebook, was utilized. The subsequent sections will delve into two main approaches employed to address the challenge of object detection within a fully unsupervised domain adaptive object detection framework.

Cross-Domain Adaptive Teacher for Object Detection (Y.-J. Li et al., 2022) To address the domain gap and improve cross-domain object identification, the researchers proposed an Adaptive Teacher (CDATOD) system that employs a teacher- student framework. This system incorporates domain adversarial learning and weak- strong data augmentation techniques. The AT system focuses on training a student model to learn domain-independent features by undergoing feature-level adversarial training. This process

ensures that the feature distributions between the source and target domains become comparable, enabling the student model to generalize well across domains.

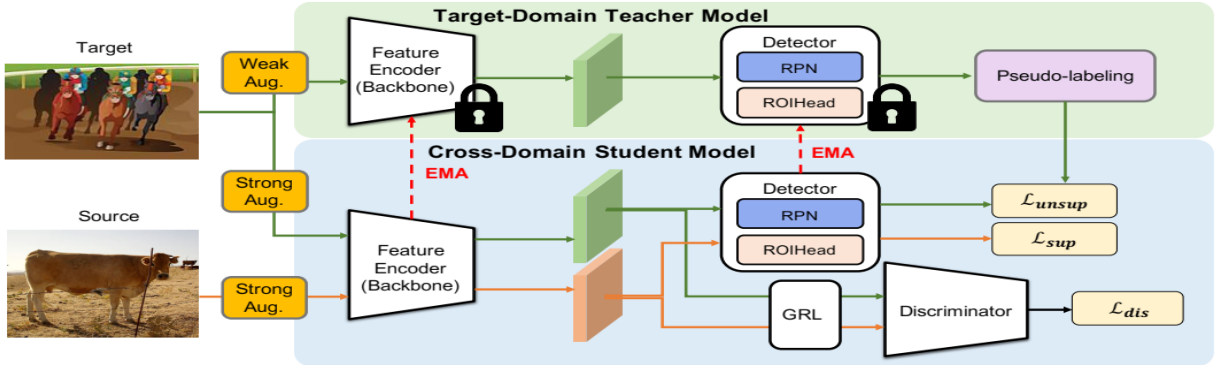


*Figure 3.3: Cross-Domain Adaptive Teacher for Object Detection (*Y.-J. Li et al., 2022)

Learning Domain Adaptive Object Detection with Probabilistic Teacher ((M. He et al., 2022.)**):** The authors propose the Probabilistic Teacher (PT) framework, which aims to capture the uncertainty of unlabeled target data using a progressive teacher-student approach. This framework supports the learning process of the student by incorporating uncertainty-guided consistency training for categorization and localization adaptation. In addition to localization adaptation, the authors concurrently perform anchor adaptation as anchors can be considered learnable parameters. To enhance uncertainty-guided self-training, they introduce the Entropy Focal Loss (EFL) within this framework.

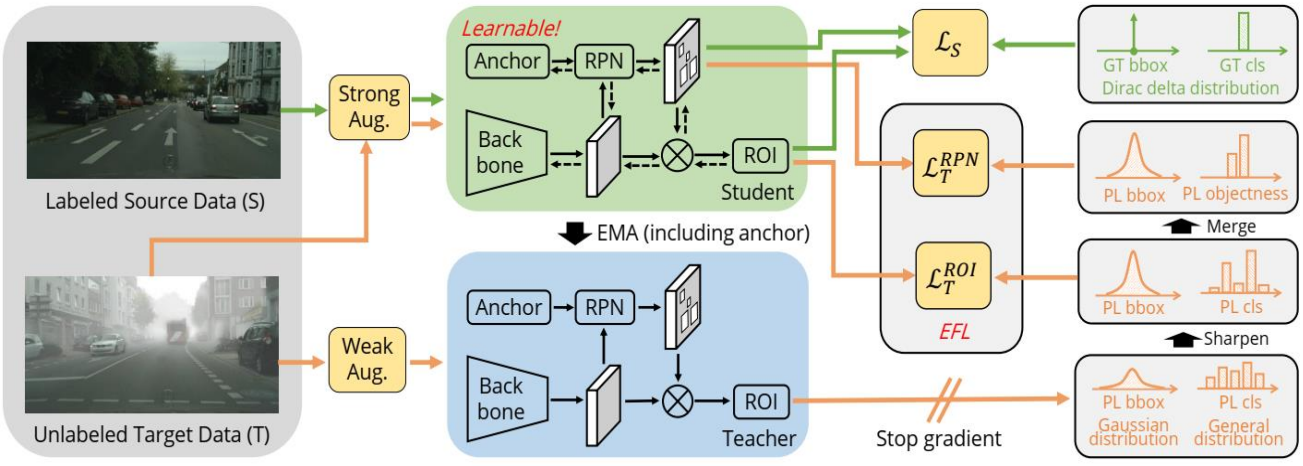


*Figure 3.4: Learning Domain Adaptive Object Detection with Probabilistic Teacher ((*M. He et al., 2022.)*)*

3.6. Evaluation Methods

Microsoft Common Objects in Context (MS COCO)(Lin et al., 2014) is widely used for evaluating the performance of trained object detection models. MS COCO is specifically designed for tasks like object detection and segmentation. It serves as a benchmark dataset for measuring the accuracy and robustness of object detection algorithms

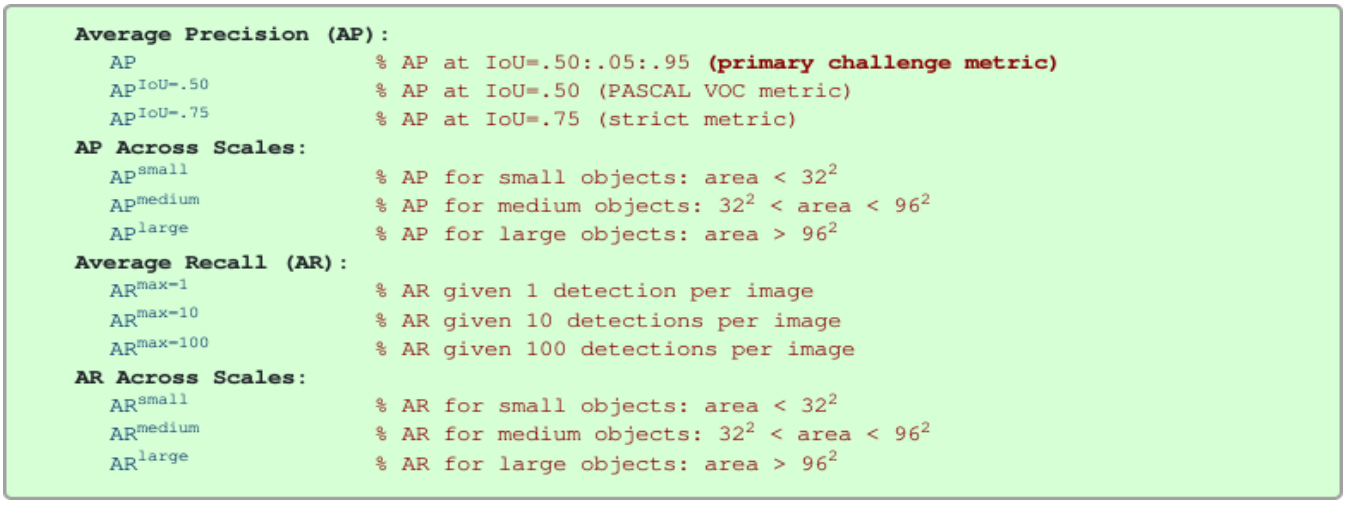


Figure 3.5: COCO Evaluation Metrics

Source: (Lin et al., 2014)

3.6.1. Precision and Recall

To assess the effectiveness of an object detection algorithm, two primary metrics, Precision and Recall, are employed. These metrics are defined by the following formulas.

$$Precision = \frac{True\ positives}{True\ positives + False\ positives} \quad Recall = \frac{True\ positives}{True\ positives + False\ negatives} \qquad Equation.3.1$$

Precision is a metric used to evaluate the accuracy of object detection. It quantifies the proportion of correctly predicted positive instances (true positives) out of all the predicted positive instances (true positives + false positives). A high precision value indicates that the algorithm has a low tendency to incorrectly identify objects, resulting in a more accurate detection. On the other hand, recall, also known as sensitivity or true positive rate, measures the algorithm's ability to correctly identify positive instances among all the actual positive instances. It is calculated as the ratio of true positives to the sum of true positives and false negatives. A high recall value signifies that the algorithm can successfully detect a larger proportion of the true positive instances present in the image.

3.6.2. Average Precision (AP)

Average Precision (AP) is a commonly used evaluation metric in object detection tasks. It provides a single numerical value that summarizes the precision-recall curve across different levels of recall. The formula to calculate AP involves computing the weighted sum of precisions at various thresholds, with the weights determined by the change in recall between consecutive thresholds.

$$AP = \sum_{k=0}^{k=n-1} (recall\,s(k) - recall\,s(k+1)) \quad {}^{*}precision\,s(k) \qquad \textit{Equation 3.2}$$

The average precision (AP) was established in VOC2007 and is often used to assess object detectors (Lin et al., 2014). The area under the precision-recall curve is used to get the average accuracy for each category. The following are the definitions of precision and recall.

3.6.3. Confidence Score and Intersection over Union

The confidence score in object detection refers to the probability that a bounding box contains an object of interest. To evaluate the accuracy of the predicted bounding box, the Intersection over Union (IoU) metric is commonly employed. IoU is computed by dividing the intersection area between the predicted bounding box (Boxp) and the ground-truth bounding box (Boxgt) by the union area of the two boxes.

$$IoU = \frac{Area(Box_{gt} \cap Box_p)}{Area(Box_{gt} \cup Box_p)} \qquad \textit{Equation 3.3}$$

It takes values between 0 and 1. In general, a threshold must be established to differentiate a legitimate detection from the rest.

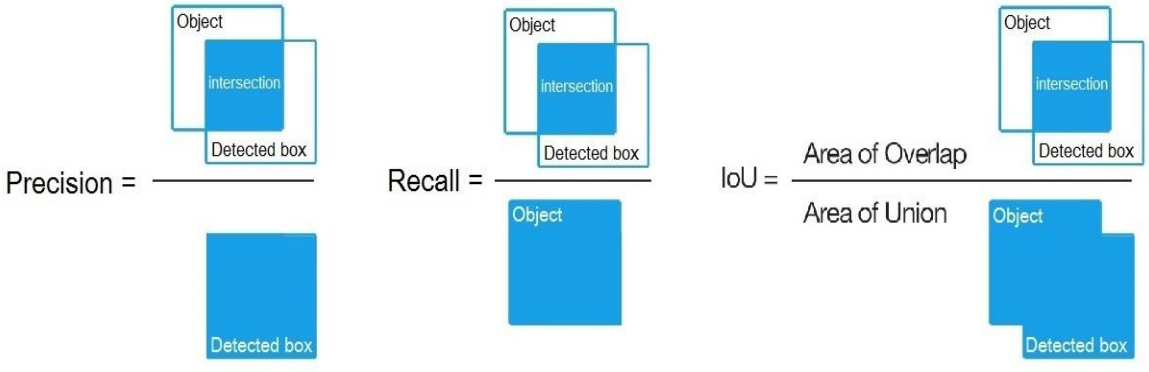


Figure 3.6: Graphical representations of precision, recall and IoU

Source: Adapted from (Aqeel Anwar, 2022.)

CHAPTER FOUR

PROPOSED MODEL ARCHITECTURE

4.1. Chapter Overview

The proposed solution architecture for the domain adaptive object detection algorithm is presented in this chapter. It includes detailed explanations accompanied by diagrammatic representations and mathematical expressions. The chapter is structured into six sections, each addressing a specific aspect of the algorithm and its implementation

- Introduction of the model architecture for detecting objects from various domains.
- Discussion of the overall structure, including the Faster RCNN Network, Feature Extraction Network, Fully Connected Layers, and Classification and Regression Network.
- Pseudo-label Generation and Selection.
- Loss functions, including Classification Loss.
- Mean Average Precision (mAP) metric for overall performance evaluation.
- Pseudocode for training the model.

4.2. Proposed Model Architecture

The proposed Deep Domain adaptive object detection approach consists of two Faster RCNN which follows a typical semi-supervised object detection framework, both networks use the same architecture, but are in charge for different mission.

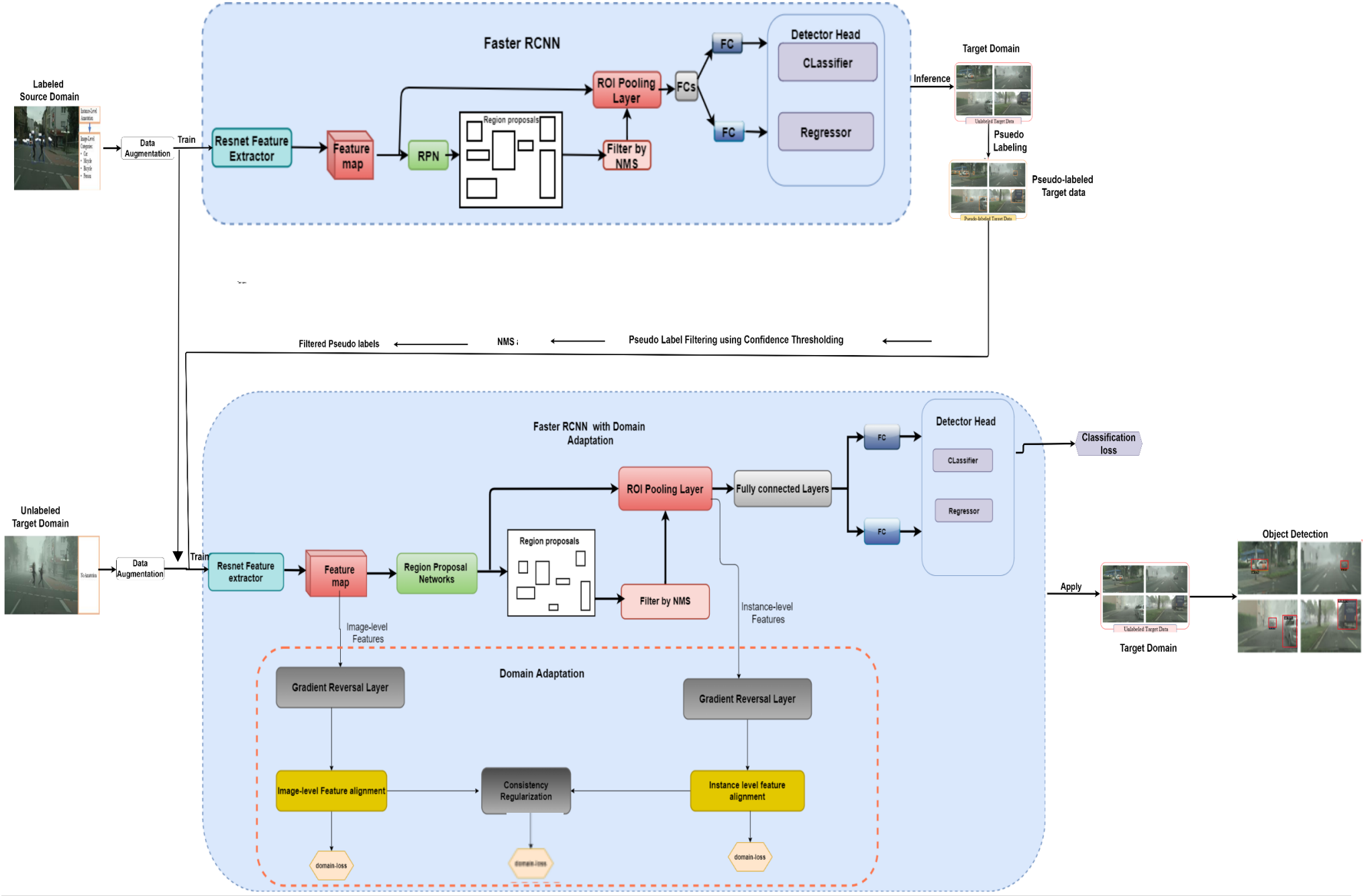


Figure 4.1:Archtecture of the Proposed Domain Adaptive Object Detection Approach

The Faster RCNN model takes augmented source images target Images and train with labelled source and pseudo-labelled target images. To address the lack of ground-truth labels for unsupervised data, the researchers adapt the pseudo-labeling method. Pseudo-labels are generated from the first network and utilized to train the second network. This integration of semi-supervised and self-training concepts facilitates training with unsupervised data.

4.3. Data Augmentation

To improve the detection performance of object detection models in diverse domains, the proposed method employs a data augmentation strategy. This strategy involves applying a range of augmentation techniques to the training data, including image rotation, scaling, flipping, and noise injection. These augmentations introduce variations in the training samples, allowing the model to learn from a wider range of scenarios and improve its robustness and generalization capabilities. It increases the diversity of data applied to faster RCNN object detectors and improve its learning performance. Data augmentation such as color, sharpness, contrast, add gaussian noise and scaling are initially applied to the images for training which serve as the input for the first network. Highly augmented images make the network work more challenging and make them to acquire more accurate representations. Augmentations on the first network used to increase the likelihood that the faster RCNN to provide accurate pseudo-labels. Firstly, augmented labeled data is used to trains the first Faster RCNN, Next, augmented unlabeled images are used to provide pseudo-detection results using the trained teacher object detector. To augment the unlabeled samples, a weak augmentation is applied, which introduces minor variations to the original data. Additionally, K strong augmentations, which involve more significant transformations or distortions, are performed on the unlabeled samples.

$$\hat{y}_u = f\big(weak(x_u)\big) \qquad \textit{Equation 4.1}$$

4.4. Source Domain Faster RCNN Model

The Faster R-CNN (Ren et al., 2017) object detection framework serves as a foundational detector, employing a two-stage approach. It can be categorized into three main components: (1) a feature extractor that uses a convolutional neural network to extract image features, (2) a region proposal network (RPN) that generates candidate regions (RoIs), and (3) a region of interest (RoI) head responsible for fine-tuning the classification and regression on the RoIs obtained from the RPN.

Initially, the Faster R-CNN model is trained on a labeled dataset using supervised learning, enabling it to make predictions on new, unlabeled dataset. These predictions are then used as pseudo-labels for the unlabeled images. To enhance the training process, random augmentation techniques, such as random cropping, rotation, scaling, and color transformations, are applied to the training images. These variations introduce input noise, making the model more robust to diverse real-world conditions. The pseudo-labels inferred from the first model serve as a form of weak supervision for training the second model, and the inclusion of random augmentation enriches the training data, leading to improved object detection.

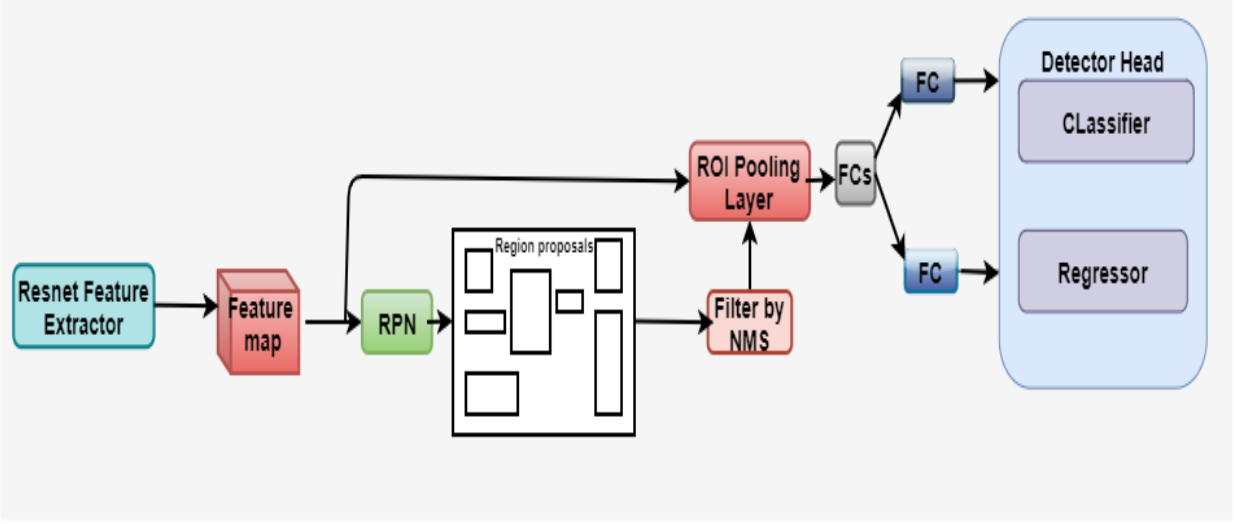


Figure 4.2:Architecture of the first Faster RCNN model for Pseudo label Generation

4.4.1. Resnet Feature Extractor

The ResNet with Feature Pyramid Networks (FPN) serves as the backbone for feature extraction. The ResNet architecture consists of Convolutional layers that are divided into five stages. The feature pyramid network (FPN) uses these stages to produce the final feature map. After the feature map is extracted, the Region Proposal Network (RPN) uses it as input and feeds it into a convolutional layer to generate region proposals. (W. Li et al., 2020) . In domain adaptive Faster R-CNN, the ResNet model is used as a feature extractor for both the RPN and the R-CNN components of Faster R-CNN. The pre-trained ResNet model, typically trained on a large-scale dataset such as ImageNet, is used to extract high-level features from input images. These features capture the visual representations of objects in the images, which are then fed into the subsequent stages of the object detection pipeline. Domain adaptive ResNet involves

adapting the pre-trained ResNet model to the target domain by adjusting its parameters to better align with the target domain's data distribution.

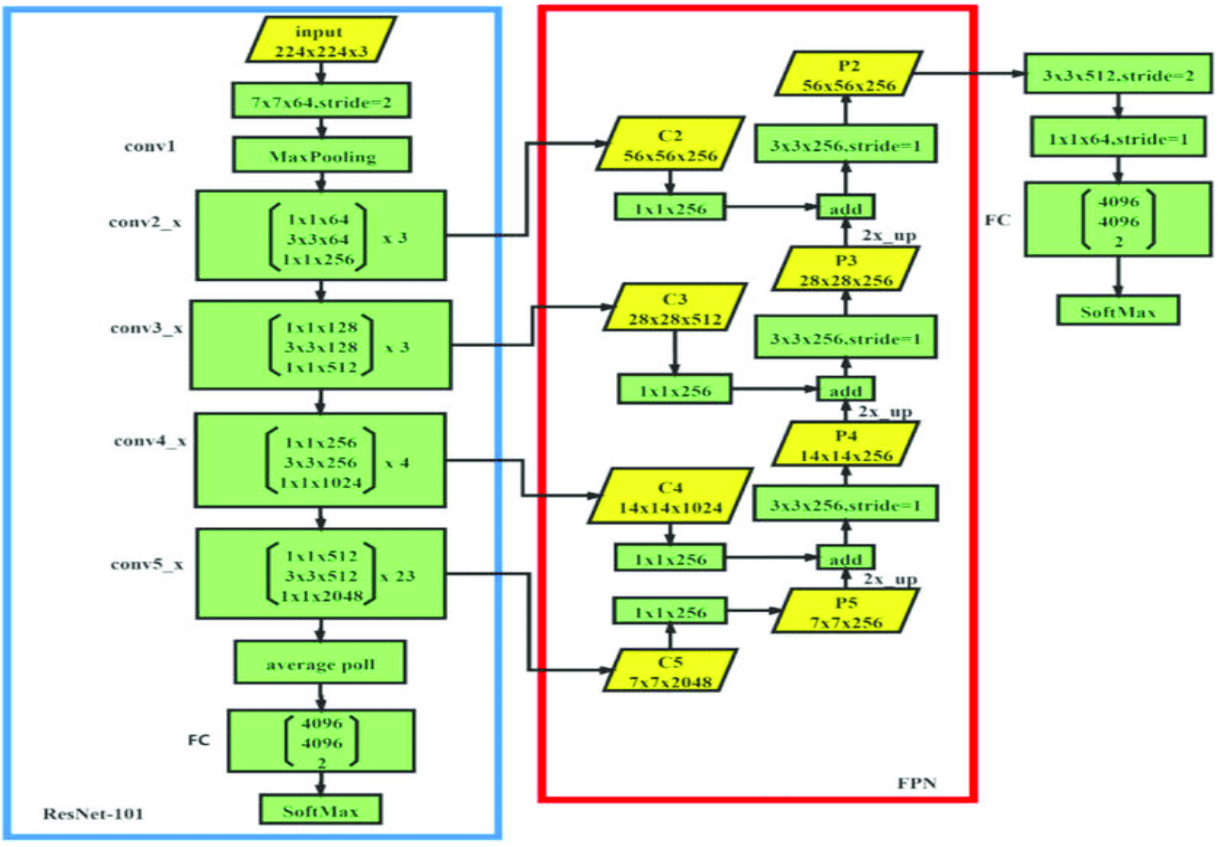


Figure 4.3:The Resnet Feature extraction network with Feature pyramid network (FPN) (Ben, Huang, 2022)

4.4.2. Region Proposal Generation

The Region Proposal Networks (RPN) takes an image as input and output a collection of proposal boxes, and produce feature maps. Each ROI having a score that represents the probability of finding an object and its coordinates in relation to the input image. This network employs SoftMax to determine if an anchor belongs in the foreground or background, and then bounding box regression to adjust the anchor in order to provide approximate proposals. At the domain adaptive faster RCNN network after the feature map produced by the RPN convolutional layer, the region-level domain classifier is applied for successfully predicting the domain label. To achieve domain adaptation in region proposal generation, several techniques can be employed. These techniques aim to leverage knowledge from a source domain, where

abundant labeled data may be available, and transfer it to the target domain, which may have limited labeled data or come from a different distribution.

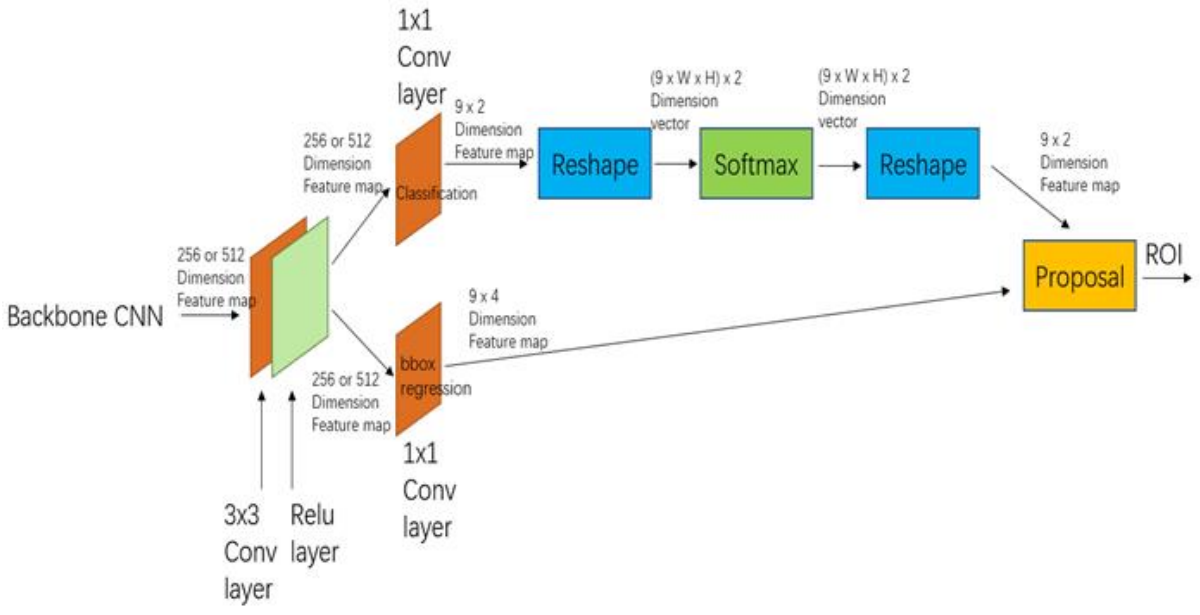


Figure 4.4:The Region Proposal Generation (Ben, Huang, 2022)

4.4.3. ROI Pooling

The Fast R-CNN architecture incorporates a pooling layer called Spatial Pyramid Pooling (SPP) to handle non-uniform input sizes and obtain a fixed-sized feature map. This pooling layer plays a crucial role in accelerating both the training and testing processes while enabling end-to-end training. In the Fast R-CNN model, the proposals generated by the Region Proposal Network (RPN) and the feature map obtained from the last layer of VGG16 (a popular backbone network architecture) are used as input. These proposals represent potential object locations in the image. The SPP layer operates on these proposals to obtain a fixed-size proposal feature map.

The SPP layer performs pooling operations at multiple scales on each proposal region. It divides each proposal into several bins or grids, where each bin represents a specific region of interest. In the case mentioned, the SPP layer utilizes three scales of pooling layers: 1x1 bin, 2x2 bin, and 4x4 bin. For each bin size, max pooling is performed independently within that bin. This means that the maximum value within each bin is

extracted, resulting in a feature vector for each bin. In this case, a total of 21 bins are used, corresponding to the combination of the three bin sizes. The 21-dimensional feature vectors obtained from the max pooling operation are then concatenated to form a unified feature representation for each proposal.

This pooled feature representation is subsequently fed into fully connected layers for target recognition and positioning. The fully connected layers process the pooled features to make predictions for the class labels and bounding box regression tasks.

Adaptive ROI pooling adjusts the pooling sizes according to the characteristics of the region proposals in the target domain. Rather than using a fixed grid size (e.g., 7x7) for pooling, the pooling sizes can be dynamically determined based on the sizes or aspect ratios of the region proposals. Adaptive ROI pooling can better accommodate variations in the target domain and capture object details effectively

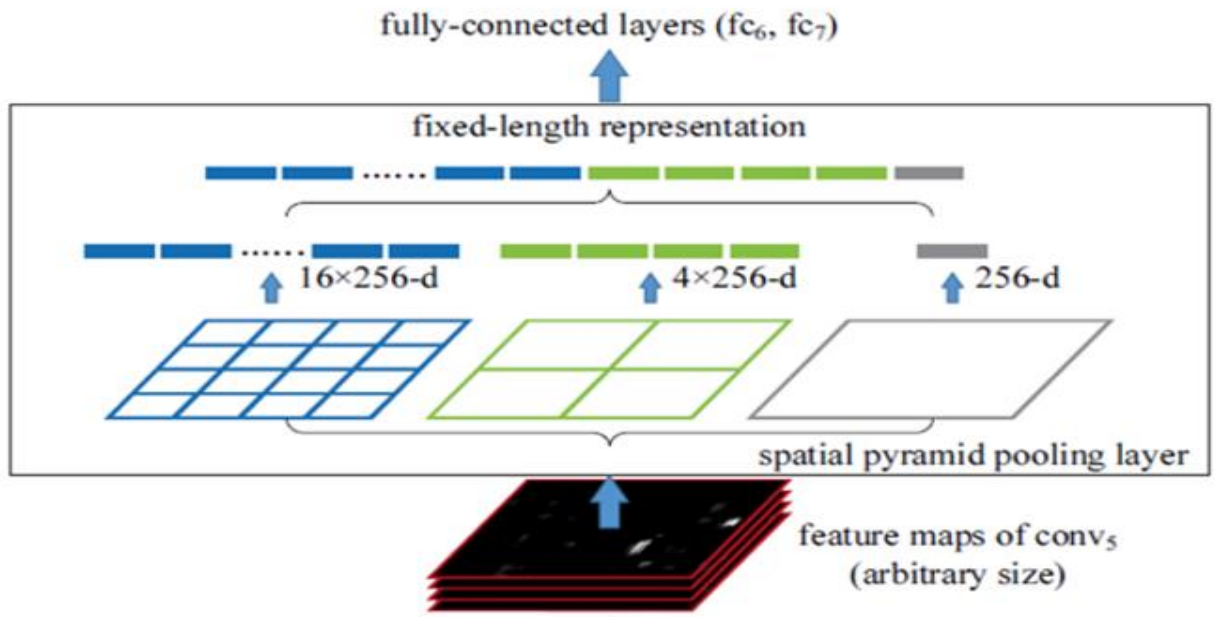


Figure 4.5: The ROI Pooling Layer (Ben, Huang, 2022)

4.4.4. Classification and Regression Sub-Network

The Classification Sub-network within Faster R-CNN is responsible for determining the probability of object presence for each class and anchor box. It consists of a Fully Convolutional Network (FCN) architecture with four convolutional layers. The number of filters in each layer is equivalent to the number of channels in each pyramid layer (C).

The final layer of the Classification Sub-network is a 3x3 convolutional layer with K filters, where K represents the number of anchor boxes used in the object detection task. This layer outputs the class probabilities for each anchor box, indicating the likelihood of each class being present.

On the other hand, the Regression Sub-network is an independent FCN that runs parallel to the Classification Sub-network. Its primary objective is to regress the offsets between each anchor box and its corresponding ground truth bounding box. Unlike the Classification Sub-network, the Regression Sub-network does not consider class information during the prediction. It focuses solely on estimating the bounding box offsets, making it class neutral.

The architecture of the Regression Sub-network is similar to the Classification Sub-network, with the same number of convolutional layers and filter sizes. However, the parameters between the two networks are different. The Regression Sub-network produces a linear output of four points that form an anchor box. .

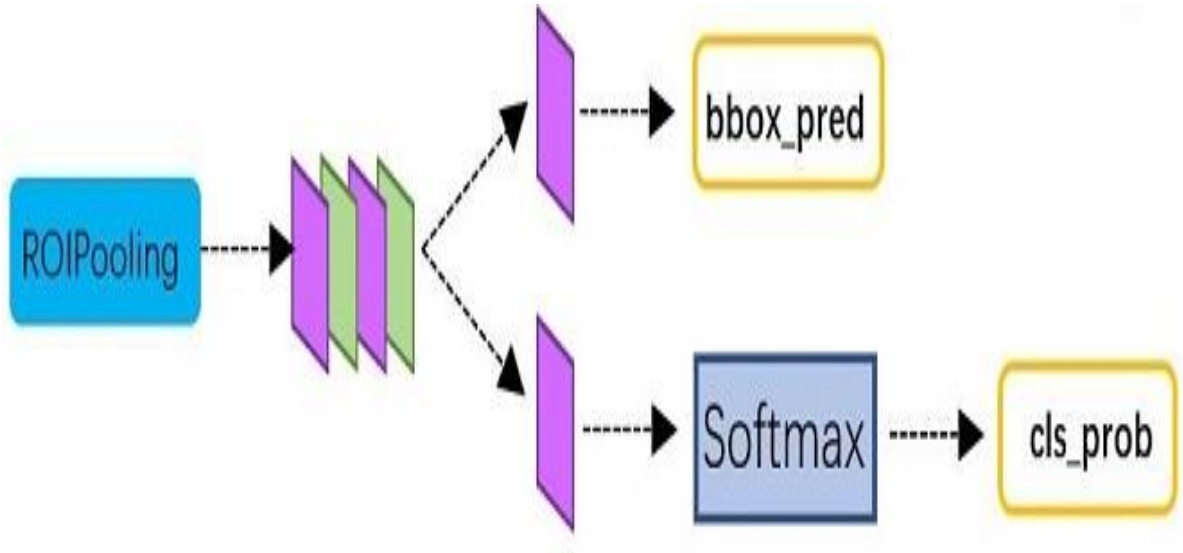


Figure 4.6:The Classification part of the network structure

4.5. Pseudo Labelling

Pseudo labeling is a technique employed to generate labels for training object detectors using images from the target domain. It addresses the challenge of lacking ground-truth labels for data in the target domain. (L. Wang & Yoon, 2022).

4.5.1. Pseudo Label Generation

The generation of pseudo labels for the second model is accompanied by certain measures to ensure their quality. One such measure is the application of a confidence threshold for the predicted bounding boxes. This threshold serves as a cutoff point, where any predicted bounding boxes with confidence scores below the threshold are considered to have low confidence. By filtering out these low-confidence predictions, the model eliminates potentially false positive samples. This helps prevent the accumulation of noisy pseudo-labels, which could adversely affect the training process of the second model. The filtered pseudo- labels are of higher quality, as they are derived from predictions that surpass the confidence threshold.

Furthermore, class-wise non-maximum suppression (NMS) is employed to remove redundant box predictions. NMS is applied separately for each class, and it ensures that only the most accurate and representative predictions are retained while discarding overlapping and less confident predictions. This post-processing step improves the precision and reliability of the pseudo-labels. The model used for selecting pseudo-labels in the second model assigns each unlabeled sample to the class with the maximum prediction probability. This ensures that the pseudo-labels correspond to the most confident class predictions, further enhancing their quality. In the fine-tuning stage with dropout, the pseudo-labels can be utilized.

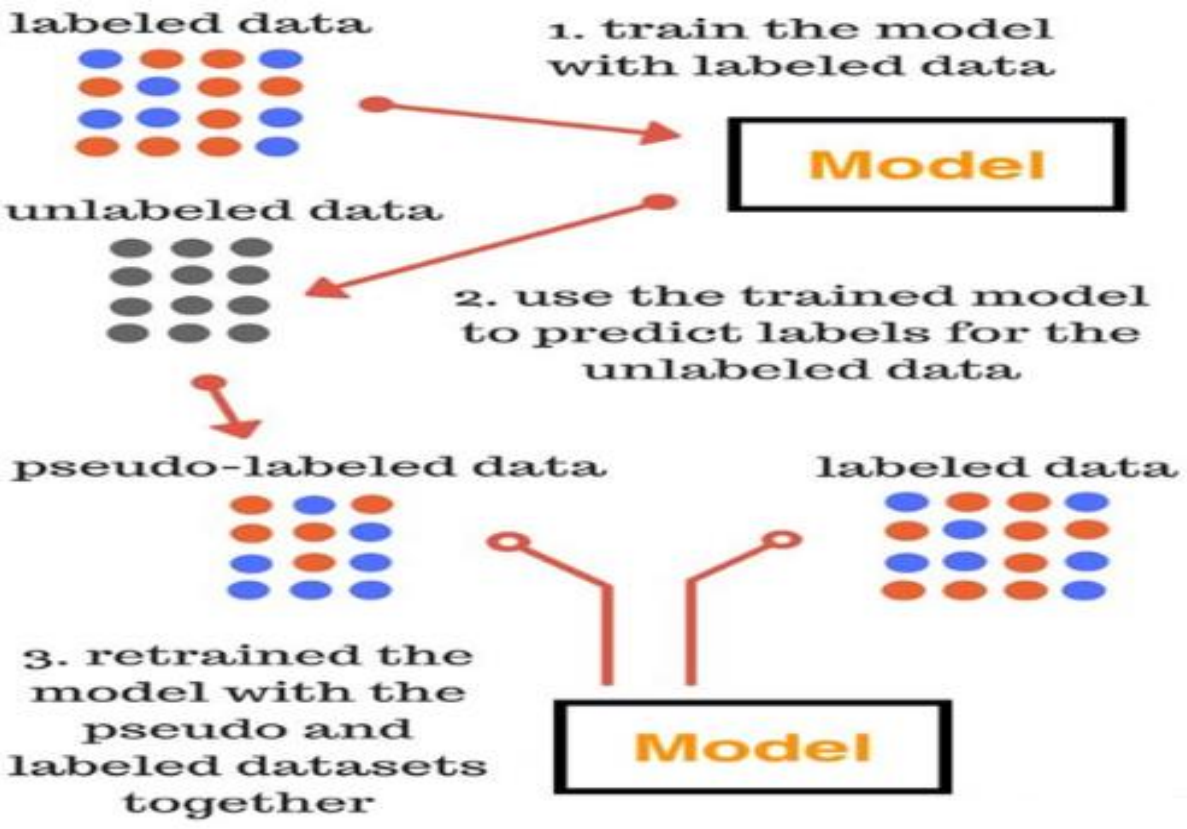


Figure 4.7:The pseudo-labeling process

The weight of the pseudo-label part is adjusted by a (t). If a (t) is particularly small, the pseudo-label will not work.

$$L = \frac{1}{n}\sum_{m=1}^{n}\sum_{i=1}^{C} L(y_i^m, f_i^m) + \alpha(t)\frac{1}{n'}\sum_{m=1}^{n'}\sum_{i=1}^{C} L(y_i'^m, f_i'^m), \qquad \textit{Equation 4.2}$$

Where a (t) is a deterministic simulated annealing process that helps to avoid poor local minima during optimization, making the pseudo labels of unlabeled data as similar as possible to the real labels.

$$\alpha(t) = \begin{cases} 0 & t < T_1 \\ \frac{t-T_1}{T_2-T_1}\alpha_f & T_1 \le t < T_2 \\ \alpha_f & T_2 \le t \end{cases} \qquad \textit{Equation 4.3}$$

4.5.2. Pseudo-Label Selection

There is a probability that noise and false positive samples are present in low confidence pseudo-labeled samples, which may harm the performance of the model. Samples with highly

confidence pseudo-labels have been chosen in order to get high quality training performance from unlabeled data.

4.5.2.1.Progressive Confidence Threshold

To evaluate the quality of pseudo-labeled data, the confidence scores produced by the box-head are used as an indicator. The confidence score of a proposal is compared to its ground-truth label to determine its classification accuracy. The objective is to minimize the classification loss (L_{cls}) between the predicted confidence score (p_i) and the ground-truth label (p_i^*) for each proposal.

$$C_i^{FRCNN} = \underset{p_i}{argmin}\, L_{cls}(p_i, p_i^*), \qquad \textit{Equation 4.4}$$

$\boldsymbol{p_i^*}$ is the ground-truth label.

Therefore, the confidence score produced by the box-head in Faster R-CNN serves as a fundamental indicator to distinguish between true positive and false positive samples, aiding in the evaluation of the quality of pseudo-labeled data.

4.5.2.2.Non- Maximum Suppression (NMS)

Non-Maximum Suppression (NMS) is an important step in object detection algorithms to refine the bounding box predictions and eliminate redundant detections. It works by selecting the most confident bounding box prediction for each object class and removing overlapping predictions. However, NMS alone is not sufficient to remove predicted bounding boxes that are located in incorrect places. To improve the quality of the pseudo-labels generated by the model, confidence thresholding is applied to the NMS outputs.

NMS works as follows:

1. Input: Given a set of bounding box detections from an object detection algorithm, each associated with a confidence score.
2. Sorting: Sort the detections based on their confidence scores in descending order.
3. Selecting the Most Confident Detection: Start with the highest-scoring detection and consider it as a kept detection.

4. Overlap Calculation: Compare the overlap (e.g., Intersection over Union - IoU) between the current kept detection and the remaining detections.
5. Suppression: Remove any detection with an overlap above a predefined threshold (e.g., IoU threshold) with the current kept detection.
6. Iteration: Move to the next highest-scoring detection and repeat steps 4-5 until all detections have been processed.
7. Output: The final set of kept detections after NMS.

4.6. Domain Adaptive Faster RCNN Network

The domain adaptive Faster R-CNN network shares several components with the original Faster R-CNN network, including the feature extractor, region proposal network (RPN), and region of interest (RoI) pooling. However, it also integrates domain adaptation components to address the domain shift problem and adapt to different domains. A crucial component in the domain adaptive Faster R-CNN network is the gradient reversal layer, which aids in aligning feature representations across the source and target domains. By reversing the gradients through this layer during training, the network is encouraged to learn domain-invariant features. The image-level domain classifier is responsible for classifying the domain of an entire image, while the instance-level domain classifier classifies the domain of individual object instances within an image. These domain classifiers help in distinguishing the domains and adapting the network accordingly. By incorporating these domain adaptation components, the domain adaptive Faster R-CNN network can effectively adapt to different domains, reduce the impact of domain shift, and improve the overall performance of object detection tasks in various domains.

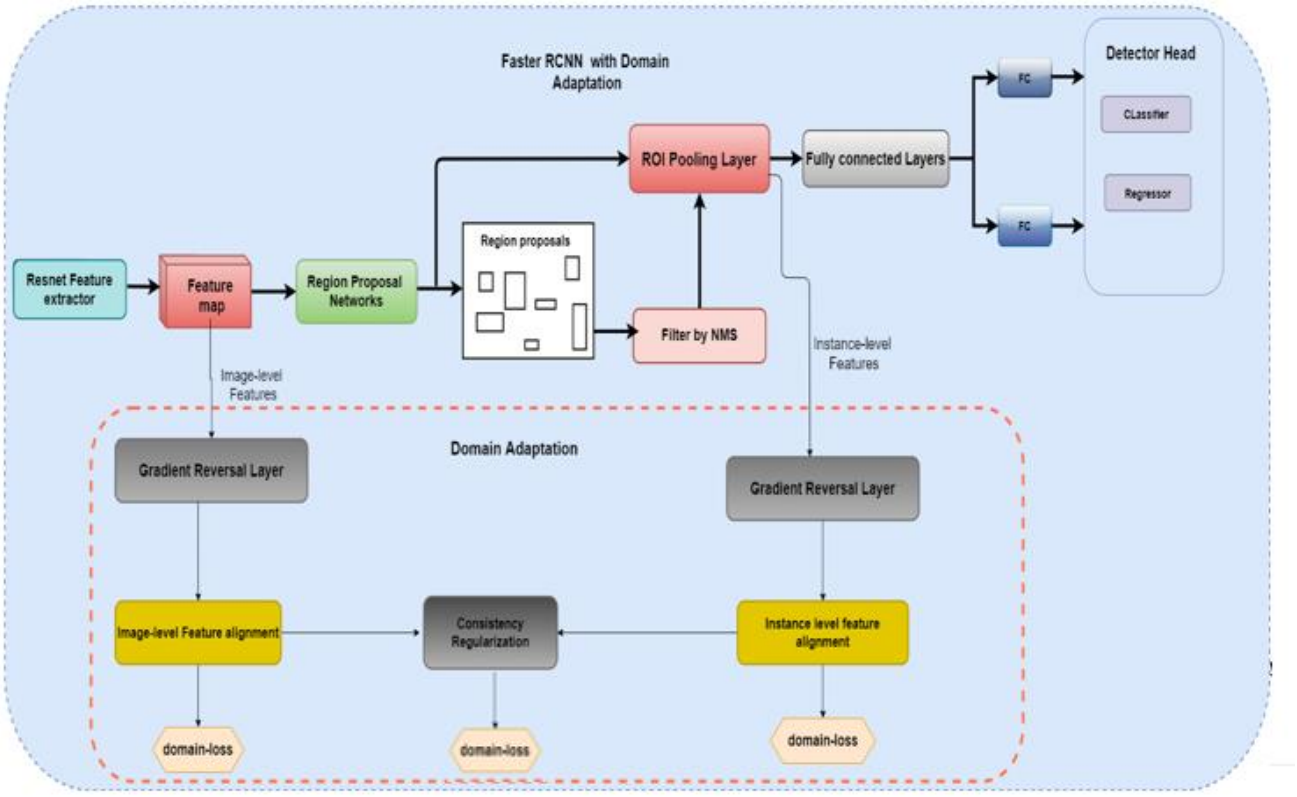


Figure 4.8:Architectures of the second Faster RCNN model with domain adaptation components

4.6.1. Gradient Reversal Layer

The Gradient Reversal Layer (GRL) is a technique utilized in domain adaptation to promote the learning of domain-invariant feature representations. It functions by preserving the input data during forward propagation and reversing the gradients during backpropagation. This reversal of gradients is achieved by multiplying them with a negative scalar, which helps align the feature distributions across different domains. Mathematically, the GRL can be represented as a "pseudo function" with two equations describing its forward and backward propagation:

$$\mathcal{R}(\boldsymbol{x}) = \boldsymbol{x}, \ \ and \ \ \frac{d\mathcal{R}}{d\boldsymbol{x}} = -\boldsymbol{I} \qquad \textit{Equation 4.5}$$

By incorporating the GRL into the network architecture, the model can learn features that are insensitive to domain-specific variations and biased information. This enables the network to better generalize to unseen domains and improves its adaptability to different data distributions. The GRL plays a crucial role in mitigating the effects of domain shift and enhancing the robustness of the model during domain adaptation tasks.

4.6.2. Image and Instance level Feature Alignment

To address the domain distribution mismatch in the Faster R-CNN model, a patch- based domain classifier is utilized. This classifier is trained on activations from the feature map and assigns a domain label to each image patch. The objective is to align image-level representations and mitigate biases caused by variations across images, such as style, size, and lighting. The parameters of the domain classifier are optimized concurrently with the base network using gradient reversal.

The benefits of this are:

1. Alignment of image-level representations: By aligning image-level representations, biases stemming from differences in image characteristics, such as style, size, and lighting, can be reduced. This is particularly important as it helps to make the model more robust and adaptable to different domains. Similar patch-based loss techniques have been successful in various tasks, including type transfer, where they handle global transformations.
2. Handling small batch sizes: Object detection networks often use small batch sizes due to the utilization of high-resolution images. By incorporating the patch- based domain classifier, the network can effectively address the domain mismatch even with limited training data.

The image-level fitting loss is calculated using the cross-entropy loss function and can be expressed as follows:

$$\mathcal{L}_{img} = -\sum_{i,u,v} \left[D_i \log p_i^{(u,v)} + (1 - D_i) \log\left(1 - p_i^{(u,v)}\right) \right] \qquad \textit{Equation 4.6}$$

During training, the parameters of the domain classifier are minimized to minimize the domain classifier loss described above, while simultaneously optimizing the loss of the base network. To achieve this, gradient reversal layers (GRL) are employed. When optimizing the base network through the GRL layer, the gradient signs are reversed, facilitating the alignment of domain distributions and encouraging domain- invariant feature learning.

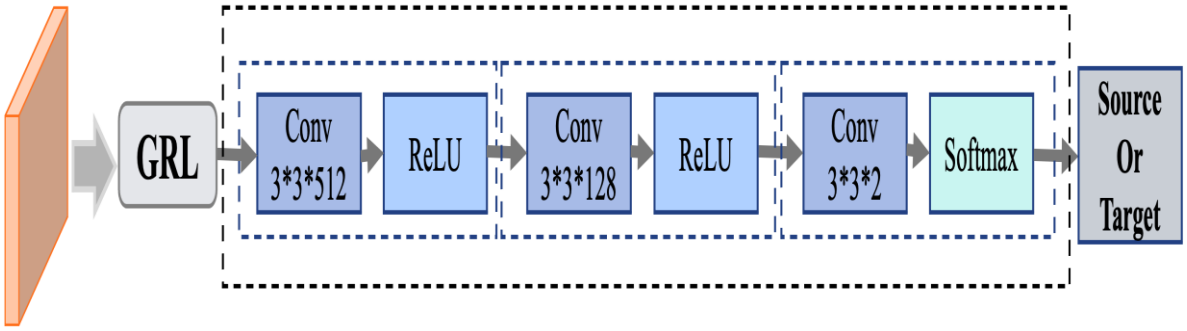


Figure 4.9:The network structure of image-level domain classifier

4.6.3. Consistency Regularization

Consistency regularization is a regularization approach used to make the two networks to provide reliable domain invariant predictions. Consistency Regularization of is essential when I use unlabeled target data to make sure that the classifier produces probability distributions that are comparable to those of the Smooth classifier., when subjected to various transformations. This is accomplished by punishing the network for prediction deviations from the Smooth network using a consistency regularization term. During the initial stages of the training process, a consistency loss is introduced to promote consistency between the predictions of the smoother network and the faster network.

To improve consistency between the domain classifiers at different levels, the activation outputs of the image-level domain classifier are averaged across all activations within an image, yielding the image-level probability. Continuous regularization is employed as a technique to achieve this consistency. continuous regularization loss is formulated as follows:

$$L_{cst} = \sum_{i,j} \left\| \frac{1}{|I|} \sum_{u,v} p_i^{(u,v)} - p_{i,j} \right\|_2 \qquad \text{Equation 4.7}$$

In this equation, I represent the total number of activations in the feature map. The term p_i^((u,v)) denotes the activation output of the image-level domain classifier at position (u,v) in the feature map. The term p_(i,j) represents the image-level probability of class i predicted by the domain classifier for the jth image. The l2 distance (‖.‖_2) is used to measure the discrepancy between the averaged activation outputs and the image-level probabilities. By applying continuous regularization with the consistency loss, the training process encourages consistency in the predictions across different levels of the domain classifiers, promoting cross-domain robustness in the bounding box predictors.

4.7. Loss functions

4.7.1. Classification Loss

The classification loss in the proposed object detection algorithm is concerned with the confidence scores assigned to the predicted classes. This loss is computed for each positive match prediction and is calculated as the SoftMax cross-entropy loss over all c classes.

$$L_{conf}(x,c) = -\sum_{i\in P_{os}}^{N} x_{ij}^{p} \, log\left(\hat{c}_i^p\right) - \sum_{i\in N_{eg}} log\left(\hat{c}_i^0\right) \; where \; \hat{c}_i^p = \frac{exp(c_i^p)}{\sum_p \, exp(c_i^p)} \qquad \textit{Equation 4.8}$$

Where N is the number of matched default boxes. Eventually, the total loss is computed as:

$$L(x,c,l,g) = \frac{1}{N}\left(L_{conf}(x,c) + \alpha L_{loc}(x,l,g)\right) \qquad \textit{Equation 4.9}$$

Here, L_conf(x, c) is the classification loss, L_loc(x, l, g) is the localization loss, and α is the weight assigned to the localization loss.

To tackle the imbalance issue between positive and negative matches in training, a strategy is implemented to address this disparity. Negative matches are sorted based on their confidence loss, and only a limited number of top negatives are chosen. This approach ensures that the ratio between negatives and positives remains within a 3:1 ratio. By adopting this strategy, the training process becomes more efficient and reliable.

4.7.2. Domain Loss

During the implementation of the proposed algorithm, a domain loss is incorporated to measure the difference between source and target domain images, effectively capturing the domain shift. The objective during training is to minimize this domain loss, thereby minimizing the domain shift between the two domains.

The domain loss L_{don} is calculated as.

$$L_{dom} = \frac{1}{B}\sum_{i=1}^{B} \sum_{j=1}^{M_1} \left|G_j\left(F_{S_i}^1\right) - G_j(F_T^1)\right|, \qquad \textit{Equation 4.10}$$

The given equation combines the global average pooling operation, represented by G_j(·), on the jth feature map, element-wise absolute value denoted as |.|, the number of feature maps in

the Pool1 layer denoted by M1, the batch size represented by B, and the total loss L is calculated by combining the detection loss L_det and the domain loss L_dom with a coefficient α.

$$L = L_{det} + \alpha L_{dom} \qquad \textit{Equation 4.11}$$

Where L_{det} is the detection loss that measures the difference between the ground truth and detection predictions; α is a coefficient.

4.8. Training Pseudocode

Dataset D_t represents the unlabeled target domain dataset. $G(\cdot)$ represents data augmentation. θ_t represents the weight parameter, initialize the trade-off parameters α, weight smoothing parameter λ. Pre-train parameter $\Re$. S_{model} , T_{model} represent the source model and target model respectively. $\{Y_{i,k}^{T}\}_{i=1}^{N_t}$ represents the pseudo-labels generated by the faster RCNN at the k th iteration. $P_k(x_i^T)$ represents the confidence threshold.

Algorithm 1: Overall training algorithm of the Domain adaptive object detection model

Input*: images of SD (Source Domain), and TD (Target Domain), labels of SD*

Require: Source Dataset: $\mathcal{D}_s = \left\{\left(x_s^{(i)}, y_s^{(i)}\right)\right\}_{i=1}^{N_s}$*; Target Dataset:* $\mathcal{D}_t = \left\{x_t^{(i)}\right\}_{i=1}^{N_t}$

Output*: best domain adaptive model for object detection*

1. *For epochs in range (0, total_epochs):*
2. *For images in range (0, len(train_set),one_batch):*
3. *Load images of SD, TD and labels of SD1, SD2*
4. *Generate samples via* $x_i^T, x_i^S \leftarrow G(x_i)$ *randomly*
5. *Use* x_i^S *as the input of* S_{model} *, and* x_i^T *as input of* T_{model}
6. *Generate* R_x^T *with* T_{model} $'sRPN$ *and shared with* S_{model}
7. *Update* S_{model} *with Eq 4.11*
8. *Update* T_{model} *from* S_{model} *with Eq 4.10*
9. *if starting adaptation then*
10. *Train* S_{model} *with pseudo-labels* $\{Y_{i,k}^T\}_{i=1}^{N_t}$

11. *Update* S_{model} *via* $\theta_s \leftarrow Adam\left(\nabla_{\theta_s}\left(\alpha L_{dist} + \beta(L_{Re} + L_{Et} + L_{In})\right), \theta_s, \alpha, \beta\right.$

12. *Update* T_{model} *via* $\theta_t = \lambda\theta_{t-1} + (1-\lambda)\theta_s, \lambda$

13. *end if*

14. *end for*

15. *Generate pseudo-labels* $\{Y_{i,k}^T\}_{i=1}^{N_t}$ *for the next iteration*

16. *end for*

17. *Get trained model*

18. *Validate My_Model on validation_set*

19. *Compare Best_Mdel and My_Model*

20. *Model_Testing*

21. *Best_model is tested on the test_set of TD*

CHAPTER FIVE

IMPLEMENTATION OF THE PROPOSED SOLUTION

5.1. Chapter Overview

In this chapter, the implementation of the proposed domain adaptive object detection algorithm is presented, along with details of the working environment. The chapter offers an overview of the implementation process, focusing on the crucial aspects of the developed code and providing a thorough discussion of these components.

The code implementation is described step-by-step, illustrating how the proposed solutions were transformed into executable code. The chapter delves into the specific details of the algorithm, highlighting the key parts of the code and explaining their functionality in depth. By providing a comprehensive walkthrough of the implementation process, the chapter aims to give readers a clear understanding of how the proposed domain adaptive object detection algorithm was translated into practical code. The explanations and discussions ensure that the implementation details are thoroughly explained, enabling readers to replicate or modify the code for their own purposes.

5.2. Working Environment

In the process of implementing object detection experiments, different hardware consisting of a laptop and a desktop computer was used.

- The laptop, operating on Windows 10, served as the platform for developing the network architecture. It was equipped with an Intel Core i5-7200U CPU running at a base frequency of 2.50GHz (with a turbo boost up to 2.70GHz), Intel Graphics 3000 for graphical processing, and 8 GB of RAM.
- The desktop computer, also operating on Windows 10, was used for various other activities related to the experiments. It featured an Intel Core i5-4580 CPU with a clock speed of 3.29GHz (4 cores), Intel HD Graphics 4600 for graphical processing, and an NVIDIA GeForce RTX 2070 Super GPU with 12 GB of RAM for accelerated computations. The desktop computer had 8.00 GB of primary memory (RAM) and ran

on a 64-bit operating system with an x64-based processor.

- For development purposes, Visual Studio Code and Jupyter Notebook were utilized as integrated development environments (IDEs), with Python 3.9 as the programming language. PyTorch 1.12 and TensorFlow 2.8.0 were the deep learning frameworks used for building and training the models, while OpenCV was employed for object detection and video processing.

  The combination of these devices, along with the specified software tools and libraries, facilitated the implementation and experimentation of object detection techniques effectively.

5.3.Environmental Setup

Additionally, the programming language and libraries employed consist of Python, Anaconda, PyTorch, TensorFlow, and OpenCV. The integrated development environments (IDEs) used are Jupyter Notebook and Visual Studio Code, both of which serve various purposes in code development, debugging, and version control.

5.4.Implementation for the Proposed Domain Adaptive Object detection

5.4.1. Data Augmentation Implementation

Data augmentation is applied to create new and realistic-looking training data by applying a change to the image without affecting its label. It is a crucial component in projects involving deep learning. It is well recognized for its value in preventing overfitting and improving the generalization of models. Adding slightly altered copies of previously existing images to the dataset may also artificially increase its size.

The trorchvision library from pytorch is used for applying augmentation which contains 30 different augmentations available in the torchvision. Transforms module. The following are used as a Strong augmentation from the torchvision library.

- **ColorJitter**: Randomly Changes saturation, contrast, and brightness at random. Using this way, it's possible to simulate day and night conditions which helps for generalization across domains.
- **RandomGrayScale**: Convert a picture to grayscale at random with a probability of p (default 0.1).
- **GaussianBlur**: Blurs a picture using a Gaussian blur that is selected at random by using both PIL and tensor images, or a batch of tensor images
- **RandomErasing:** applies random transformation by selecting a rectangular region in an input image and erases its pixels. To execute various sorts of manipulations on the picture data, the torchvision. Transforms module offers a number of significant transformations.
- **RandomCrop**: Crop the given image at a random point.

```
augmentation = []
        augmentation.append(
            transforms.RandomApply([transforms.ColorJitter(0.4, 0.4, 0.4, 0.1)], p=0.8)
        )
        augmentation.append(transforms.RandomGrayscale(p=0.2))
        augmentation.append(transforms.RandomApply([GaussianBlur([0.1, 2.0])], p=0.5))

        randcrop_transform = transforms.Compose(
            [
                transforms.ToTensor(),
                transforms.RandomErasing(
                    p=0.7, scale=(0.05, 0.2), ratio=(0.3, 3.3), value="random"
                ),
                transforms.RandomErasing(
                    p=0.5, scale=(0.02, 0.2), ratio=(0.1, 6), value="random"
                ),
                transforms.RandomErasing(
                    p=0.3, scale=(0.02, 0.2), ratio=(0.05, 8), value="random"
                ),
                transforms.ToPILImage(),
            ]
        )
        augmentation.append(randcrop_transform)
```

Figure 5.1: Code snippet For Data augmentation implementation

5.4.2. Faster RCNN Module Implementation

The Faster RCNN implementation module is mainly divided into four modules:

- **The Backbone Feature extraction**. The feature extraction process employs a ResNet network to extract feature maps from images. This is achieved through a sequence of convolutional, activation, and pooling layers.
- **RPN (Region Proposal Network).** That is, the regional candidate network, which replaces the Selective Search of the previous RCNN version and is used to generate candidate boxes.
- **RoI Pooling**. The RoI Pooling technique is then applied to collect proposals generated by the RPN (Region Proposal Network) and extract them from the feature maps.
- Classification and Regression. The proposal feature maps are utilized to calculate specific categories, and subsequent bounding box regression is performed to acquire the final precise position of the detection frame.

```
class GeneralizedRCNN(nn.Module):
    """
    Generalized R-CNN. Any models that contains the following three components:
    1. Per-image feature extraction (aka backbone)
    2. Region proposal generation
    3. Per-region feature extraction and prediction
    """
    @configurable
    def __init__(
        self,
        *,
        backbone: Backbone,
        proposal_generator: nn.Module,
        roi_heads: nn.Module,
        pixel_mean: Tuple[float],
        pixel_std: Tuple[float],
        input_format: Optional[str] = None,
        vis_period: int = 0,
    ):
```

Figure 5.2: Code Snippet for Faster RCNN module Implementation

5.4.3. Implementation for Creating Faster RCNN Models

The faster way to use Faster-RCNN is using detectron2. In detectron2, users have the flexibility to implement custom models that can support any input format. For example, you can create the first and second models using the Faster R-CNN meta-architecture. The standard input format detectron2 models format list [dict] which contains architecture.

The build_model(cfg) function is responsible for constructing the architecture of the entire model based on the specified configuration (cfg). However, it does not load any pre-trained weights from the configuration. Instead, it initializes the model architecture according to the specified meta-architecture defined in fg.MODEL.META_ARCHITECTURE. Once the model is built, it can be used by invoking it with input data. The input data should be provided as a list of dictionaries, where each dictionary corresponds to an image and contains the required keys depending on the specific model and mode (training or evaluation) being used.

Once the models are trained, DefaultPredictor instances, source_predictor and target_predictor, are created using their respective configurations. These instances will be used for inference on source and target images, respectively.

```
# Create and train the source Faster R-CNN model
source_model = build_model(source_cfg)
source_trainer = DefaultTrainer(source_cfg)
source_trainer.resume_or_load(resume=False)
source_trainer.train()

# Create and train the target Faster R-CNN model
target_model = build_model(target_cfg)
target_trainer = DefaultTrainer(target_cfg)
target_trainer.resume_or_load(resume=False)
target_trainer.train()
# Use the trained source model for inference
source_predictor = DefaultPredictor(source_cfg)
# Use the trained target model for inference
target_predictor = DefaultPredictor(target_cfg)
```

Figure 5.3: Code snippet creating Faster RCNN models

5.4.4. Training Faster RCNN Model

The Faster R-CNN model is trained to recognize and localize objects in the source domain by minimizing classification and regression losses. The goal of training on labeled source data is to develop a model that can generalize well and detect objects in the unlabeled target domain (Xt) without access to ground-truth labels. By aligning features learned from the labeled source domain with the unlabeled target domain, the model aims to generalize its object detection capabilities to the target domain. This adaptation process allows the model to effectively detect objects in the unlabeled target domain, utilizing knowledge learned from the labeled source dataset.

```
# Move the model to the device
model.to(device)
# Define the optimizer
optimizer = torch.optim.SGD(model.parameters(), lr=0.005, momentum=0.9, weight_decay=0.0005)
# Training loop
num_epochs = 10
for epoch in range(num_epochs):
    model.train()
    for images, targets in data_loader:
        # Move images and targets to the device
        images = list(image.to(device) for image in images)
        targets = [{k: v.to(device) for k, v in t.items()} for t in targets]
        # Zero the gradients
        optimizer.zero_grad()
        # Forward pass
        loss_dict = model(images, targets)
        losses = sum(loss for loss in loss_dict.values())
        # Backward pass
        losses.backward()
        optimizer.step()
    # Print the epoch and loss
    print(f"Epoch: {epoch+1}/{num_epochs}, Loss: {losses.item()}")
# Save the trained model
torch.save(model.state_dict(), "model.pth")
```

Figure 5.4: Code Snippet for Training Faster RCNN

5.4.5. Implementation of Applyinng Inference on Unlabeled Images

During the inference phase, the target domain images are passed through the Faster RCNN network. The network generates predictions for object categories and bounding box coordinates. These predictions are used as pseudo labels for the target domain. In the first stage, anchor boxes with high classification scores are selected based on the network's confidence in predicting object presence. These anchor boxes are further processed in the second stage to refine the predicted categories and offsets (adjustments to the bounding box coordinates). This

refinement improves the accuracy of the predicted bounding boxes.

```
result_per_image = [
        fast_rcnn_inference_single_image(
            boxes_per_image, scores_per_image, image_shape, score_thresh, nms_thresh,
 topk_per_image, cls_logist,
            sigma_logit
        )
```

Figure 5.5: Code Snippet for applying Inference on images for pseudo label Generation

5.3.1. Implementation for Generating Pseudo-labels for Unlabeled Data

After generating the pseudo labels, a selection criterion is implemented to only utilize pseudo labeled data that the model is more confident about. This confidence is determined by measuring the SoftMax output prediction entropy. The second model can be trained in a supervised manner using this pseudo labeled target data. Here's a Python code that uses a trained Faster R-CNN model to generate pseudo labels for unlabeled data:

```
# Define the data loader for the unlabeled data
unlabeled_data_loader = DataLoader(unlabeled_dataset, batch_size=2, shuffle=False)
# Load the pre-trained Faster R-CNN model
model = torchvision.models.detection.fasterrcnn_resnet50_fpn(pretrained=True)
model.to(device)
model.eval()
# Generate pseudo labels for unlabeled data
pseudo_labels = []
with torch.no_grad():
    for images in unlabeled_data_loader:
        images = list(image.to(device) for image in images)
        # Forward pass
        outputs = model(images)
        # Extract class predictions and bounding box coordinates
        for output in outputs:
            labels = output['labels']
            boxes = output['boxes']
            # Create pseudo label dictionary
            pseudo_label = {
                'labels': labels.cpu(),
                'boxes': boxes.cpu()
            }
            pseudo_labels.append(pseudo_label)
```

Figure 5.6: Code snippet for adding label to unlabeled images implementation

5.1.4. Implementation for Filtering Pseudo-labels using Confidence thresholding and Non-maximum Suppression

For each unlabeled sample, the model used to choose the pseudo-labels predicts the class with the highest prediction probability: Pseudo-labels may be employed at the dropout fine-tuning step. Pretrained networks are trained in a supervised way utilizing both labeled and unlabeled data.

```
filtered_pseudo_labels = []
# Apply progressive confidence thresholding and NMS
current_threshold = initial_threshold
while current_threshold >= 0:
    # Check if there are any remaining indices to process
    if len(sorted_indices) == 0:
        break
    # Get the highest confidence pseudo label index
    highest_confidence_index = sorted_indices[0]
    # Get the corresponding pseudo label
    highest_confidence_pseudo_label = pseudo_labels[highest_confidence_index]
    # Add the pseudo label to the filtered list
    filtered_pseudo_labels.append(highest_confidence_pseudo_label)
    # Get the remaining indices after NMS
    remaining_indices = sorted_indices[1:]
    # Calculate IoU between the highest confidence pseudo label and remaining
    ious = box_iou(highest_confidence_pseudo_label['boxes'].unsqueeze(0),
                   bounding_boxes[remaining_indices])
    # Find the indices of pseudo labels that have IoU less than the threshold
    mask = ious.squeeze(0) < iou_threshold
    remaining_indices = remaining_indices[mask]
    # Update the sorted indices with the remaining indices
    sorted_indices = remaining_indices
    # Decrease the threshold for the next iteration
    current_threshold -= threshold_step
```

Figure 5.10: Code snippet for processing pseudo label using thresholding and NMS

5.1.5. Gradient reversal layer (GRL) Implementation

The proposed domain adaptive object detection algorithm incorporates adversarial learning to extract domain invariant features. The Gradient Reversal Layer (GRL) is inserted between the detection network (G) and each domain classifier (Cimg for image-level and Cins for instance-level). The GRL reverses gradients during back-propagation, causing the detection network to update parameters in the opposite direction, fooling domain classifiers into perceiving features as domain-invariant. This fosters the creation of robust features resilient to domain shifts.

The Gradient Reversal Layer is a wrapper module that takes an optional alpha parameter and serves as an interface for using the Gradient Reversal Function. It creates an instance of the Gradient Reversal Function during the forward pass, passing the input tensor x and the alpha value. The forward method of the Gradient Reversal Layer simply calls the forward method of the Gradient Reversal Function to apply the gradient reversal transformation

```
################ Gradient reverse function###############
class GradReverse(torch.autograd.Function):
    @staticmethod
    def forward(ctx, x):
        return x.view_as(x)

    @staticmethod
    def backward(ctx, grad_output):
        return grad_output.neg()

def grad_reverse(x):
    return GradReverse.apply(x)
```

Figure 5.11: Code snippet for the gradient reversal layer implementation

Instance-level feature alignment and image-level feature alignment are two different proaches used in domain adaptation to align features between the source and target domains. Performing image-level and instance-level feature alignment. During the training loop, the forward pass is performed on both the source and target domains. The image-level alignment is achieved by calculating the MMD loss between the source and target features. The instance-level alignment is achieved using the domain discriminator (typically a binary classifier) to predict the domain labels (source or target) based on the features. The domain loss is then calculated using the predicted domain labels and the ground truth domain labels. The total loss is the sum of the classification loss, MMD loss, and adversarial loss. The backward pass and parameter update are performed as before.

In image-level adaptation, the classification entropy of area suggestions is used to weight uncertainty-aware adversarial loss when the image-level features are fed into the gradient

reversal layer (GRL). During instance-level adaptation, the instance-level domain classifiers (Cins) receive instance-level features from the Gradient Reversal Layer (GRL). These features are specifically tailored to each instance and are used as input for the instance-level domain adaptation process. The instance-level domain classifier Cins is then optimized using a curriculum loss reduction strategy guided by instance-level uncertainty. The curriculum loss is weighted by the filtered detection entropy, which helps prioritize instances with higher uncertainty and focus the adaptation process on challenging samples. This approach aims to improve the robustness and adaptability of the model at the instance level, enhancing its ability to handle domain shifts and accurately detect objects in different domains.

```
            features_s = grad_reverse(features[self.dis_type])
            D_img_out_s = self.D_img(features_s)
            loss_D_img_s = F.binary_cross_entropy_with_logits(D_img_out_s, torch.FloatTensor(D_img_out_s.data.
size()).fill_(source_label).to(self.device))

            features_t = self.backbone(images_t.tensor)

            features_t = grad_reverse(features_t[self.dis_type])
            D_img_out_t = self.D_img(features_t)
            loss_D_img_t = F.binary_cross_entropy_with_logits(D_img_out_t, torch.FloatTensor(D_img_out_t.data.
size()).fill_(target_label).to(self.device))
```

Figure 5.12: Code Snippet for Domain classifier Implementation

Training Faster R-CNN for domain adaptation using labeled source domain data and pseudo-labeled target domain data involves a combination of standard Faster R-CNN training with additional steps to incorporate the target domain data. Here's an example of Python code for training Faster R-CNN with domain adaptation.

5.1.4. Implementation for Retraining Faster RCNN using Labeled source and Pseudo labeled target data

Training Faster R-CNN for domain adaptation using labeled source domain data and pseudo-labeled target domain data involves a combination of standard Faster R-CNN training with

additional steps to incorporate the target domain data. Here's an example of Python code for training Faster R-CNN with domain adaptation:

```
num_iterations = 1000
# Training loop
for iteration in range(num_iterations):
    # Get a batch of labeled source domain data
    source_images, source_targets = next(iter(source_dataloader))
    # Get a batch of unlabeled target domain data
    target_images, _ = next(iter(target_dataloader))
    # Create pseudo-labels for the target domain data
    with torch.no_grad():
        target_predictions = model(target_images)
        target_labels = [torch.argmax(pred["scores"]) for pred in target_predictions]
    # Combine source and target data and labels
    images = torch.cat([source_images, target_images], dim=0)
    targets = torch.cat([source_targets, target_labels], dim=0)
    # Forward pass and compute loss
    model.train()
    loss_dict = model(images, targets)
    total_loss = sum(loss for loss in loss_dict.values())
```

Figure 5.12: Code Snippet for retraining faster RCNN using labeled source domain and pseudo-labeled target domain

5.2. Hyper parameter Configuration

The training process involves optimizing the performance of the object detection network by tuning various parameters. These parameters include the learning rate, batch size, number of epochs, optimization algorithm, learning rate schedule, and loss weights.

The network is trained using a two-step learning rate schedule. Initially, the entire network is trained for 50,000 iterations with an initial learning rate of 0.0001. After that, the learning rate is reduced to 0.00001 for an additional 20,000 iterations. This schedule aids in achieving convergence to a good solution.

To update the network weights, backpropagation and stochastic gradient descent (SGD) are employed as optimization techniques. A momentum of 0.9 and weight decay of 0.0005 are utilized for regularization purposes, preventing overfitting. During training, each batch comprises images from both the source domain and the target domain. This mixed batch training strategy helps the model learn features that are invariant to the domain and adapt effectively to the target domain. The training is performed on a GeForce RTX3090 GPU

card with 12GB RAM, which provides computational power for efficient training and inference. The algorithm is implemented using PyTorch, a popular deep learning framework, in conjunction with the Detectron Toolbox, which provides pre-implemented modules and utilities for object detection tasks.

By carefully tuning these parameters and using powerful hardware for training, the algorithm aims to optimize the performance and achieve accurate and reliable results. During the pseudo label generation, clear background samples are filtered using a score threshold. In order to avoid noisy pseudo label samples in this study, whose behavior is uncertain and scarcely supplies label for unlabeled target domain, a score threshold is established to a reasonably high value. Empirically, the Non-Maximal Suppression (NMS) is set at 0.45 in the NMS of the R-CNN stage and 0.9 in the NMS of the RPN stage. to steady the instruction. Throughout whole training phase, the data augmentations are applied. The target image's threshold for obtaining fictitious annotations is set at 0.7. Random color jittering, gray scaling, Gaussian blurring, and chopping off regions are employed for strong augmentations whereas random horizontal flipping is used for mild augmentation. Five different classes of tests were carried out to demonstrate performance enhancement of object identification across various domains, and two baseline models were created, as shown in table 5.1.

5.3. Experimental Class

The experimental configuration for the object detection framework is based on Faster R-CNN with ResNet101 as the backbone network. The source domain data is only used during the pre-training phase of the model. The experiments were conducted using NVIDIA GeForce 1080 graphics cards with 12GB of RAM, Driver version 520.61.05, and CUDA version 11.8.

To evaluate the performance improvement of object detection across different domains, six classes of experiments were performed. Two baseline models were implemented for comparison. In the first experiment, the Cross-Domain Adaptive Teacher for Object Detection (CDATOD) baseline approach was employed. This approach involved a target-domain Teacher model and a cross-domain second model that mutually learn from each other. Augmentation procedures and adversarial learning techniques were applied in the

training pipeline to address domain bias in both models.

In the second experiment, Cross-Domain Adaptive Teacher for Object Detection (CDATOD) approach with confidence thresholding and class-wise non-maximal suppression (NMS) was proposed. These techniques helped filter out noisy pseudo labels, improving the quality of the annotations. The third experiment combined the Cross-Domain Adaptive Teacher for Object Detection (CDATOD) approach with a Gradient Reversal Layer (GRL) and domain loss to align the distributions of the two domains. This alignment aimed to reduce the domain shift and enhance the adaptability of the model. In the fourth experiment, weak and strong data augmentation techniques were introduced to both models, along with the domain loss. This modification aimed to reduce false positives in the pseudo labels generated by the first model. In the fifth experiment, the proposed confidence thresholding approach and domain loss were integrated into the second model, with GRL embedded to extract domain- invariant features. Strong and weak data augmentation techniques were applied to both models.

Table 5.1: List of Experimental Class

Notation	Experiment	Dataset Used
Cross-Domain Adaptive Teacher for Object Detection (CDATOD) Y.-J. Li et al., 2022)	Baseline work	Cityscape, Foggy Cityscape, Pascal VOC, WaterColor2k, Clipart1k
CDATOD +T	Experiment by Applying progressive confidence thresholding pseudo labelling on the first model	Cityscape, Foggy Cityscape, Pascal VOC, WaterColor2k, Clipart1k
CDATOD +NMS	(Cross-Domain Adaptive Teacher for Object Detection (CDATOD) with non-maximal suppression)	Cityscape, Foggy Cityscape, Pascal VOC, WaterColor2k, Clipart1k
CDATOD +Aug+CR	Experiment with Addition of target-like data augmentation and consistency regularization	Cityscape, Foggy Cityscape, Pascal VOC, WaterColor2k, Clipart1k
CDATOD +Aug+T+CR	Experiment With addition of data augmentation, confidence Thresholding and consistency regularization	Cityscape, Foggy Cityscape, Pascal VOC, WaterColor2k, Clipart1k
CDATOD+Aug+CR+T+ NMS	Experiment With addition of data augmentation, On-maximum suppression Thresholding and consistency regularization	Cityscape, Foggy Cityscape, Pascal VOC, WaterColor2k, Clipart1k

CHAPTER SIX

RESULTS AND DISCUSSIONS

6.1. Chapter Overview

In this chapter, the thesis discusses the experimental findings, presenting both qualitative and quantitative results. The experiments aim to tackle domain discrepancy challenges such as weather variations, camera differences, and adaptation to large-scale datasets. The effectiveness of the proposed approach is demonstrated through various experiments conducted under different scenarios.

The first scenario involves adapting from clear weather conditions to foggy weather, where the proposed method is evaluated. The second scenario focuses on adapting from synthetic images to real-world images. In this case, a simulation engine is utilized to generate source domain images, while target domain images are collected from the real world. This scenario offers the advantage of avoiding the manual collection and annotation of real-world images. To assess the performance of the proposed method, benchmark models are considered.

The reported results emphasize the effectiveness of the proposed method in tackling domain discrepancy challenges and enhancing object detection performance across various adaptation scenarios. By combining qualitative and quantitative analysis, a comprehensive understanding of the proposed approach's capabilities is presented, showcasing its superiority over the baseline models.

6.1. Experimental Results

Experiment is conducted in different domain adaptation scenarios to evaluate the effectiveness of the proposed approach. These domain adaptation scenarios are:

- Clear to foggy weather adaptation (Cityscape →Foggy Cityscape)
- Synthetic to real adaptation (SIM10K→ Cityscape),
- Cross-camera adaptation (KITTI →Cityscape),
- Cross-Scene Adaptation (Daytime→Nighttime)
- Real to Artistic Adaptation (Pascal VOC→Clipart1k, Pascal VOC→Watercolor2k)

6.1.1. Cross Weather Adaptation Experiment (Cityscape→Foggy Cityscape)

Weather conditions can significantly impact visual data, and accurate object detection is crucial in applications like autonomous systems. However, collecting data encompassing various weather conditions for training object detection models can be expensive. Through the utilization of semi-supervised learning and self-training techniques, the proposed method surpasses the current state-of-the-art approaches in terms of performance. It achieves improved classification boundaries and demonstrates superior quantitative performance compared to the baseline method. The adaptation process enables the detector to effectively handle the challenges presented by foggy weather conditions, enhancing its ability to accurately detect and classify objects. Object detection results on the source domain dataset is shown on figure 6.1 which is used as a ground truth for the foggy cityscape adaptation results.

Firstly, the performance of the baseline Cross-Domain Adaptive Teacher for Object Detection (CDATOD) is evaluated. As shown Figure the Adaptation result of the baselisne shows low performance for cityscape→foggy cityscape adaptation due to low quality pseudo labels and not applying consistency regularization.Even though there is accuracy gain for some objects some of the objects are completely missied on the detection result. On the second experiment Confidence thresholding and non-maximal suppression (NMS) are applied. At the beginning of the training, the coverage of the pseudo-labels is relatively low due to the strict confidence thresholding mechanism. however, as semi-supervised learning progresses, the performance of the object detection increases and so as to the confidence thresholding, which in turn increases the object coverage of the pseudo-labels. As a result, more objects are detected as a result of applying Adaptive confidence thresholding and non-maximal suppression on the first model. Domain loss was first added in the fourth experiment in order to further examine the significance of adversarial learning in the suggested strategy. It is observed that, first, raising weights results in better performance, supporting the second model’s domain classifier's potency. The models were degrading because of the inaccuracy without using the adversarial loss. And in the fourth experiment it is shown that domain loss has a significant performance effect.

The impact of a target-like data augmentation strategy is assessed in the fifth and sixth experiments of the proposed approach. The results demonstrate that making slight adjustments to the training pipeline, such as using weak and strong augmentations both models, respectively, is crucial.

Figure 6.1: Qualitative Results for cityscape→foggy cityscape adaptation on each Experiments

6.1.1. Cross Scene Adaptation (Daytime to Nighttime Adaptation Experiment)

In practical scenarios where object detection is applied, such as in autonomous driving systems operating in various cities, the scenes encountered can differ in their layout and characteristics. To evaluate the effectiveness of an adaptation framework designed for scene adaption, I conducted tests using a specific portion of the BDD100k dataset as the target domain, while the Cityscapes training set served as the source domain. The evaluation specifically focused on three subsets of BDD100k representing different times of the day: daytime, nighttime, and dawn/dusk.

For the experiment, I utilized a total of 5,027 training images captured during dawn or dusk as the target data. The source data consisted of 36,728 training images taken during the day and 27,971 training images captured at night. Additionally, 778 validation images taken at dawn or dusk were used for evaluation. The BDD100k dataset was divided into three subsets based on different times of the day: daylight, nighttime, and dawn/dusk. The training model employed in this experiment utilized over 60,000 images from the day and night subsets, which is a significantly larger number compared to the 5,027 dawn/dusk training images used in the oracle experiment. The experiment focuses on adapting from the daytime to the nighttime domain, comparing the performance of the proposed DAOD approach with official implementations of Faster R-CNN, AT, and PT. Despite the difficulty of object detection in nighttime environments, the proposed DAOD approach surpasses other state-of-the-art methods, as shown in Figure 6.4. These findings highlight the superiority of the proposed DAOD approach in addressing object detection challenges in nighttime scenarios compared to existing approaches.

Baseline Approach

Proposed Approach

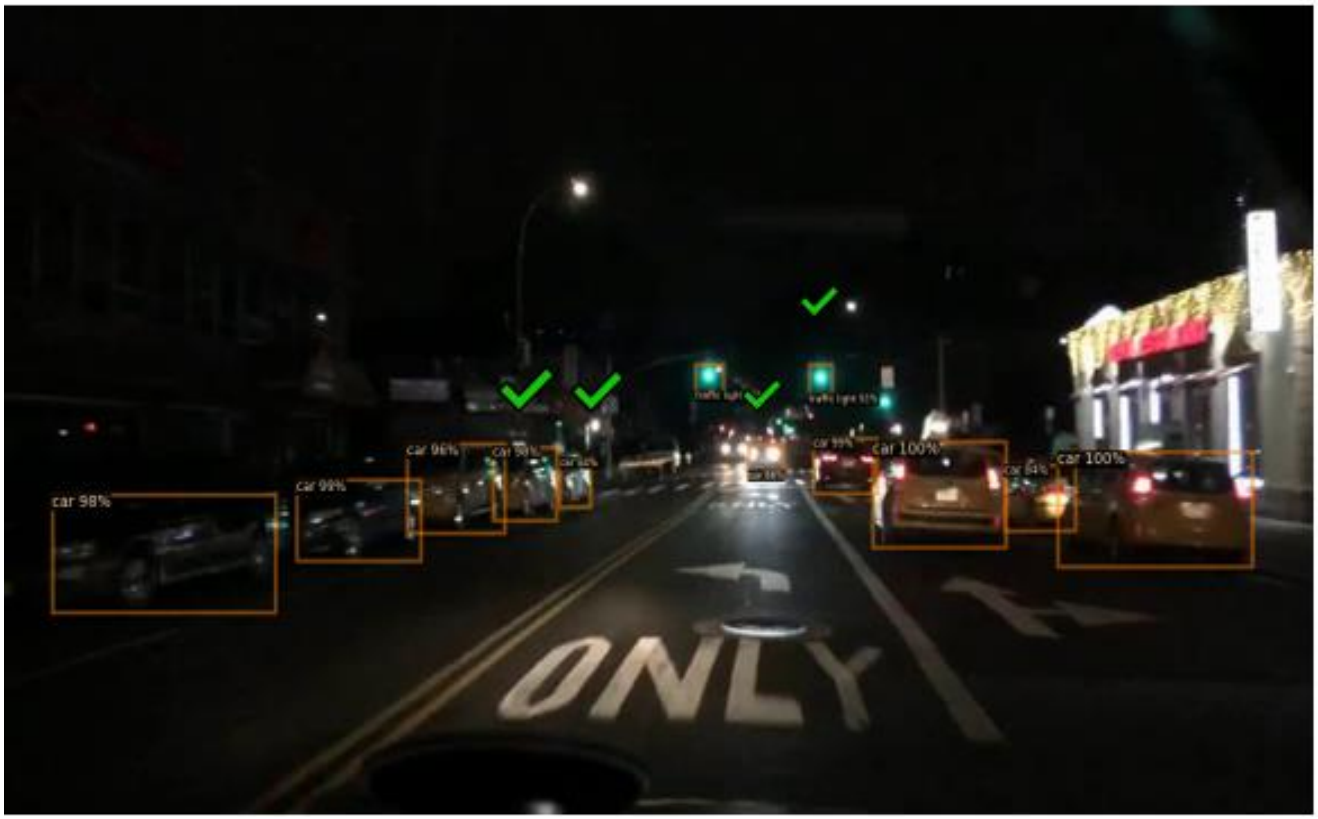


Figure 6.2: Experimental Result on BDD dataset

6.1.2. Cross-camera Adaptation Experiment (KITTI→Cityscapes)

In this section, a domain adaptation experiment is conducted between the KITTI and Cityscapes datasets, which are captured using different camera devices in the same city. The goal is to address the domain shift between these datasets and improve object detection performance in cross-camera scenarios. Figure 6.3 illustrates the ground truth and detection results from the source domain, and the same procedure is applied for the above scenario adaptation. The last detection result demonstrates the performance of the proposed approach, which exhibits higher accuracy and precision compared to the baseline Cross-Domain Adaptive Teacher for Object Detection (CDATOD) approach. The findings suggest that the proposed approach significantly enhances the accuracy and precision of object detection in cross-camera scenarios, surpassing the performance of the baseline AT approach.

Baseline results

Proposed Approach Results

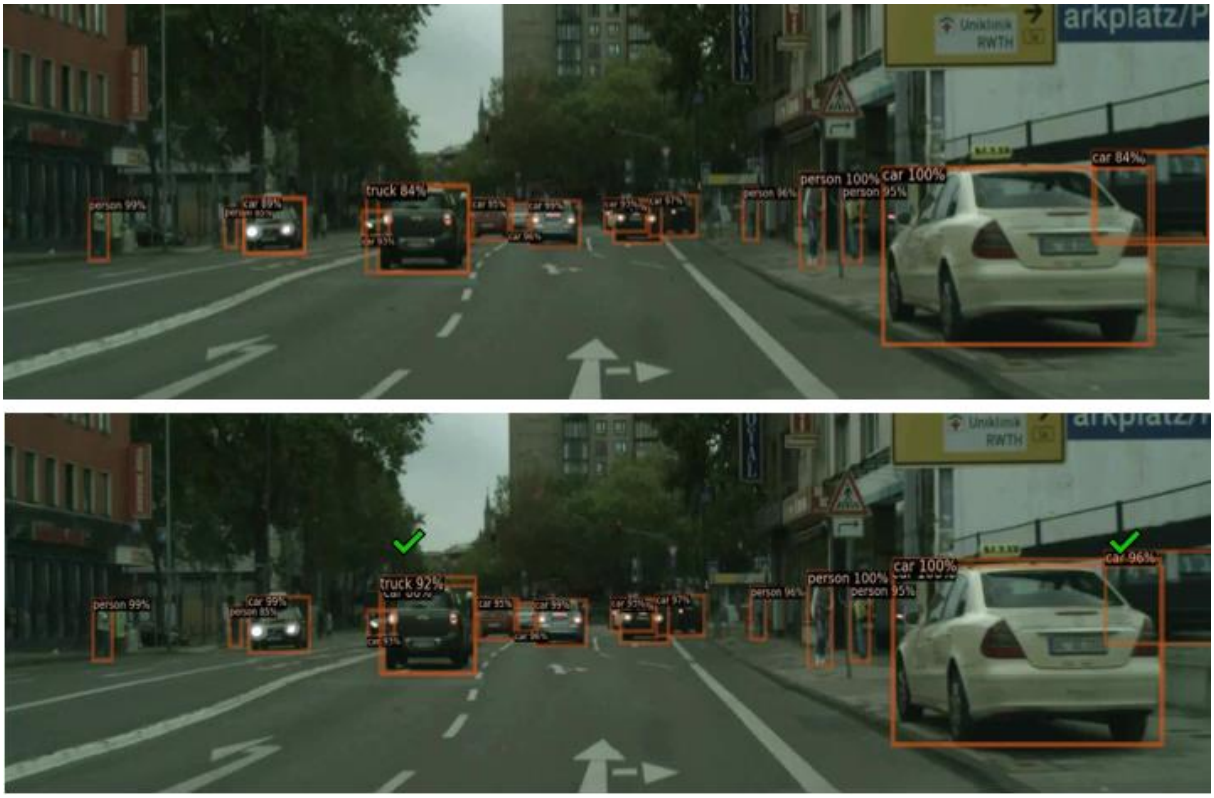


Figure 6.3: Qualitative result on KITTI→Cityscape Adaptation

6.1.3. Synthetic to Real Adaptation Experiment (Cityscapes→ Sim10k)

To address the challenges of limited annotated training data in applications like autonomous driving, a domain-adaptive object detection method is proposed. The proposed method utilizes synthetic datasets generated through simulation, which require less computational effort for

training a detector model compared to annotating real-world data. In this study, a source model is trained using the Sim10k dataset, which is generated from the Grand Theft Auto game engine. The training data comprises 10,000 images and 58,701 bounding boxes specifically for the car category. The target domain for evaluation is the Cityscapes dataset, with the evaluation conducted on the Cityscapes validation set. Since cars are common objects between the Sim10k and Cityscapes datasets, the detector is trained exclusively on annotated cars, focusing on this specific object category.

Baseline results

Proposed Approach Results

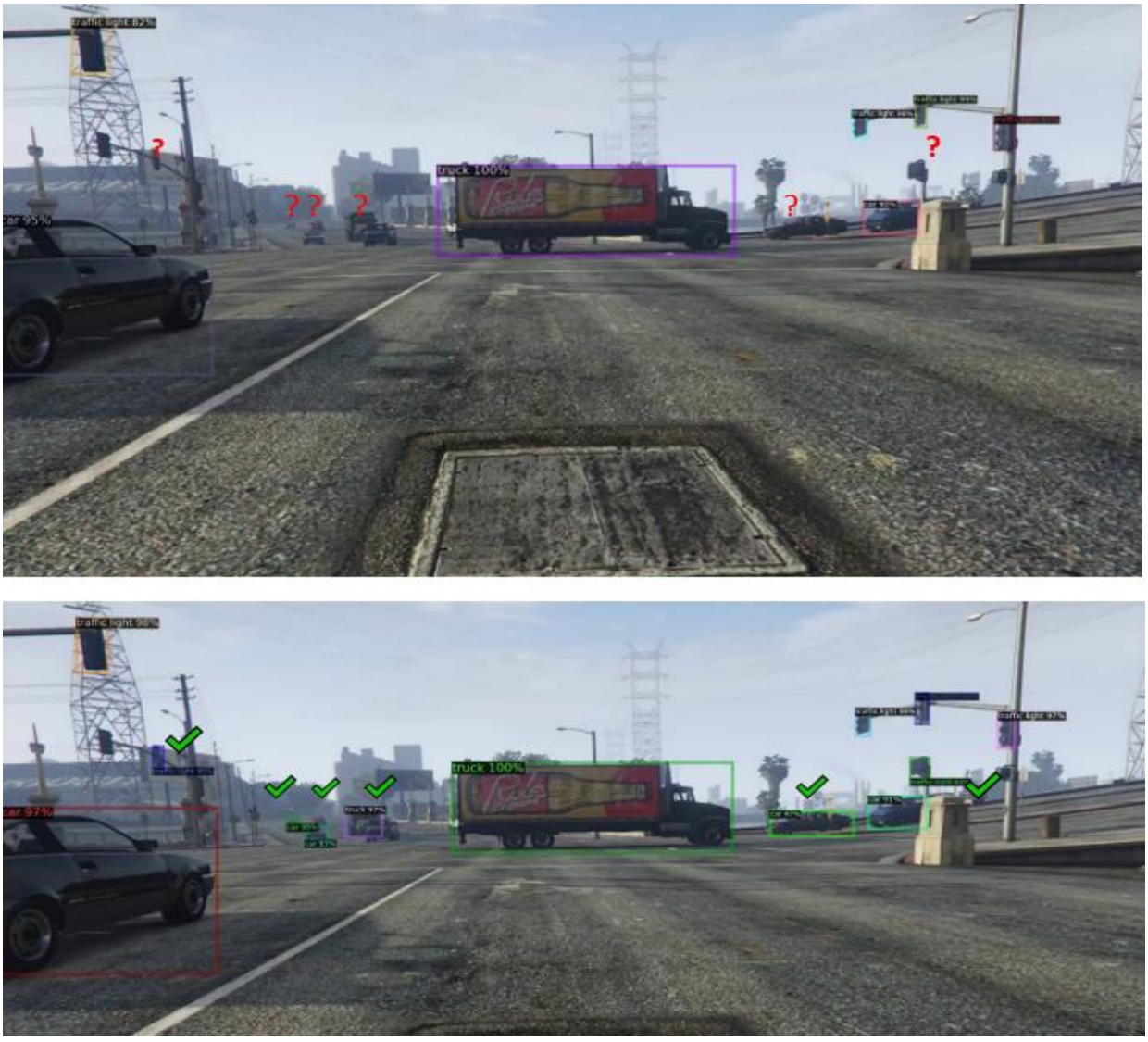


Figure 6.4: Qualitative result on Cityscape→SIM10k Adaptation

6.2. Quantitative Experimental results

6.2.1. Quantitative Results for Cityscape→Foggy Cityscape Adaptation (Cross-Weather Adaptation)

During the cross-weather adaptation experiment, the proposed model demonstrates superior performance compared to the current state-of-the-art methods, achieving a significant improvement of over 9%. Notably, the methods AT and PT, which incorporate semi-supervised learning, suffer from generating noisy labels due to augmentation issues in the first model and bias towards the source domain. These issues result in a performance disparity between the CDATOD and LDAOPT methods and the proposed model.

In the experiment it's found that the eight categories of objects (person, rider, car, truck, Bus, Train, motor, bike) have average precisions (APs) of 50.7,53.7,68.2,35.1,53.0,45.1,58.9 and 49.1, respectively. The suggested domain adaptive object detector with self-training approach raises the mAP from 50.9% to 53.5% in comparison to this baseline model. The mAP may be increased to even better with the use of better feature alignment approach. In conclusion, the suggested DAOD model outperforms the baseline model in terms of mean average precision (mAP) and across all eight categories. The primary objective of adapting the detector to foggy weather conditions using labeled images collected in clear weather is achieved. The experimental results demonstrate a 3.0% performance gain compared to the baseline and the current state-of-the-art, as presented in Table 6.1. Notably, both modules significantly enhance detection performance in foggy conditions, with Faster R-CNN achieving a performance gain of 25.2%, resulting in an overall improvement from 28.3% to 53.5% mAP.

Table 6.1: Quantitative results for Cityscapes → Foggy Cityscape Adaptation Experiment

	person	rider	car	truck	Bus	Train	motor	bike	mAP
source only (Faster RCNN)	29.0	37.2	39.6	8.1	20.1	5.2	16.9	31.9	**23.5**
LDAOPT ((M. He et al., 2022.))	43.2	52.4	63.4	33.4	56.6	37.8	41.3	48.7	**47.1**
CDATOD (Y.-J. Li et al., 2022))	45.5	55.1	64.2	35.0	56.3	54.3	38.5	51.9	**50.9**
CDATOD +AUG+T+NMS+CR (proposed)	**50.7**	**53.7**	**68.2**	**35.1**	**54.0**	**54.1**	**58.9**	**52.1**	**53.5**

6.2.2. Quantitative Results for KITTI→Cityscape Adaptation (Cross-Camera adaptation)

In summary, a cross-camera adaptation experiment was conducted to evaluate the proposed method for adapting object detection models in autonomous driving. The Cityscapes dataset was used as the source, and the KITTI dataset was used as the target. The goal was to improve detection performance across different camera configurations commonly encountered in autonomous driving scenarios by mitigating the domain shift between the datasets. The experiment alternated between using KITTI and Cityscapes as the source and target datasets, respectively, to assess the effectiveness of the proposed method.

During the training phase, the entire train split of the Cityscapes dataset is used, while the KITTI dataset is utilized for testing. According to Table 6.2, the proposed method shows a significant improvement over the baseline Faster R-CNN model, with a mean average precision (mAP) that is more than 20% higher. These results highlight the effectiveness of the proposed methods. Specifically, in the car category, where the domain shift between the source and target datasets is substantial, the proposed method outperforms LDAOPT and CDATOD by 2.1 % and 0.6%, respectively. This suggests that the proposed method is slightly affected by overfitting in this category. However, overall, the proposed method consistently demonstrates superior mAP

performance across various datasets with different camera types.

In summary, the evaluation results confirm that the proposed method achieves remarkable performance in adapting object detection models to different datasets with diverse camera configurations. The improvements in mAP validate the effectiveness of the proposed approach in addressing the challenges of domain shift and adapting to varying camera settings.

Table 6.2: Qualitative results for KITTI→Cityscapes Adaptation Experiment

Method	**AP (car)**
Source only (Faster RCNN)	40.3
LDAOPT (M. He et al., 2022.)	60.2
CDATOD (Y.-J. Li et al., 2022))	61.5
CDATOD +AUG+T+NMS+CR (proposed)	62.1

6.2.3. Quantitative Results for Sim10k → Cityscapes Adaptation (Synthetic to Real Adaptation)

The effectiveness of the proposed approach in adapting from synthetic to real data is demonstrated through evaluation using the SIM10K and Cityscapes datasets. SIM10K represents the synthetic data generated by the GTA-V game engine, while Cityscapes consists of real-world images. In this evaluation. The results in Table 6.3 show that the proposed model achieves a mAP of 60.1%, which is 1.9% higher than AT and 5% higher than PT. This indicates that adapting from virtual to real data can significantly improve detection accuracy by capturing inconsistencies in visual representations. The proposed method, with Resnet backbone and feature-level self-training, achieves a substantial performance gain of 24.6% over the baseline and outperforms the current state-of-the-art by 6.4%, resulting in a mAP of 60.1%. The suggested technique performs much better than all of the previous DAOD methods, including AT and PT. As a result, the suggested DAOD approach can generalize well when the domain is changed from synthetic to real-world.

Table 6.3: Qualitative results for Sim10k→Cityscapes Adaptation Experiment

Method	**mAP (S→ C)**
source only (Faster RCNN)	35.5
LDAOPT (**Y.**-J. Li et al., 2022))	**55.1**
CDATOD (**(**M. He et al., 2022))	**58.2**
AT+AUG+T+NMS+CR (proposed)	**60.1**

6.3. Evaluation of Pseudo Labels

In domain adaptive object detection, pseudo labels are commonly used to bridge the domain gap between the source and target domains. These pseudo labels are generated by applying a pre-trained source model to the target domain data. However, it is crucial to evaluate the quality and reliability of these pseudo labels to ensure their effectiveness in the domain adaptation process. One way to evaluate the pseudo labels is by analyzing the True Positives (TP), False Positives (FP), and False Negatives (FN). By analyzing the distribution of True Positives, False Positives, and False Negatives, we can gain insights into the performance and reliability of the pseudo labels. A high number of True Positives and a low number of False Positives and False Negatives indicate accurate and reliable pseudo labels, suggesting successful domain adaptation. Evaluating the TP, FP, and FN values allows us to calculate important metrics such as Precision and Recall. Precision represents the proportion of correctly identified target objects among all the instances labeled as positive. It indicates the accuracy of the pseudo labels in identifying the target objects. Recall, on the other hand, represents the proportion of correctly identified target objects among all the actual instances of the target objects. It indicates the completeness or coverage of the pseudo labels in capturing the target objects. This evaluation helps in understanding the performance of the pseudo labeling process and making informed decisions regarding the use and further improvement of the pseudo labels in the domain adaptation pipeline.

Table 6.4: Evaluation of pseudo labels for cityscape→foggy cityscape adaptation

Class	True Positives	False Positives	False Negatives
Person	2337	25	10
Rider	265	15	9
Car	5685	40	18
Truck	1069	50	13
Bus	808	23	8
Train	204	12	3
Motor	285	16	2
Bike	76	18	6

6.4. Sample Training log of the Proposed Model

6.4.1. Classification Loss

In the case of classification loss, each pixel needs to be assigned a specific class. The variable yi takes on a binary value of 0 or 1, depending on whether pixel I belongs to the current reconstructed class or object being tracked. The assignment of classes to pixels can be obtained from supervised data or through an unsupervised approach. The classification loss and domain confusion loss serve different purposes in training the network. The classification loss is used to encourage distinct feature representations for different classes, ensuring accurate classification. On the other hand, the domain confusion loss aims to promote domain-invariant representations, enabling the model to generalize well across different domains. During training, the network optimizes both losses alternatively, striking a balance between class distinctiveness and domain invariance to achieve optimal performance.

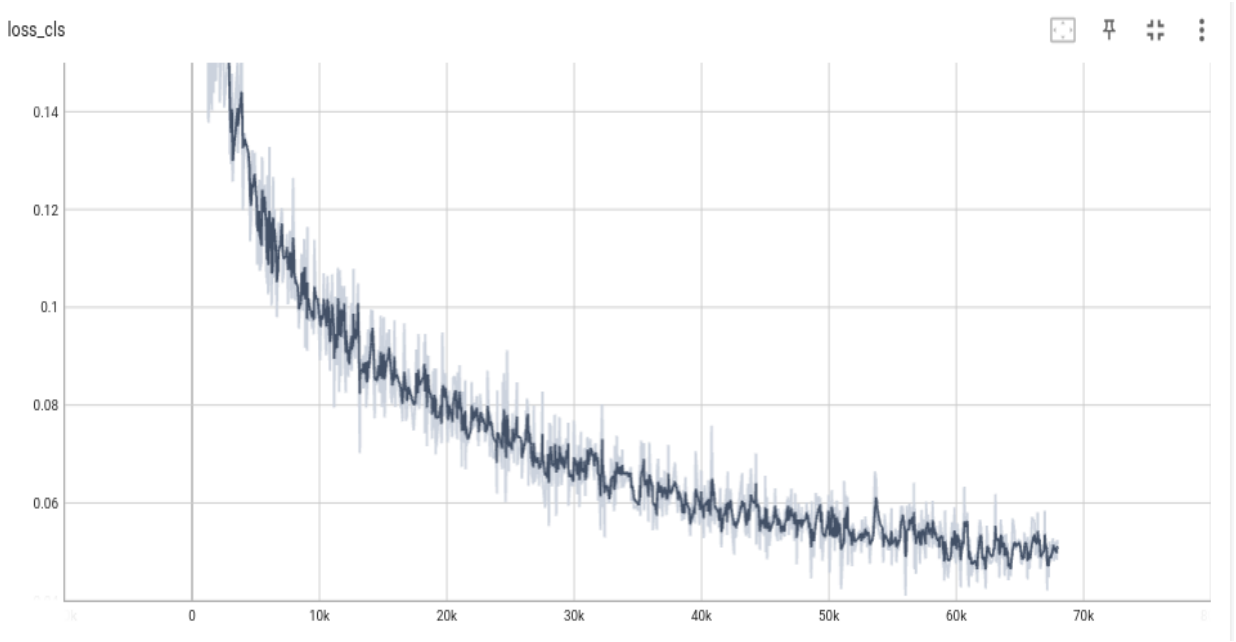


Figure 6.4: The classification Loss

6.4.2. Domain Loss

The proposed method includes a domain loss metric to evaluate and minimize the domain shift between the source and target domain images during training. This metric focuses on aligning the low-level characteristics of the two domains to reduce the disparity. The architecture introduces additional convolutional layers on top of the backbone network to generate multi-scale detection predictions. The Pool1 layer's output is used to compute the domain loss, which measures the domain shift. By optimizing the model based on this loss, the network learns to align the low-level properties of the domains, improving the model's adaptability. The domain adaptor consists of three fully connected layers and is introduced after the reversal layer. Both the domain loss (Ld) and the classification loss (Ly) are defined as cross-entropy losses. During training, batches consisting of 128 samples are utilized. Mean subtraction is employed as a preprocessing step for all input samples. The batch composition consists of an equal distribution of original materials.

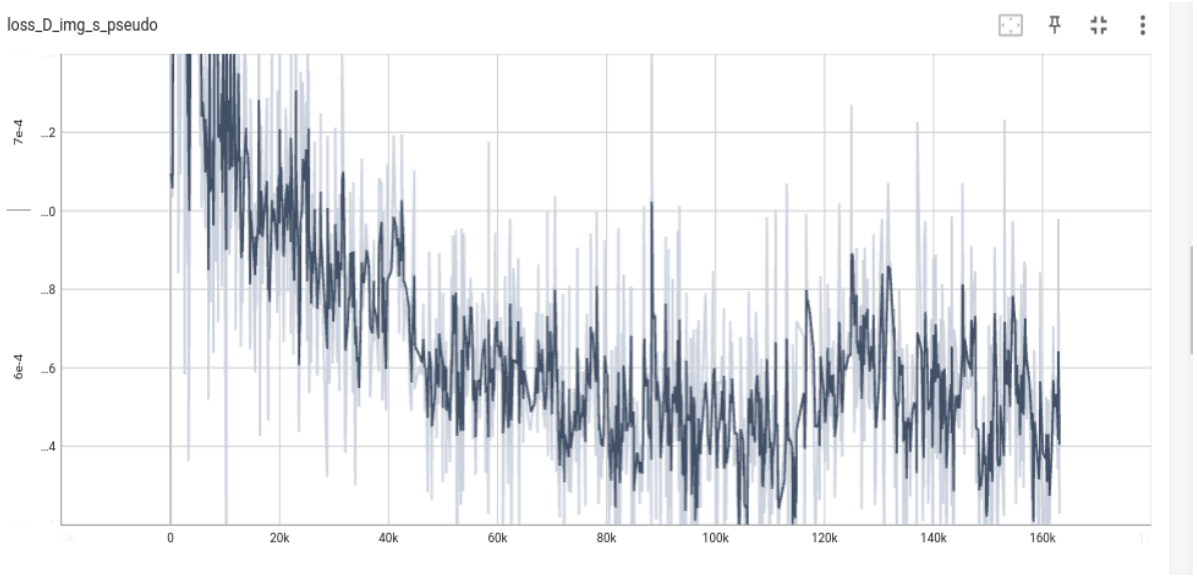


Figure 6.5: The domain Loss

6.5. Results Discussion

6.5.1. Discussion on each Adaptation Scenarios

In this section, I extensively compare and evaluate the proposed domain adaptive object detection algorithm, which utilizes semi-supervised learning and self-training techniques. I thoroughly examine its effectiveness in various domain shift scenarios, including cross weather, cross camera, synthetic to real, and cross scene adaptation. I compare the algorithm's performance against the baseline Faster R-CNN model and other existing domain adaptation methods. Across all experiments, the results consistently show that the proposed method outperforms the baselines, achieving significant improvements in detection accuracy and adaptability. These findings strongly highlight the effectiveness and superiority of the proposed domain adaptive object detection algorithm in addressing domain shift challenges in diverse scenarios.

In order to give an accurate and thorough comparison, I consider relative mAP improvement, which measures the performance of a certain technique relative to the matching baseline as published in the relevant study. The AP50 evaluation measure, which is based on the conventional COCO evaluation metrics, has been used to thoroughly to analyze the stated improvements on the custom distinct dataset. It is demonstrated that the proposed method outperforms the state-of-the-art domain adaptive semi-supervised models proposed by Liu et al. (2022) and Xu et al (2022).

Furthermore, the proposed framework performs better not only mAP50 but also mAP75 in all settings consistently which shows the improvement on the detection performance.

In the cityscape to foggy cityscape scenario, the proposed method exhibits a remarkable improvement of over 2.6% in average precision (AP) compared to the baseline model. It surpasses other state-of-the-art models and achieves higher accuracy specifically in this scenario. the proposed method nearly doubles the mean average precision (mAP) of the source-only Faster RCNN model with a ResNet backbone, significantly increasing it from 28.3% to 53.5%. Additionally, it outperforms other approaches by at least 1% in terms of mAP.

Modifications were made to the baseline Cross-Domain Adaptive Teacher for Object Detection (CDATOD), incorporating instance-level domain alignment and domain loss components. Instance-level alignment generated domain-invariant features in specific regions of the Faster-RCNN network, which serves as the base detector in the CDATOD model. The domain loss term reduced the discrepancy between image-level and instance-level alignment losses. A comparison with various methodologies was conducted, demonstrating that the original Adaptive Teacher model achieved a maximum average precision at an intersection over union threshold of 50% (max (AP50)) of 50.9%. When combined with instance-level alignment and adversarial learning, the suggested semi-supervised detection architecture delivered competitive results, reaching a max (AP50) of 53.5% more efficiently than the original model.

The qualitative results shown in Figure 6.1 illustrate the effectiveness of the proposed approach in the cross-weather adaptation experiment. The proposed method demonstrates superior generalization to challenging cases, although there is still a performance gap compared to the oracle result, particularly in the SIM10K to Cityscapes scenario. This indicates the need for more advanced domain adaptation methods to further enhance performance and reduce the gap. Overall, while the proposed approach shows promising results in cross-weather adaptation, there is room for further improvement to match the performance of the oracle result. Nonetheless, it demonstrates superior performance compared to baseline approaches, particularly in the Cityscapes to Foggy Cityscapes scenario

To enhance the representation of domain-invariant features, the inclusion of multi-adversarial domain classifiers on additional convolutional layers can significantly improve domain adaptation

performance. In the context of unconstrained object detection, the final domain alignment using well-aligned domain features produces notably better results compared to the proposed feature alignment approach. This indicates the importance of effectively aligning domain-specific features to achieve better adaptation performance in object detection tasks.

For synthetic to real Adaptation case a performance discrepancy between the source and target domains from the baseline and Oracle results presented in Table 6.3. To conduct the experiment, I utilize the category "Car." The table shows that the AP of proposed model higher than that of AT (baseline), PT. As a result, investigations suggest that virtual reality style transfer may enhance detection precision by enabling the model to more accurately catch inconsistencies across objects.

Once the source and target domains are aligned using the proposed approach, the network may encounter challenges when adapting to a more diverse dataset without the inclusion of synthetic data. However, by incorporating synthetic data into the source training set, the model learns to generalize more effectively to the target domain. This results in a performance improvement of 6.4% compared to when synthetic data is not utilized.

CHAPTER SEVEN

CONCLUSION AND FUTURE WORK

7.1. CONCLUSIONS

This study aimed to create a neural network that excels in detecting objects accurately and with high recall across different domains. The research results emphasize the effectiveness of domain adaptation techniques, such as self-supervision and training on a diverse dataset covering various domains. Moreover, the study demonstrates the benefits of domain adaptation in improving the accuracy of pseudo- labeling processes and proposes methods for evaluating such annotations within the context of semi-supervised learning. This thesis serves as a valuable contribution to the field of domain adaptation, which still requires further exploration. Additionally, the research showcases that semi-supervised learning techniques utilizing pseudo-labeling face challenges in complex scenarios. This can be attributed to the continuous improvement of predicted annotations, which negatively impacts the performance of convolutional neural networks (CNNs). Furthermore, the thesis presents a minor contribution related to techniques for mutually adjusting a CNN detection model and the training dataset, particularly focusing on two-stage object detection architectures.

I presented a new cross-domain object identification architecture after examining various approaches. The design is based mostly on the Cross-Domain Adaptive Teacher for Object Detection (CDATOD) implementation, which uses semi-supervised learning, with feature alignment, and pseudo labeling to align two domains.

The proposed approach has achieved state-of-the-art results in domain adaptation scenarios, including both single and multiple source settings, on established benchmarks. In situations where the features become diverse due to variations in visual distributions and weather conditions, the network demonstrates superior ability to distinguish these features compared to other approaches. Specifically, when dealing with target domains that have a similar scene structure but different weather conditions, the proposed approach shows remarkable performance even with a deep neural network (DNN) that has fewer parameters. This highlights the effectiveness of the approach in capturing and leveraging the subtle differences in features to improve domain adaptation performance.

To overcome the domain shift challenges related to global image style and local object appearance, a comprehensive approach combining image-level and object-level feature alignment was introduced. In addition, a gradient reversal layer was proposed to facilitate adversarial learning and domain adaptation. The proposed object detection method has extensive applications and effectively tackles general domain adaptive object detection challenges. Through domain adaptation experiments, the approach was evaluated in scenarios such as adapting from Cityscapes to Foggy Cityscapes, Cityscapes to KITTI, and other datasets. These experiments aimed to showcase the method's ability to adapt across diverse domains and datasets successfully.

The presented approach effectively tackles overfitting in semi-supervised object detection by enhancing the quality of pseudo labels. This is achieved through the use of non-maximum suppression, which merges recent detection results with historical pseudo label results, ensuring high-quality labels are retained while eliminating low-quality ones. The proposed method's performance is evaluated using mean average precision (mAP) across different datasets, and extensive experiments demonstrate its feasibility. It is worth noting that the proposed method outperforms many source-based domain adaptation models. However, it is important to consider the potential limitations arising from limited source domain datasets. Careful selection of the appropriate source model becomes crucial to achieve optimal performance, especially when a single source dataset may not fully represent the target domain's features.

7.2. Future Works

There hasn't been much work done to predict the domain shift across datasets or to understand what causes this bias. The research of knowledge transfer across domains once domains have been identified from a combination of several datasets has been done. Although it may not explicitly model a specific domain, this approach offers insights for identifying domains through analysis. In computer vision, challenges related to domain shift primarily arise from differences in representation, not just variations in the data being represented. Variations can stem from various aspects such as imaging (camera viewpoint, occlusion), storage (resolution, size), and representation (color, brightness, contrast). Including contextual information in the image can further enhance diversity. Additionally, the visual changes can arise from the inherent diversity in the "actual data" itself. Most domain adaptation systems aim to build adaptive models capable of general domain adaptation. On the other hand, it can be advantageous to modify the adaptation model to account for a particular data variance. However, doing so would need a thorough grasp of domain shift. Additionally, depending on the nature of domain-shift, it may result in task-specific domain adaptation models, which would boost the use of domain adaptation in practical applications.

Future research in the domain of domain-adaptive object recognition using semi supervised learning should focus on enhancing the data augmentation approach. There is potential for further experimentation and refinement of the parameters in strong data augmentation techniques to improve the performance of the models. Additionally, the concept and principles of the proposed technique can be applied in the development of more complex object detectors. Currently, the ResNet101 Faster R-CNN algorithm is used as the instructor object detector, but by switching the backbone network to the first network, the second object detector can potentially achieve better performance by leveraging more accurate predictions. The proposed approach enables the training of a stable and efficient object detector, and efforts can be made to align different domain distributions to further enhance the detector's performance.

REFERENCES

Almabdy, S., & Elrefaei, L. (2019). Deep convolutional neural network-based approaches for face recognition. Applied Sciences (Switzerland), 9(20). https://doi.org/10.3390/app9204397

Alqasir, H., Muselet, D., & Ducottet, C. (2020). Region Proposal Oriented Approach for Domain Adaptive Object Detection. Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 12002 LNCS. https://doi.org/10.1007/978-3-030-40605-9_4

Aqeel Anwar. (n.d.). What is Average Precision in Object Detection & Localization Algorithms and how to calculate it? Retrieved January 26, 2023, from https://towardsdatascience.com/what-is-average-precision-in-object-detection-localization-algorithms-and-how-to-calculate-it-3f330efe697b

Cai, Q., Pan, Y., Ngo, C. W., Tian, X., Duan, L., & Yao, T. (2019). Exploring object relation in mean teacher for cross-domain detection. Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2019-June. https://doi.org/10.1109/CVPR.2019.01172

Callier, P., & Sandel, O. (2021). Introduction to Artificial Intelligence. Actualites Pharmaceutiques, 60(611). https://doi.org/10.1016/j.actpha.2021.10.005

Chaudhari, S., Malkan, N., Momin, A., & Bonde, M. (2020). Yolo Real Time Object Detection. International Journal of Computer Trends and Technology, 68(6). https://doi.org/10.14445/22312803/ijctt-v68i6p112

Chen, M., Chen, W., Yang, S., Song, J., Wang, X., Zhang, L., Yan, Y., Qi, D., Zhuang, Y., Xie, D., & Pu, S. (2022). Learning Domain Adaptive Object Detection with Probabilistic Teacher. http://arxiv.org/abs/2206.06293

Chen, Y., Liu, Q., Wang, T., Wang, B., & Meng, X. (2021). Rotation-invariant and relation-aware cross-domain adaptation object detection network for optical remote sensing images. Remote Sensing, 13(21). https://doi.org/10.3390/rs13214386

Chen, Y., Li, W., Sakaridis, C., Dai, D., & van Gool, L. (2018). Domain Adaptive Faster R-CNN for Object Detection in the Wild. Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition. https://doi.org/10.1109/CVPR.2018.00352

Chen, Y., Wang, H., Li, W., Sakaridis, C., Dai, D., & van Gool, L. (2021). Scale-Aware Domain Adaptive Faster R-CNN. International Journal of Computer Vision, 129(7). https://doi.org/10.1007/s11263-021-01447-x

Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., & Schiele, B. (2016). The Cityscapes Dataset for Semantic Urban Scene Understanding. Proceedings of the IEEE Computer Society Conference on Computer

Vision and Pattern Recognition, 2016-December. https://doi.org/10.1109/CVPR.2016.350

Deng, J., Li, W., Chen, Y., & Duan, L. (2021). Unbiased Mean Teacher for Cross-domain Object Detection. Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition. https://doi.org/10.1109/CVPR46437.2021.00408

D'Innocente, A., Borlino, F. C., Bucci, S., Caputo, B., & Tommasi, T. (2020). One-Shot Unsupervised Cross-Domain Detection. Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 12361 LNCS. https://doi.org/10.1007/978-3-030-58517-4_43

Ganin, Y., & Lempitsky, V. (2015). Unsupervised domain adaptation by backpropagation. 32nd International Conference on Machine Learning, ICML 2015, 2.

Geiger, A., Lenz, P., & Urtasun, R. (2012). Are we ready for autonomous driving? the KITTI vision benchmark suite. Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition. https://doi.org/10.1109/CVPR.2012.6248074

Geng, H., Jiang, J., Shen, J., & Hou, M. (2022). Cascading Alignment for Unsupervised Domain-Adaptive DETR with Improved DeNoising Anchor Boxes. Sensors (Basel, Switzerland), 22(24). https://doi.org/10.3390/s22249629

Girshick, R. (2015). Fast R-CNN. http://arxiv.org/abs/1504.08083

Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition. https://doi.org/10.1109/CVPR.2014.81

Gou, J., Yu, B., Maybank, S. J., & Tao, D. (2021). Knowledge Distillation: A Survey. International Journal of Computer Vision, 129(6). https://doi.org/10.1007/s11263-021-01453-z

Hassan, M. F. A., Hussain, A., Muhamad, M. H., & Yusof, Y. (2019). Convolution neural network-based action recognition for fall event detection. International Journal of Advanced Trends in Computer Science and Engineering, 8(1.6 Special Issue). https://doi.org/10.30534/ijatcse/2019/6881.62019

He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2020). Mask R-CNN. IEEE Transactions on Pattern Analysis and Machine Intelligence, 42(2). https://doi.org/10.1109/TPAMI.2018.2844175

He, M., Wang, Y., Wu, J., Wang, Y., Li, H., Li, B., Gan, W., Wu, W., & Qiao, Y. (n.d.). Cross Domain Object Detection by Target-Perceived Dual Branch Distillation.

He, Z., & Zhang, L. (2019). Multi-adversarial faster-RCNN for unrestricted object detection. Proceedings of the IEEE International Conference on Computer Vision, 2019-October. https://doi.org/10.1109/ICCV.2019.00677

He, Z., & Zhang, L. (2020). Domain Adaptive Object Detection via Asymmetric Tri-Way Faster-RCNN. Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 12369 LNCS. https://doi.org/10.1007/978-3-030-58586-0_19

He, Z., Zhang, L., Gao, X., & Zhang, D. (2022). Multi-adversarial Faster-RCNN with Paradigm Teacher for Unrestricted Object Detection. International Journal of Computer Vision. https://doi.org/10.1007/s11263-022-01728-z

Hnewa, M., & Radha, H. (2021). MULTISCALE DOMAIN ADAPTIVE YOLO FOR CROSS-DOMAIN OBJECT DETECTION. Proceedings - International Conference on Image Processing, ICIP, 2021-September. https://doi.org/10.1109/ICIP42928.2021.9506039

Hsu, H. K., Yao, C. H., Tsai, Y. H., Hung, W. C., Tseng, H. Y., Singh, M., & Yang, M. H. (2020). Progressive domain adaptation for object detection. Proceedings - 2020 IEEE Winter Conference on Applications of Computer Vision, WACV 2020. https://doi.org/10.1109/WACV45572.2020.9093358

Johnson-Roberson, M., Barto, C., Mehta, R., Sridhar, S. N., Rosaen, K., & Vasudevan, R. (2017). Driving in the Matrix: Can virtual worlds replace human-generated annotations for real world tasks? Proceedings - IEEE International Conference on Robotics and Automation. https://doi.org/10.1109/ICRA.2017.7989092

Khodabandeh, M., Vahdat, A., Ranjbar, M., & MacReady, W. (2019). A robust learning approach to domain adaptive object detection. Proceedings of the IEEE International Conference on Computer Vision, 2019-October. https://doi.org/10.1109/ICCV.2019.00057

Kim, S., Choi, J., Kim, T., & Kim, C. (2019). Self-training and adversarial background regularization for unsupervised domain adaptive one-stage object detection. Proceedings of the IEEE International Conference on Computer Vision, 2019-October. https://doi.org/10.1109/ICCV.2019.00619

LeCun, Y., Hinton, G., & Bengio, Y. (2015). Deep learning (2015), Y. LeCun, Y. Bengio and G. Hinton. Nature, 521.

Lin, T. Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., & Zitnick, C. L. (2014). Microsoft COCO: Common objects in context. Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 8693 LNCS (PART 5). https://doi.org/10.1007/978-3-319-10602-1_48

Liu, L., Ouyang, W., Wang, X., Fieguth, P., Chen, J., Liu, X., & Pietikäinen, M. (2020). Deep Learning for Generic Object Detection: A Survey. International Journal of Computer Vision, 128(2). https://doi.org/10.1007/s11263-019-01247-4

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C. Y., & Berg, A. C. (2016). SSD: Single shot multibox detector. Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 9905 LNCS. https://doi.org/10.1007/978-3-319-46448-0_2

Liu, X., Yoo, C., Xing, F., Oh, H., el Fakhri, G., Kang, J.-W., & Woo, J. (2022). Deep Unsupervised Domain Adaptation: A Review of Recent Advances and Perspectives. APSIPA Transactions on Signal and Information Processing, 11(1). https://doi.org/10.1561/116.00000192

Li, W., Li, F., Luo, Y., Wang, P., & Sun, J. (2020). Deep Domain Adaptive Object Detection: A Survey. 2020 IEEE Symposium Series on Computational Intelligence, SSCI 2020. https://doi.org/10.1109/SSCI47803.2020.9308604

Li, Y.-J., Dai, X., Ma, C.-Y., Liu, Y.-C., Chen, K., Wu, B., He, Z., Kitani, K., & Vajda, P. (n.d.-a). Cross-Domain Adaptive Teacher for Object Detection.

Li, Y.-J., Dai, X., Ma, C.-Y., Liu, Y.-C., Chen, K., Wu, B., He, Z., Kitani, K., & Vajda, P. (n.d.-b). Cross-Domain Adaptive Teacher for Object Detection.

Long, M., Cao, Y., Wang, J., & Jordan, M. I. (2015). Learning transferable features with deep adaptation networks. 32nd International Conference on Machine Learning, ICML 2015, 1.

MICHAEL COPELAND. (n.d.). What's the Difference Between Artificial Intelligence, Machine Learning and Deep Learning? Retrieved January 29, 2023, from https://blogs.nvidia.com/blog/2016/07/29/whats-difference-artificial-intelligence-machine-learning-deep-learning-ai/

Mohan Dogra. (n.d.). Weakly Supervised Learning for Object Localization. Retrieved January 26, 2023, from https://medium.com/analytics-vidhya/weakly-supervised-learning-for-object-localization-4b73d4f4f4a6

Motiian, S., Piccirilli, M., Adjeroh, D. A., & Doretto, G. (2017). Unified Deep Supervised Domain Adaptation and Generalization. Proceedings of the IEEE International Conference on Computer Vision, 2017-October. https://doi.org/10.1109/ICCV.2017.609

Nguyen, D. K., Tseng, W. L., & Shuai, H. H. (2020). Domain-Adaptive Object Detection via Uncertainty-Aware Distribution Alignment. MM 2020 - Proceedings of the 28th ACM International Conference on Multimedia. https://doi.org/10.1145/3394171.3413553

Oza, P., Sindagi, V. A., VS, V., & Patel, V. M. (2021). Unsupervised Domain Adaptation of Object Detectors: A Survey. http://arxiv.org/abs/2105.13502

Ravindra Parmar. (n.d.-a). Demystifying Convolutional Neural Networks. Retrieved January 26, 2023, from https://towardsdatascience.com/demystifying-convolutional-neural-networks-384785791596

Ravindra Parmar. (n.d.-b). Detection and Segmentation through ConvNets. Retrieved January 26, 2023, from https://towardsdatascience.com/detection-and-segmentation-through-convnets-47aa42de27ea

Ren, S., He, K., Girshick, R., & Sun, J. (2017). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. IEEE Transactions on Pattern Analysis and Machine Intelligence, 39(6). https://doi.org/10.1109/TPAMI.2016.2577031

Rodriguez, A. L., & Mikolajczyk, K. (2020). Domain adaptation for object detection via style consistency. 30th British Machine Vision Conference 2019, BMVC 2019.

Saito, K., Ushiku, Y., Harada, T., & Saenko, K. (2019). Strong-weak distribution alignment for adaptive object detection. Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2019-June. https://doi.org/10.1109/CVPR.2019.00712

Sakaridis, C., Dai, D., & van Gool, L. (2018). Semantic Foggy Scene Understanding with Synthetic Data. International Journal of Computer Vision, 126(9). https://doi.org/10.1007/s11263-018-1072-8

Sambasivarao. K. (n.d.). Region of Interest Pooling. Retrieved January 26, 2023, from https://towardsdatascience.com/region-of-interest-pooling-f7c637f409af

Shubham.jain Jain. (n.d.). Introduction to Pseudo-Labelling: A Semi-Supervised learning technique. Retrieved January 26, 2023, from https://www.analyticsvidhya.com/blog/2017/09/pseudo-labelling-semi-supervised-learning-technique/

Sindagi, V. A., Oza, P., Yasarla, R., & Patel, V. M. (2020). Prior-Based Domain Adaptive Object Detection for Hazy and Rainy Conditions. Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 12359 LNCS. https://doi.org/10.1007/978-3-030-58568-6_45

Soviany, P., Ionescu, R. T., Rota, P., & Sebe, N. (2021). Curriculum self-paced learning for cross-domain object detection. Computer Vision and Image Understanding, 204. https://doi.org/10.1016/j.cviu.2021.103166

Tang, Y., Chen, W., Luo, Y., & Zhang, Y. (n.d.). Humble Teachers Teach Better Students for Semi-Supervised Object Detection. http://yihet.com/humble-teacher

Tarvainen, A., & Valpola, H. (2017). Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. Advances in Neural Information Processing Systems, 2017-December.

Wang, L., & Yoon, K. J. (2022). Knowledge Distillation and Student-Teacher Learning for Visual Intelligence: A Review and New Outlooks. In IEEE Transactions on Pattern Analysis and Machine Intelligence (Vol. 44, Issue 6). https://doi.org/10.1109/TPAMI.2021.3055564

Wang, M., & Deng, W. (2018). Deep visual domain adaptation: A survey. Neurocomputing, 312. https://doi.org/10.1016/j.neucom.2018.05.083

Wang, X., Huang, T. E., Liu, B., Yu, F., Wang, X., Gonzalez, J. E., & Darrell, T. (2021). Robust Object Detection via Instance-Level Temporal Cycle Confusion. Proceedings of the IEEE International Conference on Computer Vision. https://doi.org/10.1109/ICCV48922.2021.00901

Wang, Y., Zhang, R., Zhang, S., Li, M., Xia, Y. Y., Zhang, X. S., & Liu, S. L. (2021). Domain-Specific Suppression for Adaptive Object Detection. Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition. https://doi.org/10.1109/CVPR46437.2021.00948

Weiss, K., Khoshgoftaar, T. M., & Wang, D. D. (2016). A survey of transfer learning. Journal of Big Data, 3(1). https://doi.org/10.1186/s40537-016-0043-6

Wei, X., Bai, T., Zhai, Y., Chen, L., Luo, H., Zhao, C., & Lu, Y. (2022). Source-free domain adaptive object detection based on pseudo-supervised mean teacher. Journal of Supercomputing. https://doi.org/10.1007/s11227-022-04915-4

Wu, A., Han, Y., Zhu, L., & Yang, Y. (2022). Instance-Invariant Domain Adaptive Object Detection Via Progressive Disentanglement. IEEE Transactions on Pattern Analysis and Machine Intelligence, 44(8). https://doi.org/10.1109/TPAMI.2021.3060446

Wulfmeier, M., Bewley, A., & Posner, I. (2018). Incremental adversarial domain adaptation for continually changing environments. Proceedings - IEEE International Conference on Robotics and Automation. https://doi.org/10.1109/ICRA.2018.8460982

Xiao, B., Zhang, Y., Chen, Y., & Yin, X. (2021). A semi-supervised learning detection method for vision-based monitoring of construction sites by integrating teacher-student networks and data augmentation. Advanced Engineering Informatics, 50. https://doi.org/10.1016/j.aei.2021.101372

Xie, R., Yu, F., Wang, J., Wang, Y., & Zhang, L. (2019). Multi-level domain adaptive learning for cross-domain detection. Proceedings - 2019 International Conference on Computer Vision Workshop, ICCVW 2019. https://doi.org/10.1109/ICCVW.2019.00401

Xu, C. D., Zhao, X. R., Jin, X., & Wei, X. S. (2020). Exploring categorical regularization for domain adaptive object detection. Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition. https://doi.org/10.1109/CVPR42600.2020.01174

Xu, M., Wang, H., Ni, B., Tian, Q., & Zhang, W. (2020). Cross-domain detection via graph-induced prototype alignment. Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition. https://doi.org/10.1109/CVPR42600.2020.01237

Yang, S., Wu, L., Wiliem, A., & Lovell, B. C. (2021). Unsupervised Domain Adaptive Object Detection Using Forward-Backward Cyclic Adaptation. Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 12624 LNCS. https://doi.org/10.1007/978-3-030-69535-4_8

Yang, X., Wan, S., & Jin, P. (2020). Domain-invariant region proposal network for cross-domain detection. Proceedings - IEEE International Conference on Multimedia and Expo, 2020-July. https://doi.org/10.1109/ICME46284.2020.9102766

Yu, F., Chen, H., Wang, X., Xian, W., Chen, Y., Liu, F., Madhavan, V., & Darrell, T. (2020). BDD100K: A Diverse Driving Dataset for Heterogeneous Multitask Learning. Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition. https://doi.org/10.1109/CVPR42600.2020.00271

Yu, Y., Xu, X., Hu, X., & Heng, P. A. (2019). DALocNet: Improving Localization Accuracy for Domain Adaptive Object Detection. IEEE Access, 7. https://doi.org/10.1109/ACCESS.2019.2915607

Zhang, D., Li, J., Xiong, L., Lin, L., Ye, M., & Yang, S. (2019). Cycle-consistent domain adaptive faster RCNN. IEEE Access, 7. https://doi.org/10.1109/ACCESS.2019.2938837

Zhang, D., Ye, M., Liu, Y., Xiong, L., & Zhou, L. (2022). Multi-source unsupervised domain adaptation for object detection. Information Fusion, 78. https://doi.org/10.1016/j.inffus.2021.09.011

Zhao, L., & Wang, L. (2022). Task-specific Inconsistency Alignment for Domain Adaptive Object Detection. https://doi.org/10.1109/cvpr52688.2022.01382

Zhao, S., Yue, X., Zhang, S., Li, B., Zhao, H., Wu, B., Krishna, R., Gonzalez, J. E., Sangiovanni-Vincentelli, A. L., Seshia, S. A., & Keutzer, K. (2022). A Review of Single-Source Deep Unsupervised Visual Domain Adaptation. In IEEE Transactions on Neural Networks and Learning Systems (Vol. 33, Issue 2). https://doi.org/10.1109/TNNLS.2020.3028503

Zheng, Y., Huang, D., Liu, S., & Wang, Y. (2020). Cross-domain object detection through coarse-to-fine feature adaptation. Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition. https://doi.org/10.1109/CVPR42600.2020.01378

Zhu, X., Pang, J., Yang, C., Shi, J., & Lin, D. (2019). Adapting object detectors via selective cross-domain alignment. Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2019-June. https://doi.org/10.1109/CVPR.2019.00078

APPENDICES

Appendix A: Ablation Experiments

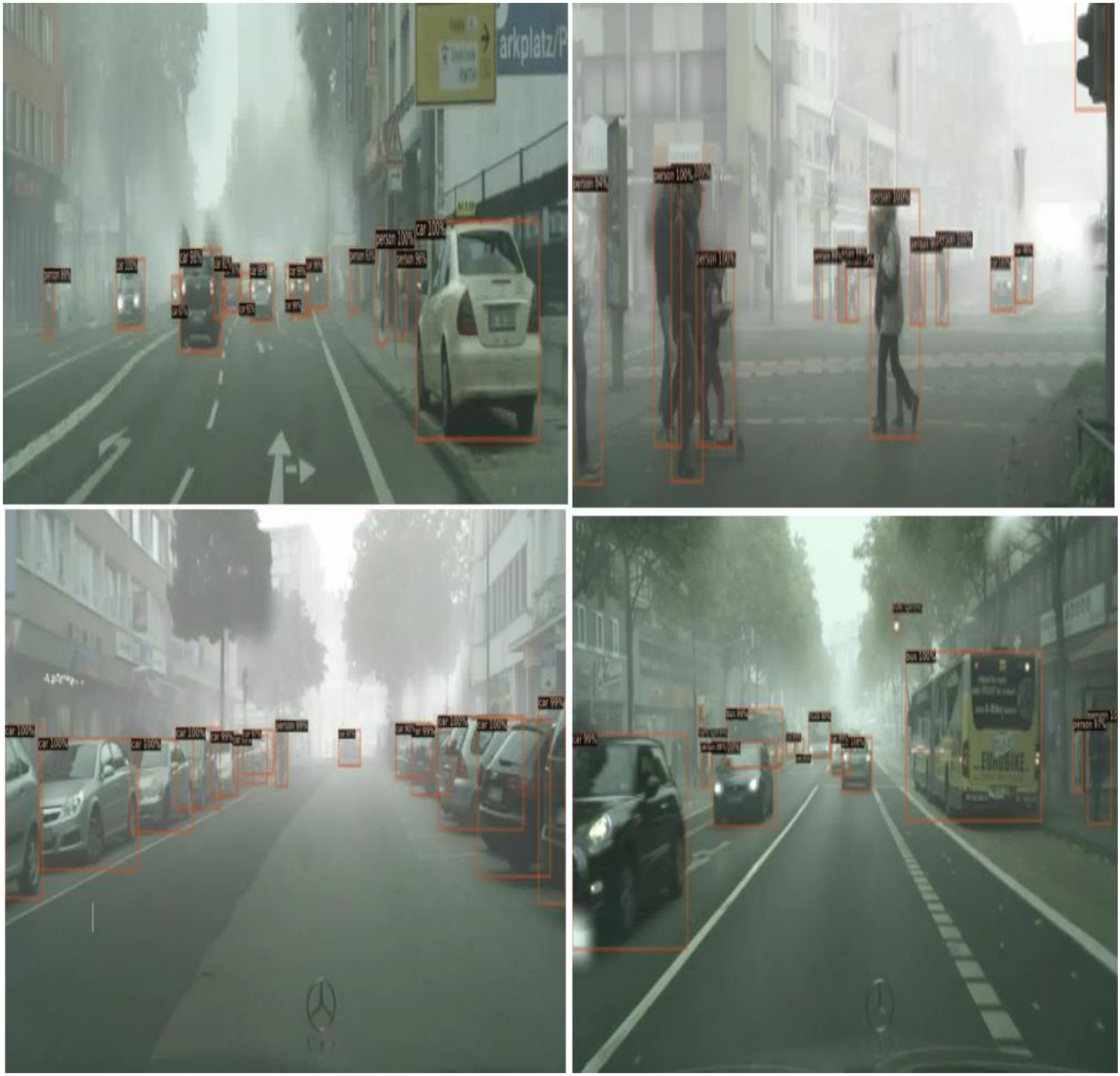
To validate the effectiveness of different parameters, ablation experiments were conducted. The impact of parameter initialization, confidence threshold, NMS rates, and normalization parameter (s) on the model's performance was evaluated. Two versions of the model were trained, one with an initialization step and one without, leading to the conclusion that this technique significantly enhances overall performance. Testing the model with various thresholding values revealed that a value of 0.7 yielded the best results.

Quantitative performance results are presented in Tables 6.1, 6.2, and 6.3. Table 6.1 focuses on weather adaptation, specifically from clear weather to foggy weather. The proposed method exhibits superior detection performance, achieving mAP of 53.5%, surpassing the second-best method (CDATOD) by a 2.6% improvement. Across different categories, the proposed method demonstrates reduced domain gap in Foggy Cityscapes. Notably, it achieves the highest AP in categories such as bus (53.0%), bicycle (49.1%), person (50.1%), rider (53.7%), and train (45.1%). In train detection, the proposed method outperforms the second-best method (CDATOD) by 2.6% AP (41.8% vs. 45.1%) and LDAOPT by a significant margin of 21.3% AP (38.3% vs. 45.1%) in Foggy Cityscapes. While LDAOPT achieves 63.4% AP in the car category and AT achieves 43.2% AP in the person category, the proposed method consistently outperforms them in all categories, exhibiting significant differences. Overall, compared to recent domain adaptation methods, the proposed method achieves the highest mAP and excels in almost all categories.

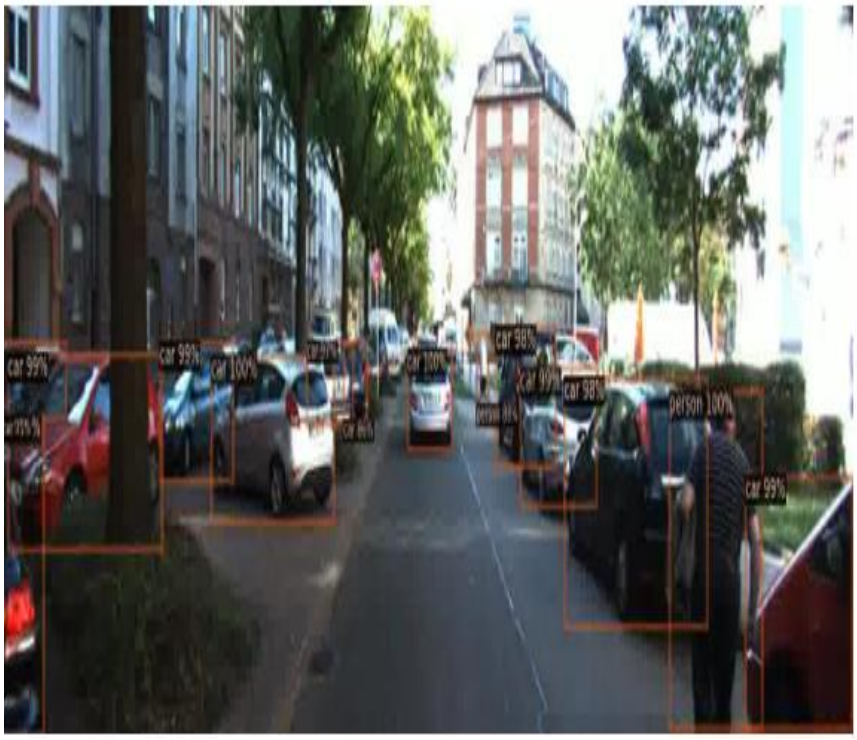
The proposed method randomly selects an unlabeled target picture for feature alignment. The subsequent trials will examine the influence of target sample selection on the effectiveness of domain adaptation.

Appendix B: Detection results on different Domains

More detection results for Cityscape→Foggy cityscape Adaptation



More results on KITTI→Cityscapes Adaptation



Appendix C: Sample Code

Train Baseline Model Network

```python
class BaselineTrainer(DefaultTrainer):
    def __init__(self, cfg):
        """
        Args:
            cfg (CfgNode):
        Use the custom checkpointer, which loads other backbone models
        with matching heuristics.
        """
        cfg = DefaultTrainer.auto_scale_workers(cfg, comm.get_world_size())
        model = self.build_model(cfg)
        optimizer = self.build_optimizer(cfg, model)
        data_loader = self.build_train_loader(cfg)

        if comm.get_world_size() > 1:
            model = DistributedDataParallel(
                model, device_ids=[comm.get_local_rank()], broadcast_buffers=False
            )

        TrainerBase.__init__(self)
        self._trainer = (AMPTrainer if cfg.SOLVER.AMP.ENABLED else SimpleTrainer)(
            model, data_loader, optimizer
        )

        self.scheduler = self.build_lr_scheduler(cfg, optimizer)
        self.checkpointer = DetectionCheckpointer(
            model,
            cfg.OUTPUT_DIR,
            optimizer=optimizer,
            scheduler=self.scheduler,
        )
        self.start_iter = 0
        self.max_iter = cfg.SOLVER.MAX_ITER
        self.cfg = cfg

        self.register_hooks(self.build_hooks())
```

Pseudo Label generation Code

```python
# ============================================================
    # ================== Pseduo-labeling ==================
    # ============================================================
    def threshold_bbox(self, proposal_bbox_inst, thres=0.7, proposal_type="roih"):
        if proposal_type == "rpn":
            valid_map = proposal_bbox_inst.objectness_logits > thres

            # create instances containing boxes and gt_classes
            image_shape = proposal_bbox_inst.image_size
            new_proposal_inst = Instances(image_shape)

            # create box
            new_bbox_loc = proposal_bbox_inst.proposal_boxes.tensor[valid_map, :]
            new_boxes = Boxes(new_bbox_loc)

            # add boxes to instances
            new_proposal_inst.gt_boxes = new_boxes
            new_proposal_inst.objectness_logits = proposal_bbox_inst.objectness_logits[
                valid_map
            ]
        elif proposal_type == "roih":
            valid_map = proposal_bbox_inst.scores > thres

            # create instances containing boxes and gt_classes
            image_shape = proposal_bbox_inst.image_size
            new_proposal_inst = Instances(image_shape)

            # create box
            new_bbox_loc = proposal_bbox_inst.pred_boxes.tensor[valid_map, :]
            new_boxes = Boxes(new_bbox_loc)

            # add boxes to instances
            new_proposal_inst.gt_boxes = new_boxes
            new_proposal_inst.gt_classes = proposal_bbox_inst.pred_classes[valid_map]
            new_proposal_inst.scores = proposal_bbox_inst.scores[valid_map]

        return new_proposal_inst
```