Domain Adaptation for Autonomous Driving

by

Xingxin Chen

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Mechanical and Mechatronics Engineering

Waterloo, Ontario, Canada, 2020

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

Metric based method is a promising approach to domain adaptation, which aims to align the marginal distribution of different domains with a similar conditional distribution. The thesis explores applying domain adaptation for autonomous driving and proposes domain adaptation methods for 2D image semantic segmentation and 3D point cloud object detection. For 2D image semantic segmentation domain adaptation, traditional approaches design metric function manually to measure the distance across domains. The adversarial methods can be considered as an automatic learning approach for the metric function. Instead of depending on the quality of metric function, this thesis outlines a generalized framework for domain randomization which first introduces moderate perturbation as randomness and then combines the advantage of metric-based domain adaptation and domain randomization. Then a simple-to-implement training pipeline of this framework is proposed, which proves that the proposed model achieves comparable performance with metric-based methods while having better generalization performance. The proposed Metric Guided Domain Randomization approach is able to improve mean intersection-over-union on target domain from 16.9 to 27.2 without using any target domain data or annotations. 3D point cloud domain adaptation is an area where researchers pay little attention. A method based on global feature alignment is proposed and experiments show that it has a better performance compared with fine-tuning on target domain data directly when having access to limited number of target data frames.

Acknowledgements

I had received lots of support when I was writing this thesis.

First, I would like to thank my supervisor Prof. Cao. He helped me decide my research direction. His insightful feedback prompts me to achieve better research results.

I want to acknowledge my colleagues from my internship at Huawei Technologies Co., Ltd. Thank you for your suggestions on my research topic and patience when I have lots of questions.

Besides, I would like to thank my parents for their wise counsel and sympathetic ear. I want to thank my friends Jinwei Zhang, Chen Sun, Wenbo Li and Zejian Deng for providing some helpful discussions.

Dedication

This is dedicated to the one I love.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

AP Average Precision 39

BDA Balanced Distribution Adaptation 9

BSP Batch Spectral Penalization 8, 29

CGAN Conditional Generative Adversarial Networks 25, 36, 37, 43

CNN Convolutional Neural Network 2, 4, 8, 10–12, 28

CORAL Correlation Alignment 28

DDC Deep Domain Confusion 10

FCN Fully Connected Neural Network 36

GAN Generative Adversarial Networks 4, 10, 21, 25

i.i.d Identically and Independent Distributed 2, 3

IoU Intersection-over-union 13

JDA Joint Distribution Adaptation 9

mIoU Mean Intersection-over-union 32, 36

MMD Maximum Mean Discrepancy 8–10, 12, 28

RNN Recurrent Neural Network 2

RoI Region of Interest 13, 14

STL Stratified Transfer Learning 9

Chapter 1

Introduction

Recent years, researchers have made significant progress in self-driving vehicles [1]. Many institutes and companies also pour lots of resources into this area. Despite a large amount of data collected every day, it is still insufficient to meet the demands of the ever-increasing AI model complexity required by autonomous vehicles. Autonomous driving has driven lots of attention recently. A fully autonomous vehicle can be safer compared to the car driven by a human being for the fact that humans can not avoid being distracted and that human's reaction time is much longer than computers'. However, fully autonomous vehicles must have human-level perception ability, which can not be satisfied for a long time. Recently, with the development of deep learning [2], computer vision has made significant progress, which makes fully autonomous driving possible.

To fulfill the requirements for data, on the one hand, collecting and labeling more data is necessary. On the other hand, using existing data more efficiently is also of great significance. In this thesis, the possibility of utilizing domain adaptation for autonomous driving is explored. Domain adaptation aims to transfer policy learned in one domain to another relative domain. With domain adaptation, existing data can be used more efficiently to fulfill the data requirements for autonomous driving tasks. In the first section, the thesis introduces the development of deep learning and why it needs lots of annotated data. Then the thesis introduces how to collect and label data more efficiently. Finally, the thesis introduces how to use domain adaptation to utilize existing data more efficiently. In the second section, this thesis introduces existing domain adaptation methods for 2D tasks, then it includes some 3D point cloud data's characteristics and how these characteristics will influence 3D tasks. In the third section, this thesis introduces our proposed methods for both 2D and 3D tasks. In the fourth section, the experiment details and results of the proposed methods are included. In the last section, the thesis discusses the current results and give some recommendations for researchers interested in this area.

1.1 Deep Learning for Autonomous Driving

For autonomous driving, the two most important things are to extract meaningful information from raw sensor data and to predict the future state based on the current state, which includes two general topics in deep learning – computer vision and motion prediction. A typical pipeline is using CNN based model to extract meaning information from raw data, for example, to generate a bird's eye view of the current surrounding environment. Then a RNN based motion predictor will take the past few bird's eye views as input and try to predict the next few bird's eye views. And then the control module can generate a control signal based on the current state and predicted future state.

Different from traditional rule-based methods which try to inference some new conclusions based on the exist axiom theorem system, learning-based methods try to extract a valid policy based on previous experiences. For supervised deep learning algorithms, the experience is data with ground truth annotation. Deep learning methods are based on the assumption that the data used for training and the data used for testing are i.i.d. To make sure i.i.d assumption can be satisfied, lots of data is needed during the training progress, and high-quality annotations [3–5] are also crucial to the final performance of trained models.

1.2 Data Collection and labelling

A large amount of data is necessary to make sure the i.i.d assumption can be satisfied. Ideally, training data needs to be collected from different environments to ensure all the situations have already been met. For autonomous driving, this means prototype autonomous vehicles equipped with all the necessary sensors need to be driven by human export drivers in different cities, different weather, different seasons to collect countless data. It is even more challenging to annotate all these collected data. Each data point needs to be annotated three times or more to ensure that most of the annotations are valid, which requires lots of labour force.

Researchers and industry have already try to make the data collection and labelling progress more efficient [6]. Tesla added a shadow mode to its in-stock vehicles. Shadow mode can collect information when human drivers are driving, and then these collected data can be used for imitation learning. Since these data are collected from Tesla's millions of sold vehicles in different areas, and human driving signals can be used directly as annotations, their data collection and labelling speed is quite remarkable. However, due to the limit of in-stock vehicles' local storage and network bandwidth, Tesla only collects the most meaningful data. For example, they collect the data in three kinds of situation: first, the Autopilot system's output is different

from human drivers' manual output; second, the motion prediction system's output is different from what happens later; third, the human driver chooses to take over the Autopilot system (which usually indicates the human driver are not satisfied with Autopilot's current control police or the surrounding environment is too complicated that the human driver is not confident about the Autopilot system). Even with these selective data collection methods, there is still much data that needs to be annotated. Amazon Web Services provides a SageMaker [7] service, which aims to make the data labelling progress more efficient. With Amazon Mechanical Turk annotators' help, hire labour force to annotate collected can be more flexible, cheap and safe. With SageMaker's built active learning algorithm, the total number of required annotations can also be reduced.

Even with all these efforts, data collection and labeling are still too expensive and time-consuming, which limits the development of fully autonomous vehicles [1]. How to use training data more efficiently then becomes quite crucial for autonomous driving, and also for general deep learning.

1.3 Data Efficiency and Domain Adaptation

Deep neural networks have been proved useful across a variety of tasks under the i.i.d assumption. However, a phenomenon called *domain shift* [8] makes it hard to guarantee this assumption in practice. Typical transfer learning approaches [9] use pre-trained feature extraction models trained on large and labeled datasets and then fine-tune the whole network on a task-specific dataset. However, collecting and annotating enough data to fine-tune the pre-trained network is still expensive and time-consuming. Meanwhile, a tremendous amount of labelled data is required to cover the diversity of real-time running scenarios for engineering applications such as autonomous driving and robotic control. The data collection process is too expensive and sometimes even dangerous.

Simulation environment, however, provides a gateway to the problems as mentioned above. With access to nearly unlimited well-annotated data, safety issues of data collecting progress no longer exist. Although simulation can be of great benefit, there is always a reality gap between simulation and the real world. Current game engine based simulators like CARLA [10] and AirSim [11] require moderate computational resources but can not provide photo-realistic rendering results. Movie level special effects are hard to be distinguished from real-world examples yet even more expensive than collecting data from the real world. One of the most popular approaches to close the gap between simulation and reality is domain adaptation, where the data distribution in one domain is mapping to a shared latent space through metric-based regularization enforced by task model.

The principal idea of current domain adaptation methods is minimizing the distance between the source and target distribution [9]. Pioneer researches attempt to minimize the marginal and conditional distribution of raw data [12, 13]. Following deep learning-based methods attempt to map data from different domains to a common feature space with a CNN feature extractor [14, 15]. Inspired by GAN, a CNN based domain discriminator is utilized instead of handcrafted metrics to measure the distance between different domains [16, 17]. In order to regularize the training process, authors in [18, 19] applied semantic transformation constraints and batch spectral penalization to maintain the content consistency in task mapping to the target domain. Besides minimizing the differences in pixel and feature level, another line of thought considers the instance level difference [20]. Authors of [20] focus on minimizing the distribution difference of scene composition between simulation and reality instead of aligning the semantic and pixel distribution. Meanwhile, extra instance level relationship information [21] and traces on how to learn [22] also contribute to the transfer progress. However, the additional graphic assets are not free, and the domain gap between simulation and real are not solved by combining the game engine's assets with the extracted scene composition distribution.

Besides computer vision, transfer learning can also be used to transfer policy learned in the source domain to the target domain. [23] provided an approach to adapt actions made based on a policy trained in a simulation environment to the real world by setting a learning algorithm after the original policy. This learning algorithm learns from the action decided by the original policy and the state in the simulation. Furthermore, with the learned model, the simulation itself can be optimized. However, it is nearly impossible to adjust the image for some image-based algorithms.

[24, 25] provided another approach. Instead of directly comparing the differences between the reflects of simulation and the real world, they try to learn an inverse dynamic model which takes the current state of the simulation and the expected state in the real world as input in order to decide which is the suitable action the real world robot needs to take. Moreover, the output action can be used directly to control the real world robot.

For some data-driven methods, if the data used for training can be transferred to the real-world domain or close to the real-world domain, the model will be good enough to perform well in the real world. For some image-based algorithms, this is where GAN [26] will help. And for the other algorithms, data from simulation and real-world is mapped the same domain, and then the mapped data can be used to train the model which can work well in the real world while the whole training process needs little real-world data. [27]

For some computer vision problems, it is ubiquitous that researchers use the pre-trained weights of ImageNet, do some fine-tuning in order for the weights to be used in some other similar problems. [28] proposes a model which combines successor representation learning to

deep reinforcement learning in order for the model to better transfer between similar environments. After training of the first environment, training of the other similar environments is quite easy. Apart from the deep learning area, policy adaptation methods can also be found in reinforcement learning area [29–31]. In order for the model to be robust enough to mismatch between simulation and the real world, some random noise is always added to the training data. Some traditional methods about control theory are also listed [32, 33].

In this thesis these approaches are concluded as metric-based approaches with typical pipeline demonstrated in Fig. 1.1. The metric-based domain adaptation highly relies on the accuracy of the metric used for the distance between different domains.

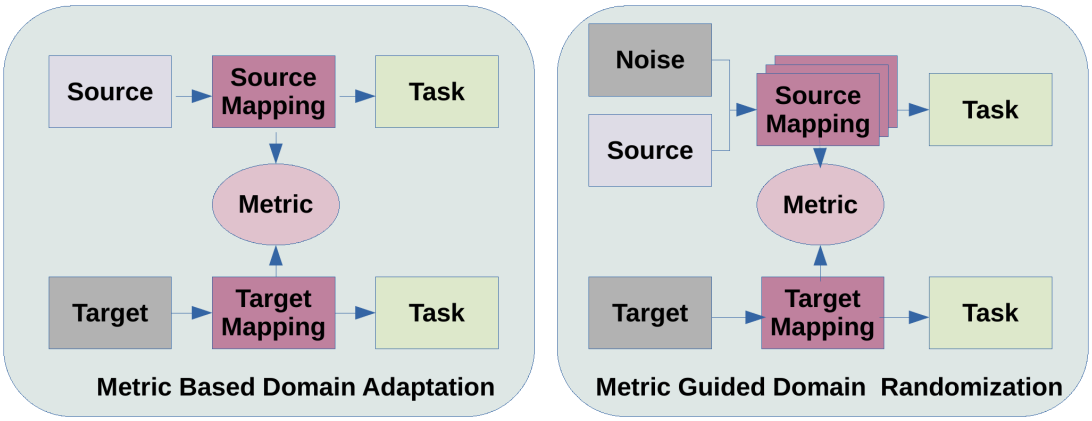

Figure 1.1: A brief framework comparison between popular metric based domain adaptation (left) and the proposed metric guided domain randomization (right).

Meanwhile, current domain randomization methods [34] mainly based on manually designed augmentation techniques, whereas augmentations are solely based on prior knowledge about the difference across two domains. This prior knowledge may not be reliable enough to find a common feature space. Recent style transfer based domain randomization methods [34] accept a style discriminator as the metric for domain randomization. Follow a similar line of thought, this thesis considers metric guided domain randomization that deploys metric based domain adaptation methods for better accuracy and robustness in a domain transfer task. A brief framework comparison between the proposed domain randomization scheme and the traditional metric-based adaptation approaches is shown in Fig. 1.1. Unlike metric based domain adaptation methods which highly rely on the design of metric, our method first combines multiple simple domain adaptation modules to augment the source data marginal distribution towards target data marginal distribution with selected perturbations. Meanwhile, the level of perturbation injected can be tracked, and the distribution transfer learning procedure includes a certain degree

of randomness. A contrastive learning module is utilized to make sure only consistent information between original and randomized data influence the training result. In general, the main contributions of our paper are as follows:

- Outlining a novel framework for domain randomization by synergizing multiple domain adaptation networks to effectively boost transferred task performance in the target domain.
- Designing a domain generalization model with triplet network based on the proposed framework for sim-to-real adaptation, which only requires the prior knowledge to transfer policy. The proposed model utilizes style transfer for domain randomization and further use contrastive learning to regulate the learning process.
- The proposed method is simple to implement and can generalize to various real-world application domains. The thesis analyzes the performance of our proposed model in GTA5 to CityScapes and BDD100K semantic segmentation policy transfer task and show that our model achieves comparable performance with other metric-based methods while having better generalization ability.

Besides trying to improve RGB image domain adaptation performance, this thesis also applies domain adaptation to an unexplored area – Point Cloud. Point cloud differs from RGB images because its raw data is disordered and sparse, which makes domain adaptation methods designed for RGB images not suitable for point cloud domain adaptation. The thesis proposes a baseline method for point cloud domain adaptation that will focus the task-specific decision boundary during global feature alignment progress in order to work on point cloud scenario.

Chapter 2

Related Work

2.1 Domain Adaptation for RGB images

Most of the domain adaptation researches focus on RGB images, try to adapt from source domain D_s to target domain D_t. Normally, domain adaptation methods have access to large amount of source domain data $\{x_i, y_i\}_{i=1}^{n_s} \in D_s$ and also limit amount of target domain data $\{x_j, y_j\}_{j=n_s+1}^{n_s+n_t}$. Task $\mathscr{F}(\cdot)$ is the policy we try to learn. For RGB images, usually it could be classification, object detection, semantic segmentation, etc.. The formulation of a typical domain adaptation problem can then be defined as:

> The source domain D_s and target domain D_t have the same characteristic space $\mathscr{X}_s = \mathscr{X}_t$, and the label space of these two domains $\mathscr{Y}_s = \mathscr{Y}_t$. Most importantly, the two domains should be related so that the conditional distribution of the two domains should be similar $\mathscr{Q}_s(y_s|x_s) \approx \mathscr{Q}_t(y_t|x_t)$. However, the marginal distribution of two domains are different $\mathscr{P}(\mathscr{X}_s) \neq \mathscr{P}(\mathscr{X}_t)$. We try to learn a policy $\mathscr{F} : x_t \mapsto y_t$ based on D_s.

If $n_t = 0$, then it becomes zero-shot learning or unsupervised domain adaptation setting. Else if $n_t < 10$, it can be called few-shot learning. The situation that only one source domain is available is mostly considered. However, some other researchers consider the situation that data from multi-source domains are available in order to train a policy for a specific target domain; this kind of problem is called multi-source domain adaptation. Most of the domain adaptation methods try to map data from the source and target domain to a shared latent space based on a metric of the distance between the distribution of source and target feature mapping, which we

call them metric based domain adaptation. Other methods try to find a mapping function that can map data from target domain to source domain; this kind of methods is called reconstruction based domain adaptation. There are also some methods dealing domain adaptation problem with self-training, which we call it self-training based domain adaptation. Furthermore, based on different $\mathscr{F}(\cdot)$, it can also be categorized as domain adaptation for image classification, domain adaptation for semantic segmentation and domain adaptation for object detection.

Simulation to reality domain adaptation aims at bridging the gap between simulations and the reality, e.g. the reality gap. Recent researches divide reality gap into two groups: instance-level gap and pixel-level gap. The former describes the differences in scenario complexity, instance-level distribution. While the latter describes the differences between the quality or the style of images from different domains. Most recent work [18, 35–37] has focused on the latter kind of gap and recently some researches [20, 38] focused on the former one also draw much attention. Metrics for the distance between different domains is the key to the optimization progress. A-Distance [39] and MMD [14, 15] are used widely in domain adaptation research. MMD is firstly used to find an optimized mapping function in order to decrease the distance between two distribution in the common mapping space, and then it is utilized in CNN structure to decrease the variance of extracted features. A-distance, on the other hand, can not be used directly in the training progress since it is hard to calculate it online. New adversarial methods [17] make significant progress in this area thanks to learning-based discriminator's ability to approximate a metric between distinct domains. We can consider adversarial methods as using A-distance as their metric for domain differences.

Other than improving the metrics for domain adaptation, authors in [18] applied the cycle-consistent loss to the training process to effectively reduced the loss of semantic information during the transferring progress. Authors of [19] observe that the most significant singularity value of feature matrix during the transferring progress is harmful to the discriminability in the target domain. Thus they managed to balance the transferability and discriminability with a BSP loss. These researches indicate that regularization terms and the inclusion of "proper level" of perturbation and randomness are beneficial to domain adaptation.

Domain Randomization is a class of techniques for domain adaptation, and researchers mainly use it for Simulation to reality domain adaptation because they can easily accomplish high-quality augmentation on the render engine. By searching domain invariant feature space using randomization instead of minimizing the distance between different domains, domain randomization transfers knowledge between diverse domains. Traditional augmentation methods like flipping, rotation, blurring are widely used for domain randomization to improve the model's generalization ability. In [40], augmentation policy is learnt for simulation-to-real transfer. The recent development in automatic augmentation methods [41, 42] can also suit the domain randomization problem. In [34], style transfer method [43] is explored to build a style-consistent

model. Unlike the basic training with randomized data pipeline, [44] utilized a Randomized-to-Canonical Adaptation Networks (RCANs), crossed the reality gap by translating randomized images into original canonical versions.

2.1.1 Metric Based Domain Adaptation

Traditional transfer learning methods try to find a mapping function in the raw data space in order to map both X_s and X_t to have the same $\mathscr{P}$. A distance metric:

$$DISTANCE(D_s, D_t) \approx \|\mathscr{P}(\mathscr{X}_s) - \mathscr{P}(\mathscr{X}_t)\| \tag{2.1}$$

is used to find a mapping function ϕ in order to minimize $\|\mathscr{P}(\phi(\mathscr{X}_s)) - \mathscr{P}(\phi(\mathscr{X}_t))\|$. [45] tries to align the marginal distribution of two domains, thus the distance metric it designs a MMD is minimize the mean of two distribution after mapping:

$$D_{MMD} = \left\| \frac{1}{n_1} \sum_{i=1}^{n_1} \phi\left(\mathbf{x}_i\right) - \frac{1}{n_2} \sum_{j=1}^{n_2} \phi\left(\mathbf{x}_j\right) \right\|_{\mathcal{H}} \tag{2.2}$$

STL [46] on the other hand aims to minimize the conditional distribution of two domains. It is hard to measure the distance of two domains' conditional distribution, and STL actually estimate the distance of conditional distribution by the mean of each category's MMD distance (for classification task). And the distance is defined as:

$$D_{STL} = \sum_{c=1}^{C} \left\| \frac{1}{n_1^{(c)}} \sum_{\mathbf{x}_i \in \mathcal{D}_s^{(c)}} \phi\left(\mathbf{x}_i\right) - \frac{1}{n_2^{(c)}} \sum_{\mathbf{x}_j \in \mathcal{D}_t^{(c)}} \phi\left(\mathbf{x}_j\right) \right\|_{\mathcal{H}}^2 \tag{2.3}$$

Where C is the total number of categories in a classification task. $D_s^{(c)}$ and $D_t^{(c)}$ represent data that belongs to category c in responding domain.

JDA [47] tries to minimize both the distance of marginal distribution and conditional distribution between two domains. And the objective function is:

$$D_{JDA} = D_{MMD} + D_{STL} \tag{2.4}$$

Since conditional distribution and marginal distribution is not equally important for many domain adaptation tasks, BDA [48] defines the distance as:

$$D_{BDA} = \lambda D_{MMD} + (1 - \lambda) D_{STL} \tag{2.5}$$

Where λ is used to balance the importance of two kinds of distribution, which is estimated with $\mathscr{A}distance$ [49].

As the development of deep learning, metrics designed for traditional methods are also utilized in CNN structure. Instead of calculating a mapping function by optimization methods, the distance between two domains is added to deep neural network's objective function as a regularization term. Here we use a classification task as an example:

$$\ell = \ell_c\left(\mathcal{D}_s, \mathbf{y}_s\right) + \lambda \ell_A\left(\mathcal{D}_s, \mathcal{D}_t\right) \tag{2.6}$$

where ℓ_c is classification loss and ℓ_A is adaptation loss. λ is used to balance the importance of the two objectives.

MMD distance is commonly used as adaptation loss for deep adaptation methods. DDC [14] applies MMD loss on the last global feature mapping. And [50] extends DDC by applying multi-kernel MMD loss on multiple layers of the CNN feature extractor.

The recent development of GAN [51] also contributes to domain adaptation. Researchers realize that the best adaptation loss is not a human-designed loss to minimize conditional loss, marginal loss or both. Another CNN based discriminator can be used to predict the distance between two domains, and with joint training with the original task network, the discriminator network outperforms all the human-designed adaptation loss [52].

Besides different kinds of adaptation loss, many regularization terms are also designed to increase the adaptation performance. [19] observes that during the transfer progress, feature extractor tend to only focus on the most useful feature in order to obtain better transfer performance, they propose the batch spectre penalization loss to encourage the model to utilize more feature during the transfer progress. [53] proposes a similar idea by placing a batch nuclear-norm maximization on the output matrix.

2.1.2 Reconstruction-based Domain Adaptation

Reconstruction-based methods try to find the mapping function that is able to map data from the target domain to source domain. The mapping is done on the raw data space. These kinds of methods usually follow the [43] structure. [54] digs into the innate reason why cycle loss works, and they conclude that shared common space in the middle layers of the two generators is the key to CycleGAN's success. They further increase the transfer performance by directly using shared layers in the middle of two generators. [55] consider not only the transfer performance but also the diversity of mapped images. These kinds of reconstruction based methods will not be biased to any specific task since their mapping is done on the raw data level. However, these kinds of

methods will also be limited by the fact that they do not strictly limit the mapped data to have some label compared the related source domain data. [18] designs a semantic consistency loss to make sure the mapped data and the original data's semantic segmentation are the same. They find that even with not the semantic consistency loss, their method also works. It is mainly because CNN based feature extractors are naturally biased towards a feature that is about texture, colour. Thus CNN based CycleGAN like the unsupervised image to image translation will naturally remain the structure and semantic information of the original images, which benefits the domain adaptation methods a lot since for most of the domain adaptation tasks, labels are constant as long as the semantic information of transferred data is constant.

2.1.3 Discrepancy-based Domain Adaptation

Self-training based methods generate labelled target domain data with a policy trained on the source domain. And then they try to modify the generated pseudo labels to make them more accurate for the target domain. These kinds of modification are mostly based on unsupervised clustering, which actually can only increase the quality of pseudo labels slightly. However, with repeating the generating, modifying and training progress, again and again, the pseudo labels will become more and more accurate, and policy will become more and more suitable for the target domain. This kind of methods is widely used for object detection domain adaptation for the reason that object detection uses local feature instead of global feature to generate output labels. So all the global feature alignment-based methods will face some challenges in such a scenario. The key differences among self-training based domain adaptation methods are the way they correct the generated pseudo labels. [56] designs a Weak Self-Training framework which uses a Supporting Region-based Reliable Score to filter out the generated labels that have high possibility to be wrong. [57] assumes that a classification network trained on the source domain can be used to correct the object detection network's classification result.

2.1.4 Domain Adaptation for Image Classification and Semantic Segmentation

Image classification learns to categorize images into a related class. Semantic segmentation learns to category each pixel of an image into a related class. Most of the domain adaptation research focuses on classification and semantic segmentation [58]. Usually, methods designed for classification task can also be used for semantic segmentation, since these two tasks both utilize the global feature to generate the final output. Global feature alignment-based methods are dominant in classification domain adaptation and semantic segmentation domain adaptation. [35]

adds two conditioning strategies for the standard adversarial adaptation strategy, the multilinear conditioning that captures the cross-covariance between feature representations and model's output to improve the discriminability, and entropy conditioning that designed to reduce the uncertainty of classification model's output to guarantee the transferability. [59] finds that even though CNN based discriminator has been better than most of the human-designed discriminator, the problem that whether the difference between the source and target domain is dominant by the marginal distribution or conditional distribution still exists and still influences the final transfer performance. Thus they propose a dynamic adversarial adaptation network to dynamic decide whether the discriminator should focus on marginal distribution or conditional distribution.

Besides finding better domain distance metric or methods to stabilize the adversarial adaptation progressresearchers start to consider some new problems. Instead of transferring the network's weight learnt from the source domain to the weight of the same network structure suitable for the target domain, [60] propose to transfer the weight learnt from source domain to the weight of a different network structure suitable for the target domain. They divide the problem into these sub-problems: which part the network is transferable, where the weights will be transfer to and how much of this weight can be transferred. This work can also be considered as solving the Knowledge Distilling problem from transfer learning's perspective. With lots of existing domain adaptation methods, which one is suitable for a specific task and dataset is also a problem. [61] proposes this new problem setting and manages to formulate and solve it with an MMD based method. [62] argues that directly global feature alignment may ignore some task-specific decision boundary's characteristics. Furthermore, their method is based on modifying the decision boundary. They use two networks that are trained in the source domain. With some dropout and different initialization, the networks' output on the target domain will be slightly different. Their adaptation progress is first training the classifiers while feature extractors are fixed in order to maximize the two networks' differences of output, second, the two networks' classifiers are fixed and training their feature extractors to minimize the differences of output. On the first stage, the networks are trained to maximize the differences, and in this way, they are encouraged to explore some unexplored area. Moreover, in the second stage, the quality of feature extractor is improved while fixing the classifier layers.

2.1.5 Domain Adaptation for Object Detection

Unlike image classification and semantic segmentation which are generate output based on global feature, object detection generates lots of bounding boxes, and each of them is based on some local feature. This difference makes the global feature-alignment-based methods that are popular for image classification and semantic segmentation unable to have satisfying results for object

detection. Thus self-training based methods are used widely for object detection domain adaptation. Some feature-alignment-based methods will need to consider how to align local feature instead of directly aligning the global feature.

Instead of aligning the distribution of global feature. [63] filter out the feature that will not influence the final output bounding boxes with the help of two-stage object detector's RoI module. They set a threshold that only the areas higher ?? will be used for distribution alignment. That is to say, only the ideally only the objects need to be aligned and meanwhile, and background is not considered transferable.

[56] uses a Weak Self-Training (WST) module, which will filter out the generated pseudo labels that are likely to be wrong. They design a Supporting Region-based Reliable Score (SRRS) which decides how likely the pseudo labels are correct. The decision is made considering all the nearby RoI instead of just one.

$$SRRS\left(r^{*}\right)=\frac{1}{N_{s}} \sum_{i=1}^{N_{s}} IoU\left(r_{i}, r^{*}\right) \cdot P\left(c^{*} \mid r_{i}\right) \tag{2.7}$$

where N_s is the number of nearby supporting RoI, $IoU(A;B)$ is the IoU between region A and region B, c^* is the pseudo label of r^*, and $P\left(c^{*} \mid r_{i}\right)$ is the probability that r_i belongs to c^*.

[57] trains an extra image classifier using source domain labelled bounding boxes and responding labels. Furthermore, after generating pseudo labels in the target domain, this classifier is used to correct the pseudo bounding boxes' predicted label. This method bases on the assumption that image classifier usually is more reliable among different domains. Furthermore, the separate image classifier can be trained with a far more complex network backbone and maybe semi-supervised training strategy [64] which can also make sure that it can provide more reliable predictions compared to object detection networks' built-in classification head.

2.2 Domain Adaptation for Point Cloud

Domain adaptation for point cloud data is an unexplored area. Here we will mostly introduce some object detection methods designed for point cloud data. Also, some point cloud data's unique characteristic compared to RGB images' will be included.

There have been lots of research on RGB image semantic segmentation and object detection. Although extracting information from RGB images and point cloud data is similar in some degree, there are two key differences: 1) point cloud a disordered which means feature extractors designed for point cloud should have similar output when the order if input data changes. 2)

point cloud data is extremely sparse. For RGB images, almost all the pixels in an image can be considered meaningful for a feature extractor, while for a frame of bird's eye view image generated from point cloud data, only part of data is scanned with Lidar while most of the data is blank and meaningless. Figure 2.1 [1]shows a RGB image with 2D and 3D annotations and its corresponding point cloud data.

Figure 2.2 [1] shows point cloud data and its corresponding bird's eye view image. All the black area is not scanned and is considered as meaningless.

Early works [65] try to use 3D convolution to deal with point cloud object detection. Some other works consider projecting point cloud data into 2D images and then extracting information from projected 2D image [66]. Following works process point cloud from a bird's eye view [67]. Recently, voxel-based methods achieve good performance mainly because voxels, to some degree, make the data less sparse. From manual designed feature encoding [68] to automatic voxel-level feature extractor [69], quality of generated voxels increases a lot and the final performance of object detection or semantic segmentation also increase correspondingly. Generated voxels [69] are usually 3D which still require 3D convolutions to extract information. [70] proposes an efficient voxel generation method and can obtain good object detection performance with only 2D voxels and much faster compared to 3D voxels based methods.

Although point cloud is extremely sparse compared to RGB image, it has the advantage that almost all the scanned points belong to objects that the object detection methods try to find (For autonomous driving, most of the points belong to road, vehicle, pedestrians), which makes the manual designed RoI module possible [71].

For RGB images, object detection methods are categorized as one-stage or two-stage. Two-stage methods like [72] first generate region proposals and then classify and refine the proposals. One-stage methods like [73] and [74] do all the region proposals generation, classification and refinement in a single stage, which make it faster than two-stage methods.

[1]From KITTI Object data transformation and visualization project. https://github.com/kuixu/kitti_object_vis

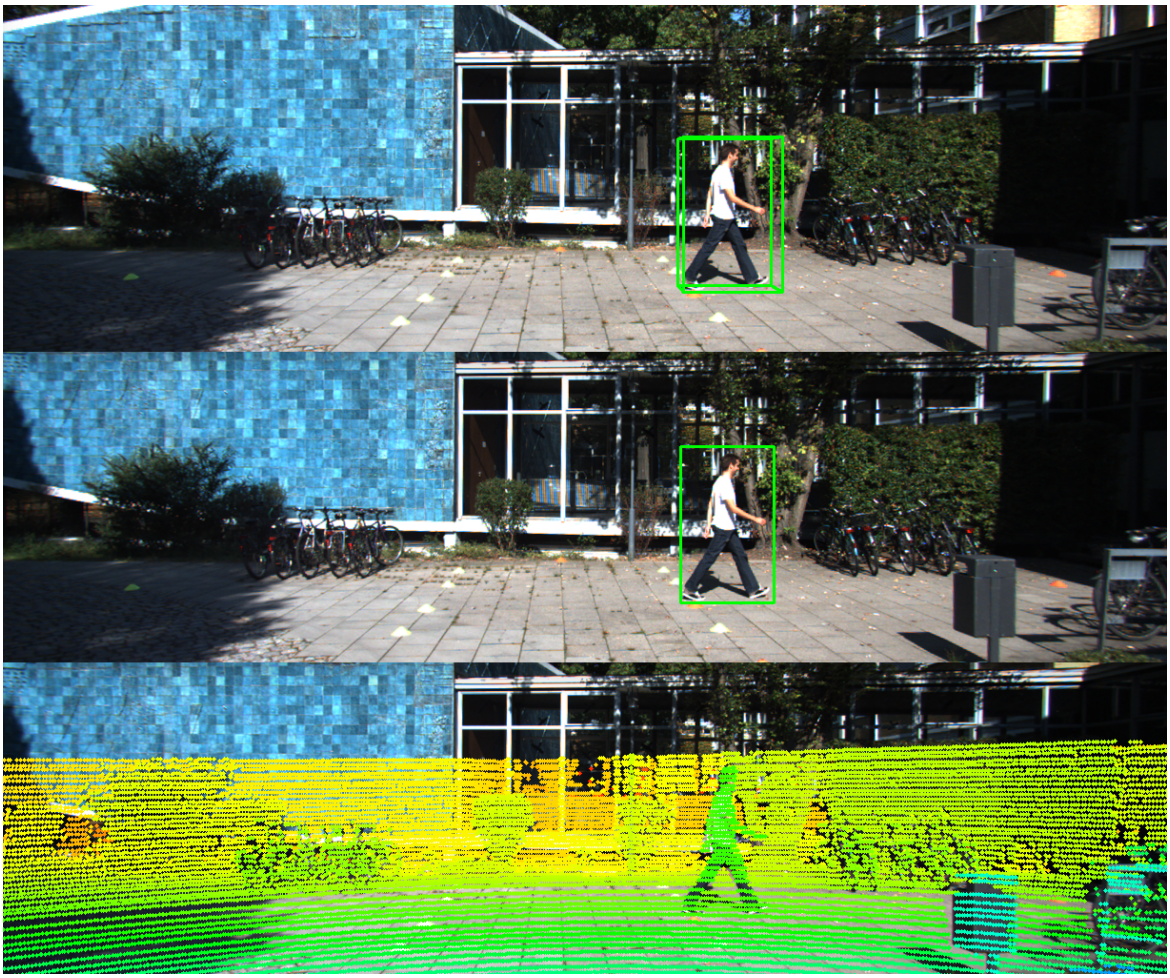

Figure 2.1: Point Cloud Data Mapped to Related RGB image. From KITTI Object data transformation and visualization project.

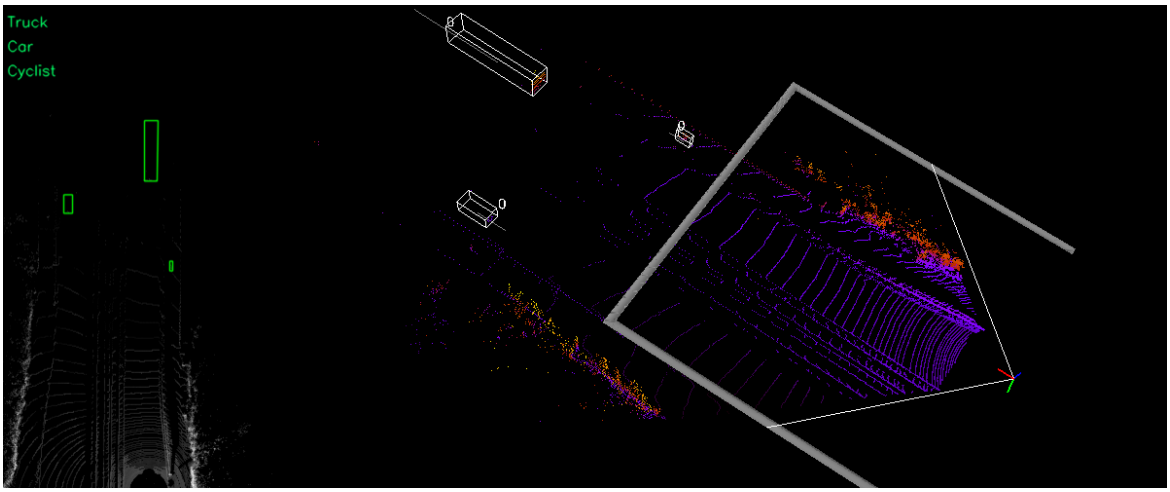

Figure 2.2: Point Cloud data and its corresponding bird's eye view.

Chapter 3

Methods

3.1 Domain Adaptation for Image Semantic Segmentation

3.1.1 Metric Guided Domain Randomization

The general framework for metric guided domain randomization is presented here. Consider having access to source images $\mathcal{X}_s$ and labels $\mathcal{Y}_s$ drown from a source domain distribution $p_s(x, y)$. In the sense of unsupervised adaptation, only limited unlabelled target images $\mathcal{X}_t$ are accessible from target domain distribution $p_t(x, y)$ where there is no label but domain adaptation aims to learn a task mapping function M to transfer knowledge. [17] assumes they have direct access to the target images. However, the limited target images often lead to model bias to the target domain distribution $p_t(x, y)$. Here this thesis considers a different setting by adding random perturbations to approximate the target distribution. In order to consider the randomness of the perturbations, the thesis denotes the noise set as $\mathcal{N}$ and specific noise $n \in \mathcal{N}$. The primary goal of our proposed framework is to randomize and augment the source mapping in order to make sure $M(\mathcal{X}_t, n)$ can not be distinguished from the demonstrated task in source domain with a collection of randomness $\{M(\mathcal{X}_s, n_i)\}_{i=1}^{k}$, where k is the size of the selected noise set. In this case, the source classifier C_s can be directly applied to the target domain, such that unified classifier $C = C_s = C_t$.

The supervised loss for the source classification model can be defined as:

$$\mathcal{L}_{\text{cls}}\left(\mathcal{X}_s, \mathcal{Y}_s, \mathcal{N}\right) = \mathbb{E}_{(x_s, y_s, n)\sim(\mathcal{X}_s, \mathcal{Y}_s, \mathcal{N})} - \sum_{k=1}^{K} \mathbb{1}_{[k=y_s]} \log C\left(M\left(\mathcal{X}_s, n\right)\right) \tag{3.1}$$

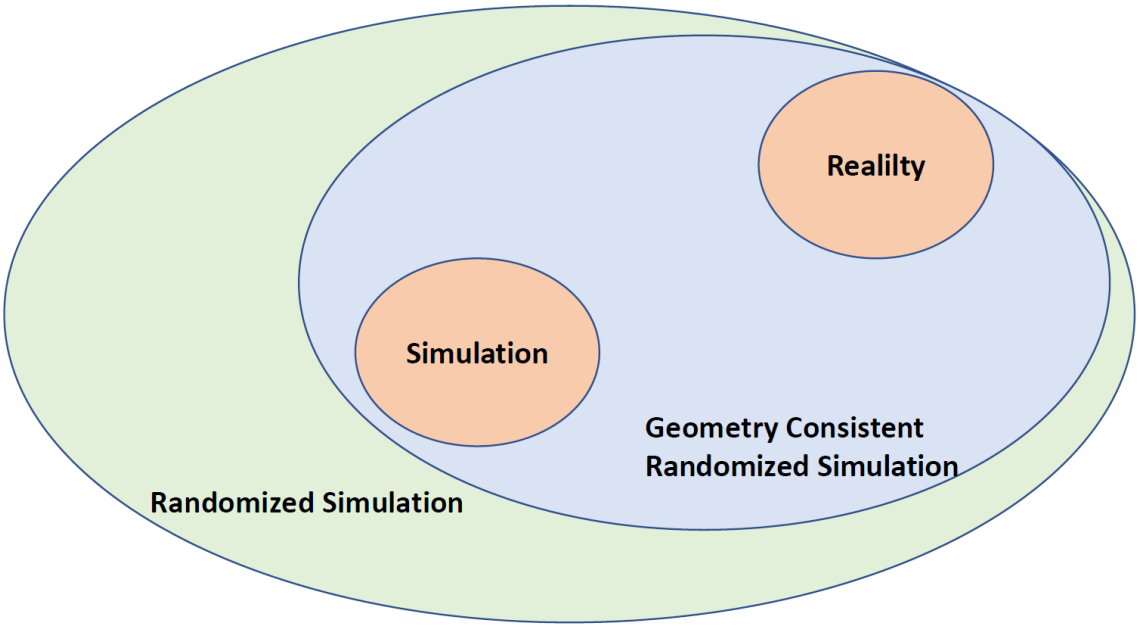

Figure 3.1: Relation between images from randomized simulation, simulation and reality

To make sure the selected perturbation noise $\mathcal{N}$ can effectively randomize the source domain mapping $M(\mathcal{X}_s, \mathcal{N})$ instead of being ignored, the proposed method presents noise classifier C_n with the supervised loss $\mathcal{L}_n(\mathcal{N}, M)$ as

$$\mathcal{L}_n(\mathcal{X}_s, \mathcal{N}, M) = \mathbb{E}_{(x_s,n)\sim(\mathcal{X}_s,\mathcal{N})} - \sum_{k=1}^{K} \mathbb{1}_{[k=n]} \log C_n\left(M\left(x_s, n\right)\right) \tag{3.2}$$

Finally, the proposed method needs to make sure the task mapping in the target domain with arbitrary perturbations $M(\mathcal{X}_t, \widetilde{n})$ can not be distinguished by a domain discriminator D, from the task model accepting the source domain input $M(\mathcal{X}_s, \mathcal{N})$ for the transfer learning robustness. The supervised loss for discriminability $\mathcal{L}_D$ is defined as

$$\begin{aligned} \mathcal{L}_D(\mathcal{X}_s, \mathcal{X}_t, M, \mathcal{N}) = &- \mathbb{E}_{(x_s,n)\sim(\mathcal{X}_s,\mathcal{N})}[\log D\left(M\left(x_s, n\right)\right)] \\ &- \mathbb{E}_{x_t\sim\mathcal{X}_t}[\log(1 - D\left(M\left(x_t, \widetilde{n}\right)\right))] \end{aligned} \tag{3.3}$$

where $\widetilde{n}$ is a specific noise.

The generic formulation for the metric guided domain randomization considering unknown

perturbations is derived below:

$$\begin{aligned} &\min_{M,C} \mathcal{L}_{\text{cls}}\left(\mathcal{X}_s, \mathcal{Y}_s, \mathcal{N}\right) \\ &\min_{M,C_n} \mathcal{L}_n(\mathcal{X}_s, \mathcal{N}, M) \\ &\min_{M} \max_{D} \mathcal{L}_D(\mathcal{X}_s, \mathcal{X}_t, M, \mathcal{N}) \end{aligned} \tag{3.4}$$

The proposed formulation of a metric guided domain randomization problem can be considered as a simultaneous optimization problem. Notably, the direction of randomness $\mathcal{N}$ included will have an impact on the efficiency of the domain transfer. Several problems remain for finding the solution of the optimization problem (3.4).

- How to find "proper" noise to the feature mapping?
- Should the introduced perturbation be the features from the target domain data, data similar to target data or just random noise?
- In terms of the training process, should we divide the whole pipeline into metric based domain randomization and training with randomized data or one-staged adversarial training progress?

These issues will be discussed in the following sections along with the proposed solution and designed model.

Randomization Based Domain Generalization

The benefit of our proposed generic framework for domain randomization is that it directly applies to the development of adaptive learning methods. In many applications, including autonomous driving and robotic vision, semantic segmentation is a fundamental visual task towards the visual understanding of the environment. Source images and labels can be obtained from the public datasets as well as simulation platforms. Typically, having access to target domain data distribution will benefit randomization progress. This thesis argues that have direct access to target dataset will make the trained model bias to the limited target sample distribution and decrease the generalization performance, since, for domain adaptation, we usually have limited target data obtainable. Instead of having direct access to the target data, the proposed method encodes a pure prior knowledge into our model. The proposed method mainly uses the prior knowledge that the simulation environment and real-world environment differ a lot in its texture, lighting, but similar in their geometric information. Hence a geometry consistent domain randomization model

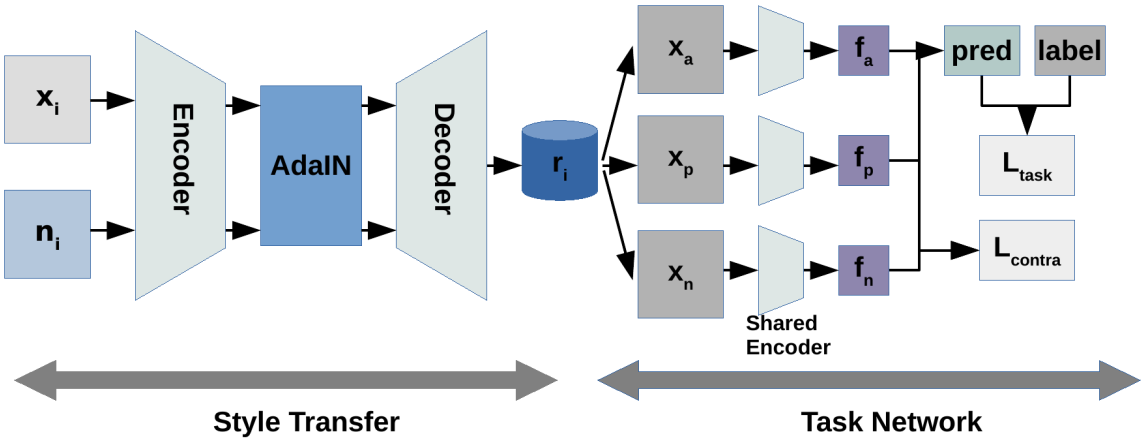

Figure 3.2: An overview of our proposed metric guided domain randomization pipeline. A style transfer network is used to generate random stylized images with noise $n_i \in N$. The generated stylized images is then labeled as positive x_p or negative x_n based on the comparison between the anchor image x_a. A contrastive loss is utilized in the task policy training process to make sure feature mappings of positive images are similar with the anchor images while feature mappings of negative images are dissimilar.

will first be used to randomize the source data. Furthermore, the proposed method uses a triplet structure to make sure the task model mainly learn from geometry consistent information.

To tackle the union optimization problem in (3.4), the thesis proposes a two-fold pipeline with style transfer and triplet network trained by contrastive learning. The style transfer stage in Fig. 1.1 handles (3.2) by implementing style transfer methods for accurate randomization [34]. The noise set $\mathcal{N}$ is a set of images from a weakly correlated domain. The adaptive instance normalization [75] is used to add noise to middle state feature mapping. Notice that the level of perturbation need to be controlled since its ideal to maintain the minimal level of content information from the source image instead of the pursue of total randomness. Hence, the proposed method adopts a style loss of $\mathcal{L}_n(N, M)$ as (3.2) in our pipeline to ensure that the noise can still be distinguished.

The discriminability loss (3.3) is a typical min-max two-player problem where a GAN model is typically accepted in [17]. Although GAN should have excellent performance in the Nash equilibrium, only in the case of convex function can gradient descent guarantee Nash equilibrium. When neural networks represent both sides of a game, it is possible for them to always adjust their strategies without actually reaching equilibrium. Another point is that the learning process of GAN often suffer from a collapse problem where the generator starts to degenerate.

Considering these problems, the thesis proposes a simple-to-implement supervised training pipeline to handle (3.1) and (3.3) simultaneously. The task network stage illustrated in Fig. 1.1 take advantage of a triplet structure to filter the augmentation with "proper" perturbations. The thesis argues that the discriminability can be measured by the consistency of the semantic segmentation results. The proposed method uses geometric consistency as an approximation measure of the model discriminability in (3.3). Moreover, the thesis uses a triplet network to tell whether the source image with injected perturbation $x_s \oplus n_i$ belongs to the source domain distribution, e.g. classify whether $x_s \oplus n_i \sim p_s(x, y)$ is real. The proposed triplet structure is advantageous in making the task model robust on the consistent geometric information.

The relation between simulation, randomized simulation and reality is shown in Figure 3.1. Given $X_{simulation} \subset R^{3\times H\times W}$, $X_{reality} \subset R^{3\times H\times W}$, the proposed method tries to find a mapping $X_{randomized} = f(X_{simulation}, s)$ given a specific style s, and make sure that $X_{reality} \subset X_{randomized}$. Also the proposed method tries to find a mapping $X_{simulation} = G(X_{randomized})$. Since $X_{reality} \subset X_{randomized}$, the proposed method will also get $X_{simulation} = G(X_{reality})$. This mapping function G will help us map a real image into $X_{simulation}$ where unlimited high quality labeled data and a policy trained with these data have already been accessible. For a unsupervised domain adaptation setting, the problem can be formalized as:

$$\theta_f^* = \arg\min_{\theta_f} Dist(f(X_{simulation}), X_{reality}) \tag{3.5}$$

And for a domain generalization problem setting, it can be formalized as:

$$\theta_f^* = \arg\min_{\theta_f} Dist(f(X_{simulation}), f(X_{simulation}, s)) \tag{3.6}$$

where s is a specified style which is **never** used by f in the training progress.

There is no guarantee that information that we are interested in keeps unchanged during the randomization progress, while the proposed method is trying to keep the information during the randomization progress explicitly, it is also important that the proposed method does not randomize $X_{simulation}$ too much.

Algorithm 1: AdaIN with $\mathcal{L}_1$ loss

Data: content image I_c, style image I_s, encoder ϕ, ith activation layer of the encoder ϕ_i, decoder g

initialization;

for *number of training iterations* **do**

Calculate feature mapping of both I_c and style image I_s. Referred to as M_c and M_s

Align data distribution of M_c and M_s: $t = \sigma(M_s)\left(\frac{M_c - \mu(M_c)}{\sigma(M_c)}\right) + \mu(M_s)$.

Decode the aligned features back to image space.

Encode the reconstructed image and calculate content loss and style loss:

$\mathcal{L}_s = \sum_{i=1}^{L} \|\mu\left(\phi_i(g(t))\right) - \mu\left(\phi_i(I_s)\right)\|_2 + \sum_{i=1}^{L} \|\sigma\left(\phi_i(g(t))\right) - \sigma\left(\phi_i(I_s)\right)\|_2$

$\mathcal{L}_c = \|f(g(t)) - t\|_2$

$\mathcal{L}_1 = \|g(t) - I_c\|_1$

Update weights of the decoder based on: $\mathcal{L} = \mathcal{L}_c + \lambda_1 \mathcal{L}_s + \lambda_2 \mathcal{L}_1$.

end

Style Transfer for Domain Randomization

The proposed method needs to add random noise to source images; meanwhile, make sure the randomized image and origin image are geometry consistent. Experiments show that add direct pixel-wise constraint like $\mathcal{L}_1$ loss in the training progress will violate equation (3.2) and resulted in the perturbations indistinguishable from randomized images. Hence, the proposed method uses a perceptual metric [76] to measure the distance between the source images and randomized images.

First, the proposed method encodes both content image c and style image s to feature space with a fixed encoder f. An adaptive instance normalization layer is then used to align the mean

and variance of the content feature maps to those of the style feature maps:

$$t = \text{AdaIN}(f(c), f(s)) \tag{3.7}$$

A decoder g is used to map the target feature map back to target image space.

$$T(c, s) = \text{g}(t) \tag{3.8}$$

The total training loss is:

$$\theta_f^* = \mathcal{L}_s + \lambda_1 \mathcal{L}_c \tag{3.9}$$

$$\begin{aligned} \mathcal{L}_s = &\sum_{i=1}^{L} \|\mu\left(\phi_i(g(t))\right) - \mu\left(\phi_i(s)\right)\|_2 + \\ &\sum_{i=1}^{L} \|\sigma\left(\phi_i(g(t))\right) - \sigma\left(\phi_i(s)\right)\|_2 \end{aligned} \tag{3.10}$$

$$\mathcal{L}_c = \|f(g(t)) - t\|_2 \tag{3.11}$$

where ϕ_i denotes activation layers of f. While the adaptive normalization operation align noise into intermediate features, $\mathcal{L}_s$ is used to make sure n is still noticeable after reconstruction, as is described in equation (3.2).

Contrastive learning

Our task network needs to learn from $M(x_s, n)$ and control the effects of n. It can be considered as a contrastive learning problem and the proposed contrastive learning module tries to learn from the differences between different generated images. In this section, the thesis explores the use of triplet network and siamese network [77]. The proposed method generates different images of the same source image with different style images, and every two of the images can be considered as a positive pair. Since GTA5 dataset is collected by time order, the proposed method chooses images that have close indices, stylize them with the same style image and then uses these as negative pair. The proposed method adds a triplet ranking loss to make sure our intermediate feature representation focuses on geometry information instead of image texture. For an anchor image x_a, positive image x_p has the same geometry information with x_a and negative image x_n has similar texture information with the anchor image. The triplet ranking loss is:

$$\mathcal{L}_{contrastive}(x_a, x_p, x_n) = \max(0, m + \|f(x_a) - f(x_p)\| - \|f(x_a) - f(x_n)\|) \tag{3.12}$$

Figure 3.3: Example of triplets for constrastive learning. Anchor image is filtered with a perceptual similarity metric. Positive image is generated with the same content as the anchor image but stylized with different style image. Negative image is generated with similar content image as the anchor image and stylized with the same style image as the anchor image.

A pairwise ranking loss can also used:

$$\mathcal{L}_{contrastive}(x_a, x_p, x_n) = \begin{cases} \|f(x_a) - f(x_p)\| & \text{if } (x_a, x_p) \text{ is positive} \\ \max(0, m - \|x_a - x_n\|) & \text{if } (x_a, x_n) \text{ is negative} \end{cases} \tag{3.13}$$

Another choice of training objective is:

$$\mathcal{L}_{task}(x_a, x_p, x_n) = \mathcal{L}_{seg}(x_a, y_a) + \lambda \mathcal{L}_{contrastive} \tag{3.14}$$

where f is a feature mapping function of our task network. The proposed method adds a spatial pyramid pooling layer before calculating the distance between two feature representation, as mentioned in [34]. Selection of negative pairs is the key to contrastive learning. The experiments showed that it is hard to always select hard negative examples in order to provide a consistent gradient. Noise contrastive estimation (NCE) [78] combines triplet loss and softmax and can be used in such a situation. However, NCE requires the storage of thousands of negative pairs; hence the proposed method only uses the positive pairs. In this case the pairwise ranking loss (3.13) is equivalent to (3.12).

3.1.2 Style-agnostic Target Domain to Source Domain Generator

A CGAN based domain adaptation method is also proposed. Unlike the Metric Guided Domain Randomization method, the CGAN based method trains directly with randomized source domain images. The CGAN based methods trains a Style-agnostic Target Domain to Source Domain Generator $G_{t->s}$. During inference, target domain images are first passed to $G_{t->s}$ to generate a corresponding source domain image, and then the generated source images can be passed to a semantic segmentation model trained with source domain data. The whole pipeline is shown in Figure 3.4

The critical component of this pipeline is the CGAN which is used to find a mapping function that can map data from the target domain to source domain. In a typical unsupervised domain adaptation setting, this can be done by training a CGAN based on X_s and X_t. However, in this thesis, the proposed method considers a harder situation where the model has no access to X_t. All we get is a prior knowledge that the source domain and target domain data are styles agnostic. With no access to target domain data, the proposed method needs to find a mapping function that can map from the randomized source domain to source domain and also make sure that target domain can not be distinguished from the randomized source domain.

CGAN [79] takes n randomized images $\tilde{x}$ of the same original image x as input, and the fake original image y as output. Since all the randomized images in a training batch are mapping to the same output, the similarity betwee the input images' feature mapping can be used as a regularization term. The objective of the conditional GAN can be denoted as:

$$\begin{aligned}\mathcal{L}_{cGAN}(G, D) =& \mathbb{E}_{\tilde{x},x}[\log D(\tilde{x}, x)] + \\ & \mathbb{E}_{\tilde{x},z}[\log(1 - D(\tilde{x}, G(\tilde{x}, z))]\end{aligned} \tag{3.15}$$

spatial pyramid pooling loss [34] is used to encourage the consistency of different feature mapping of images in the same batch. And $\mathcal{L}_1$ loss is used to encourage less blurring. Overall, our objective can be expressed as:

$$G^* = \arg\min_G \max_D \mathcal{L}_{cGAN}(G, D) + \lambda_1 \mathcal{L}_{spp} + \lambda_2 \mathcal{L}_1 \tag{3.16}$$

In practice, the proposed method uses patch wise loss in the discriminator to focus on discriminating high frequency differences as mentioned in [79]. Specific training process is shown in Algorithm 2

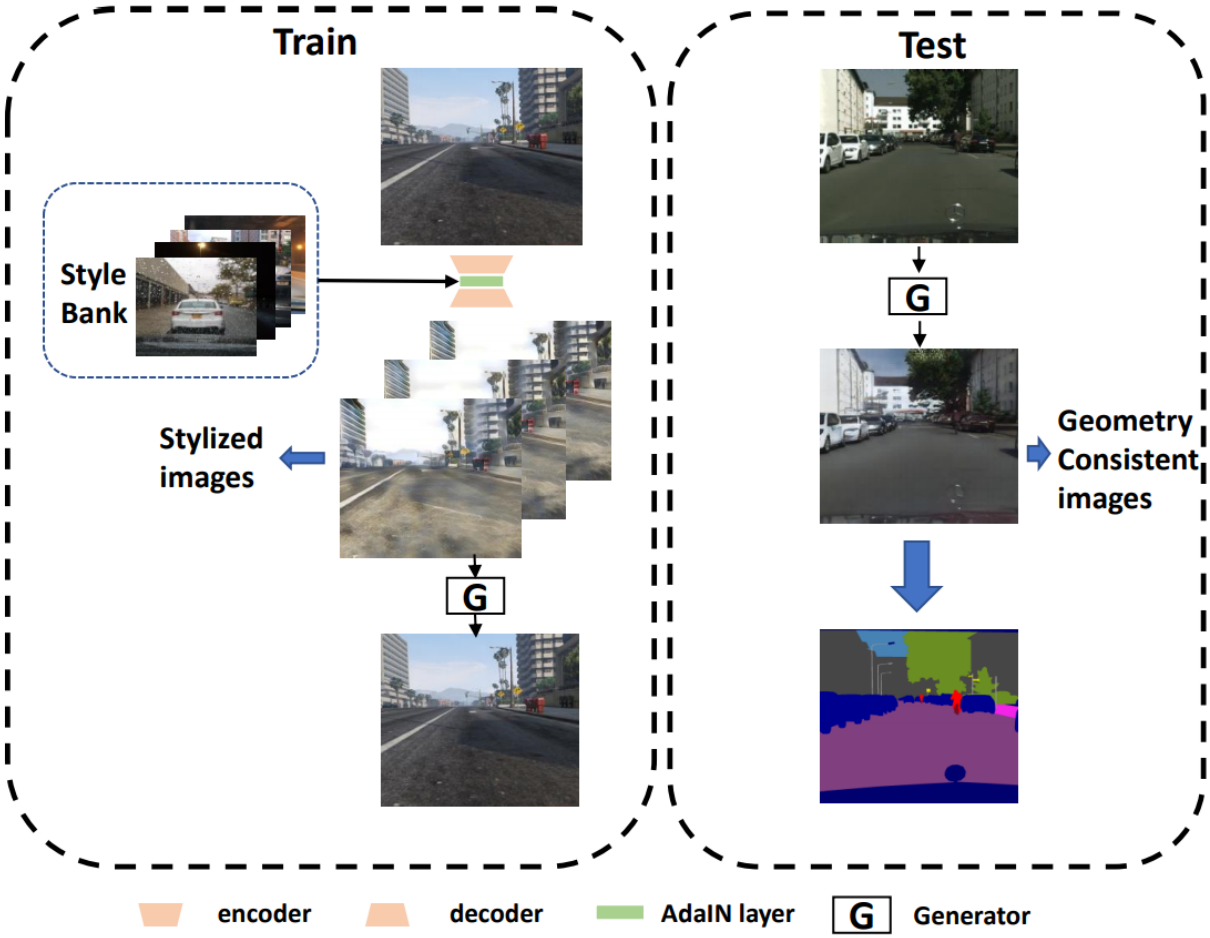

Figure 3.4: Style-agnostic Target Domain to Source Domain Generator Pipeline

3.2 Domain Adaptation for Point Cloud Object Detection

3.2.1 Adaptive PointPillars

Point Pillars

This section is a brief introduction of original PointPillars [70]. For autonomous driving, inference speed of perception module should be over 30 fps. Only projection-based methods can be that fast while still relatively accurate. GCN-based methods' inference speed is mostly less than five fps, and Point-based methods' inference speed is usually 10-20 fps. PointPillars is an end-

Algorithm 2: Conditional GAN

Data: Original image I_o, Randomized image I_r
initialization;
for *number of training iterations* **do**
  Generate fake original image $\tilde{I}_r$ from corresponded randomized image I_r.
  Discriminate $\tilde{I}_r$ given I_r, calculate L_1 loss between I_o and I_r.
  Update weights of the discriminator by gradient by gradient ascend:
  $\nabla_{\theta_d}[\mathbb{E}_{I_r,I_o}[\log D(I_r, I_o)] + \mathbb{E}_{I_r,z}[\log(1 - D(I_r, \tilde{I}_r)] + \lambda L_1]$
  Update weights of the generator by gradient descent:
  $\nabla_{\theta_g}[\mathbb{E}_{I_r,I_o}[\log D(I_r, I_o)] + \mathbb{E}_{I_r,z}[\log(1 - D(I_r, \tilde{I}_r)]$
end

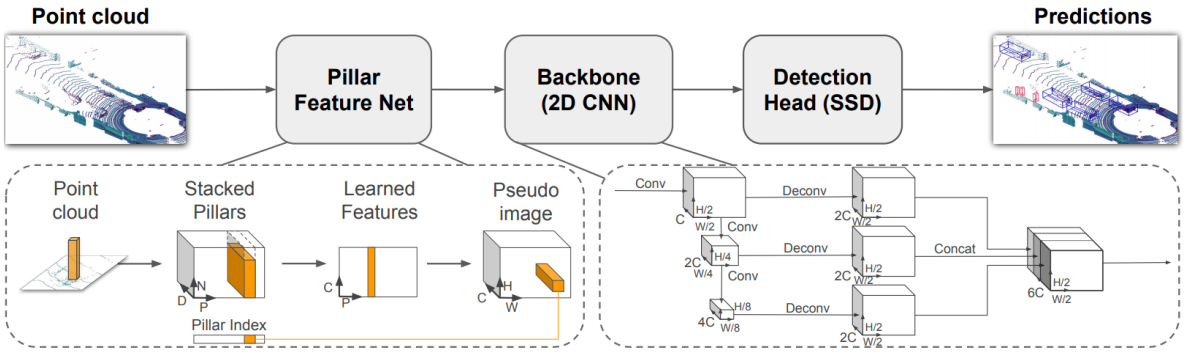

Figure 3.5: PointPillars Pipeline

to-end projection-based 3D object detection network that achieves state-of-the-art performance. It uses a one-stage object detection method SSD and a novel pillars extraction module that makes it faster than most other methods. It can infer at 62Hz.

The key contribution of PointPillars is its PointCloud to Pseudo-Image extraction module. A point in the point cloud is l with coordinates x, y, z and a reflectance r. Instead of dividing the whole point cloud into 3D grids, PointPillars divides the point cloud into a grid in the x-y plane and results in pillars $\mathscr{P}$ with $|\mathscr{P}| = B$. Then the original coordinates are converted into pillar-relative coordinates x_c, y_c, z_c, x_p and y_p. Here c subscript means distance is calculated against the mean of all points a the same pillar, and p subscript means distance is calculated against the centre of the pillar. The converted point will have $D = 9$ dimensions.

And then, the generated pseudo-images are used to train a Single Shot MultiBox Detector (SSD). The whole pipeline is shown in 3.5.

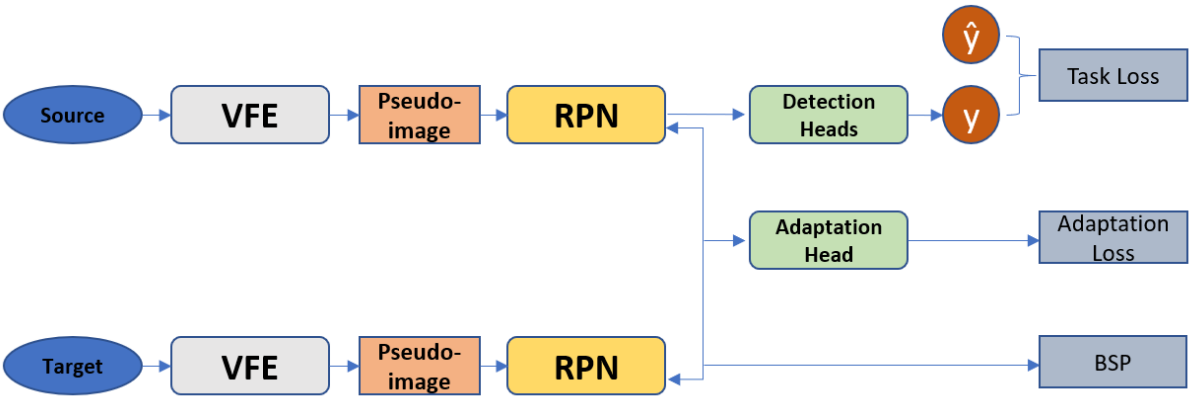

Figure 3.6: Our Adaptive PointPillars Structure

Adaptive PointPillars

The section describes how the proposed method manages to adapt a PointPillars network trained in the source domain to the target domain.

The proposed method uses the typical metric based domain adaptation structure, as shown in the left part of 1.1. In a metric based domain adaptation method, the bottleneck of the adaptation performance are: first, where is the adaptation head placed; second, what kind of adaptation loss can be used.

To address the first issue, multiple locations are tested and finally the proposed method places the adaptation head after the 2D CNN backbone and the other detection heads. In this layer, the feature is less sparse than the original point cloud and the extracted pseudo-images.

And to address the second issue, the proposed method tests three different adaptation loss, MMD [14], CORAL loss [80] and RevGrad loss [81].

MMD loss minimizes the distance between mean of two data distribution with respect to a particular representation ϕ. It can be represented as:

$$\begin{aligned} &\mathrm{MMD}\left(X_S, X_T\right) = \\ &\left\| \frac{1}{|X_S|} \sum_{x_s \in X_S} \phi\left(x_s\right) - \frac{1}{|X_T|} \sum_{x_t \in X_T} \phi\left(x_t\right) \right\| \end{aligned} \tag{3.17}$$

where ϕ is a CNN based feature extractor.

CORAL loss is the distance between the second-order statistics (covariances) of the source

and target features with respect to a specific representation ϕ:

$$\ell_{CORAL} = \frac{1}{4d^2} \|\phi(X_s) - \phi(X_t)\|_F^2 \tag{3.18}$$

RevGrad loss uses a discriminator network as distance metric and update parameters based on:

$$\begin{aligned} \theta_f &\leftarrow \theta_f - \mu \left(\frac{\partial \mathcal{L}_y^i}{\partial \theta_f} - \lambda \frac{\partial \mathcal{L}_d^i}{\partial \theta_f} \right) \\ \theta_y &\leftarrow \theta_y - \mu \frac{\partial \mathcal{L}_y^i}{\partial \theta_y} \\ \theta_d &\longleftarrow \frac{\partial \mathcal{L}_d^i}{\partial \theta_d} \end{aligned} \tag{3.19}$$

where l_d is domain adaptation loss and l_y is task loss.

PyTorch implement of all these three losses will be provided in Appendix A.

And as mentioned in 2.2, domain adaptation for object detection should not only focus on global feature, so the proposed method uses a BSP [19] to encourage the feature extractor to use more feature instead of focusing on the most important few principal components. With this penalization, the adaptation progress will not only focus on global feature alignment but pay more attention to task specific decision boundary. BSP loss bases on the observation that during domain adaptation the largest singular value of the feature matrix that adaptation loss is placed on is significantly larger than the others, which will increase the transferability of model but at the same time decrease the discriminability. To reduce increase discriminability on target domain, they propose a BSP:

$$L_{\text{bsp}}(F) = \sum_{i=1}^{k} \left(\sigma_{s,i}^2 + \sigma_{t,i}^2 \right) \tag{3.20}$$

BSP is easy to understand, it is the sum of largest k singularity value of feature matrix (both source and target domain). And with this penalization, performance on target domain is actually increased.

The whole structure of our Adaptive PointPillars in shown in Figure 3.6

Experiments and comparison of different choice of adaptation loss is shown in section 4.2.1

Chapter 4

Experiments

4.1 Domain Adaptation for Image Semantic Segmentation

4.1.1 Metric Guided Domain Randomization

In this section, the thesis describes the experiment set up and list results of the proposed metric guided domain randomization method on the semantic segmentation task. This section also provides analysis of different experiment set up and comparison with other methods. It is worth mentioning that the proposed policy transfer progress does not require any target domain data or data relevant to the target domain. All it need is a prior knowledge that the current simulation environment is different from the real-world environment in their texture, lighting, physics, but similar in their geometry information. The proposed method uses this strict experiment set up to guarantee the generalization ability of our proposed method.

Experiment Settings

The experiments use FCN8s-VGG16 [82] as task policy. The proposed method evaluates the semantic segmentation performance of the model trained on one synthesized dataset on different real-world datasets.

Datasets The experiments choose GTA5 dataset [83] as our synthesized dataset, Cityscapes [84] and BDD100K [85] as our real-world dataset. The experiments use Kaggle's Painter by Numbers data set [1] as our noise set.

[1] Painter by Numbers dataset. https://www.kaggle.com/c/painter-by-numbers/data

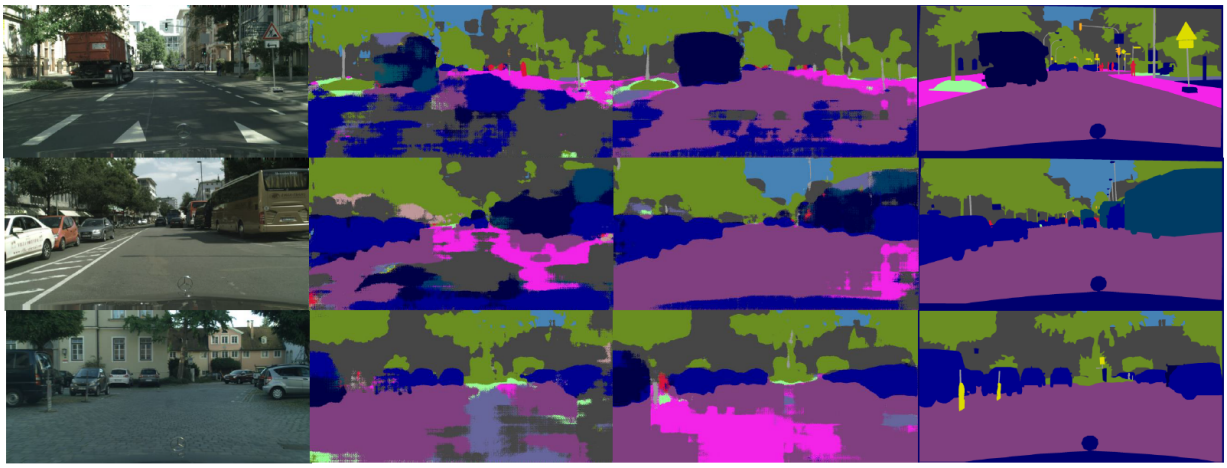

Figure 4.1: Example of CityScapes semantic segmentation results. The first column is original RGB image, the second column is result of FCN8s-VGG16 model trained directly on GTA5 dataset, the third column is result of our proposed method and the last column is the ground truth.

GTA5 is a vehicle-egocentric dataset collected from a computer game. The authors extract pixel-wise semantic segmentation labels from render engine directly. Therefore, their semantic segmentation labels are cheap but extremely accurate. They provide a total of $24,966$ images with resolution 1914×1052. The class labels are compatible with the CityScapes datasets.

CityScapes contains 5000 fine annotated frames and 20000 coarsely annotated frames. Researchers collect data on the urban environment from 50 different cities in Europe. The resolution is 2048×1024, and they provide a total of 30 classes. The experiments only use 19 of the classes.

BDD100K is collected with a dashcam. They provide a total of over $10,000$ diverse images with pixel-level and rich instance-level annotations. Resolution of images is 1280×720.

Training The experiment trains an AdaIN network using the training set of GTA5 dataset as content images and images from Painter by Numbers dataset as style images. The proposed method augments the training set of GTA5 with trained AdaIN model and randomly selected style images from Painter by Numbers dataset. For one source domain image from GTA5 dataset, the proposed method generates two positive images as a positive pair. All the generated images are filtered by a perceptual loss with a constant threshold. The proposed method uses FCN8s-VGG16 as base model. The input of the semantic segmentation network is pairs of images consist of two generated images of the same source image. For all the experiments train all the semantic segmentation models with images of a resolution of 512×1024. Contrastive loss is calculated on different activation layers of FCN8s.

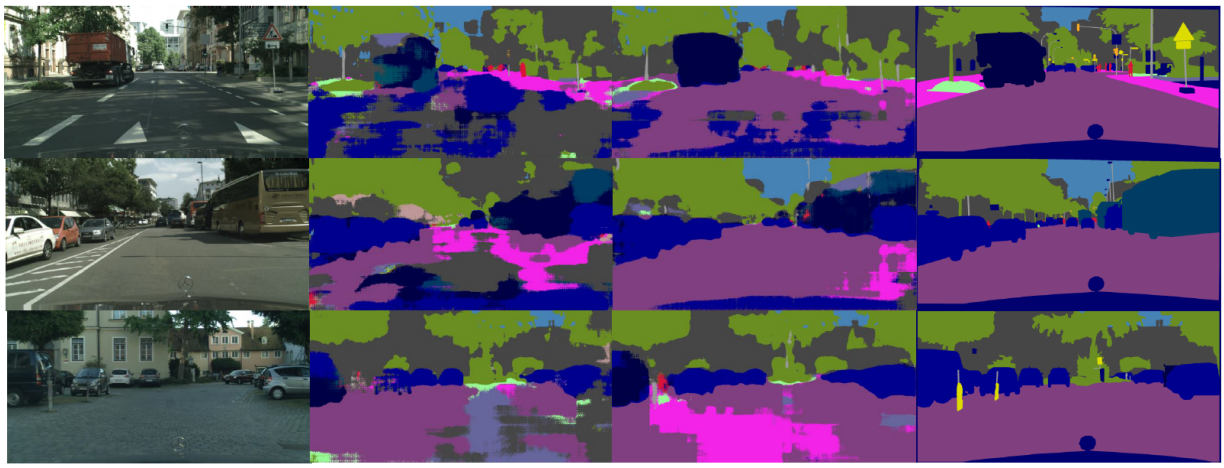

Figure 4.2: Example of BDD100K semantic segmentation results. The first column is original RGB image, the second column is result of FCN8s-VGG16 model trained directly on GTA5 dataset, the third column is result of our proposed method and the last column is the ground truth.

Evaluation This section follows the evaluation protocol mentioned in [86]. The real-world RGB image is resized to 512×1024 and then passed through to the model trained with synthesized images. For evaluation the output semantic predictions are resized to the original resolution and evaluated with the standard Pascal VOC intersection-over-union (IoU). The evaluation uses mIoU as evaluation metric. The model trained on augmented GTA5 dataset is tested with CityScapes dataset, and the result is shown in Table 4.1.

Results Discussion Effect of resolution is discussed in previous articles [88]. For metric based domain adaptation methods, high resolution usually leads to better performance in both the source and target domain. In our experiment, high resolution will lead to higher source (randomized) domain performance, but target domain performance does not increase. For example, the proposed method resizes the image from GTA5 to a resolution of 512×1024 and then random crop a 512×512 part. With lower resolution, the proposed method can still get a target domain mIoU (FCN8s-VGG16) of 27.4 (mIoU is 27.1 for 512×1024) while source domain semantic segmentation performance is worse compared to the performance of the higher resolution model (34.7 to 40.1). Furthermore, as mentioned in other research [88], even with the same base model, performance on source domain reported by different papers is very different due to some implementation differences. Performance of our method does not increase when changing the base model from VGG16 to VGG19, which indicates that the style transfer module is the bottleneck of the final performance.

Results of Generalizing from GTA5																						
Method	**target**	**architecture**	**road**	**sidewalk**	**building**	**wall**	**fence**	**pole**	**light**	**sign**	**vegetation**	**terrain**	**sky**	**person**	**rider**	**car**	**truck**	**bus**	**train**	**motorcycle**	**bycycle**	**mIoU (%)**
FCN wld [37]	C	A	70.4	32.4	62.1	14..9	5.4	10.9	14.2	2.7	79.2	21.3	64.6	44.1	4.2	70.4	8.0	7.3	0.0	3.5	0.0	27.1
AdaSeg [87]	C	A	72.9	30.0	74.9	12.1	13.2	15.3	16.8	14.1	79.3	14.5	75.5	35.7	10	62.1	20.6	19	0.0	19.3	12	31.4
MCD [62]	C	A	86.4	8.5	76.1	18.6	9.7	14.9	7.8	0.6	82.8	32.7	71.4	25.2	1.1	76.3	16.1	17.1	1.4	0.2	0.0	28.8
Source	C	A	8.3	10.3	37.2	4.0	12.9	14.6	8.4	3.4	74.6	9.9	31.3	36.5	0.1	53.4	9.5	6.1	0.6	0.1	0.0	16.9
Proposed	C	A	77.9	31.5	72.0	10.2	0.0	19.3	2.1	0.0	79.6	23.6	76.9	28.6	0.0	60.4	8.6	6.2	0.0	0.0	0.0	26.2
Source	C	B	7.4	10.7	35.0	3.8	10.7	13.4	13.8	2.4	76.0	14.2	30.2	40.6	0.0	48.3	9.3	4.2	2.0	0.8	0.0	17.0
Proposed	C	B	82.6	32.4	74.2	16.4	0.0	22.9	0.2	0.0	81.1	21.2	77.1	33.6	0.0	61.4	10.4	2.7	0.0	0.2	0.0	27.2
Source	B	A	21.3	15.4	42.7	1.8	9.9	19.0	13.2	2.7	52.4	11.9	74.3	27.0	0.0	45.9	5.9	7.0	0.0	0.3	0.0	18.4
Proposed	B	A	54.8	21.1	60.0	4.3	0.0	24.5	8.4	0.0	68.1	21.7	84.4	28.6	0.0	53.0	8.9	9.6	0.0	0.0	0.0	23.6

Table 4.1: Results of Generalizing from GTA5. The performance of our proposed method is evaluated by using GTA5 as source labelled training data while evaluating the algorithm on the validation set of Cityscapes. Experiments compare our model using two base semantic segmentation architectures FCN-VGG16 (A) and FCN-VGG19 (B). Target C represents Cityscapes dataset and target B represents BDD100K dataset.

Our adaptation progress requires no target domain data or data related to the target domain. The adaptation progress relies on the prior knowledge that the simulation environment and real-world environment are similar in their geometry information while being different in their texture, lighting situation. This thesis proposes to train a semantic segmentation model which depends mainly on geometry information. To further test the generalization ability of our trained model, the same model is also tested on BDD100K dataset. Performance is also shown in Table 4.1.

Having access to target data typically benefits the policy transfer, since data can be augmented towards target domain distribution. However, in our experiment, using target domain data as noise does not benefit the transfer progress since the learned intermediate feature representation of real-world data is relatively stable and stylized images generated with real-world can not provide enough randomization. This finally leads to bad transfer performance.

Figure 4.3: Results of CGAN, the first column shows original cityscapes images and the second column shows the corresponding fake Carla images

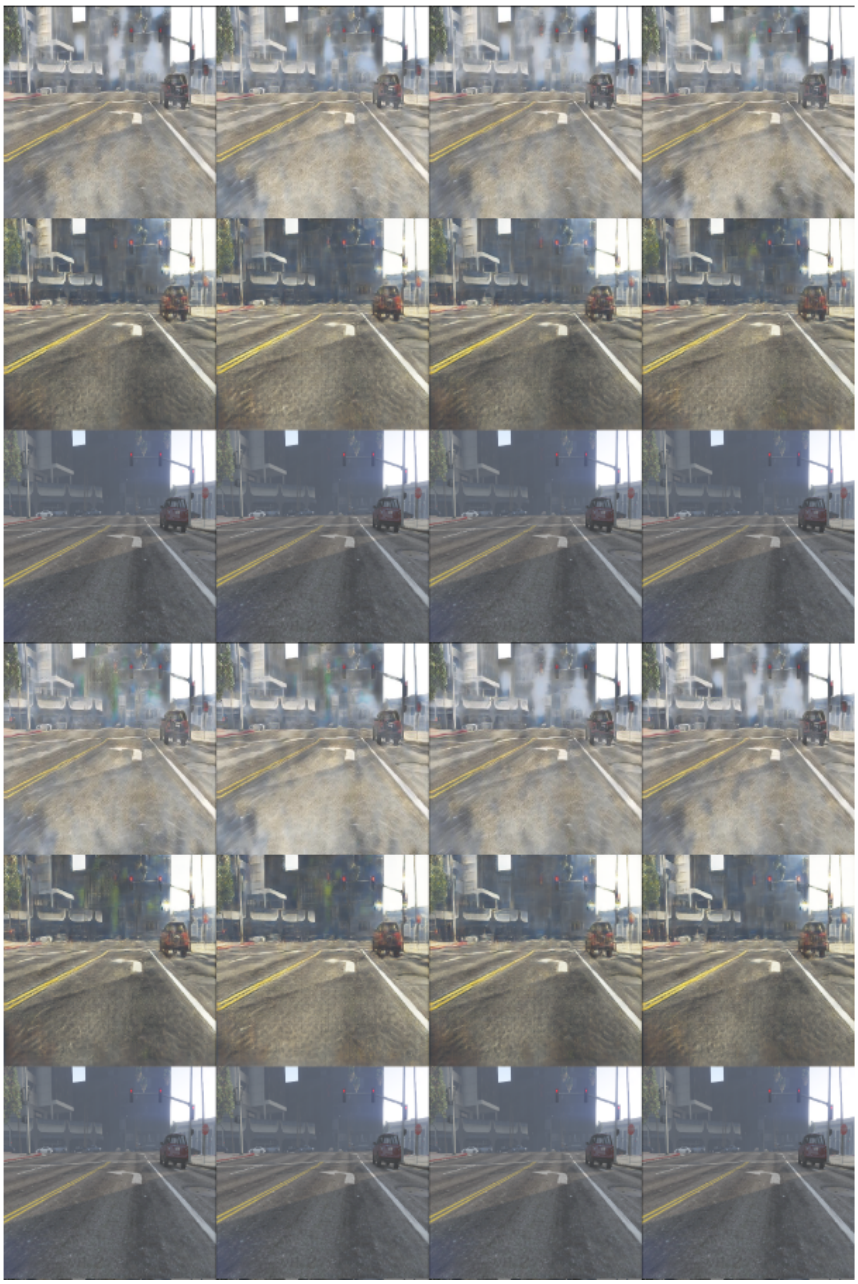

Figure 4.4: Results of CGAN, even lines shows cityscapes images and the odd lines shows the corresponding fake GTA5 images

4.1.2 Style-agnostic Target Domain to Source Domain Generator

Experiment Settings

In experiments, the proposed method uses U-Net [89] structure as generator follows the original CGAN [90]. The semantic segmentation performance of the model is evaluated by passing target domain data to a trained generator and then passing the generated fake source domain data to a FCN trained directly on source domain data.

Datasets The experiments choose GTA5 dataset [83] as our synthesized dataset, Cityscapes [84] and BDD100K [85] as our real-world dataset. This experiment uses Kaggle's Painter by Numbers data set [2] as our noise set. The experiment setting is similar to our Metric Guided Domain Adaptation. Another experiment using Carla dataset generated with Carla's official data collector[3] is also added.

Training There are two experiments, one of them uses GTA5 dataset and the other one uses the generated Carla dataset. The CGAN is trained using randomized source domain (randomized GTA5 and randomized Carla in our experiment) as domain A, and source domain (GTA5 dataset and Carla dataset in our experiment) as domain B. Examples of training data and results is shown in Figure 4.3.

Evaluation The evaluation follows the protocol mentioned in [86]. The real-world RGB image is resized to 512×512 and then passed through to the generator model trained with randomized synthesized images and original synthesized images.The output fake source domain images are passed through to a FCN model trained with source domain data, and then the outputs are resized to the original resolution and evaluated with the standard Pascal VOC intersection-over-union (IoU). mIoU is utilized as evaluation metric.

Results Discussion For Carla-Cityscapes experiments, our Style-agnostic Target Domain to Source Domain Generator shows better performance compared to our Metric Guided Domain Randomization method. While for GTA5-Cityscapes experiment, Metric Guided Domain Randomization has a better performance compared to the other methods. This result is because GTA5 synthetic images are far more complicated compared to Carla synthetic images with more realistic texture, lighting situation. In contrast, Carla synthetic images mostly use the same texture for such category of objects and use similar lighting rendering for most of the situation. Complexed source domain makes CGAN hard to find the mapping function that can map randomized images back to the original images (GTA5 images). This low-quality fake image is shown in 4.4. Since

[2] Painter by Numbers dataset. https://www.kaggle.com/c/painter-by-numbers/data
[3] CARLA 0.8.4 Data Collector https://github.com/carla-simulator/data-collector

the CGAN is not able to reconstruct high-quality source domain images, there is no way that the whole domain adaptation pipeline can be successful.

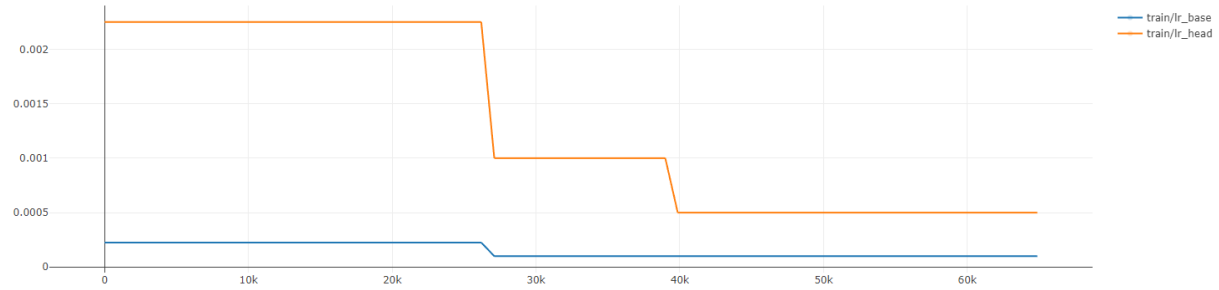

Figure 4.5: learning rate of adaptive PointPillars.

4.2 Domain Adaptation for Point Cloud Object Detection

4.2.1 Adaptive PointPillars

Experiment Settings

The proposed method uses a typical task loss plus domain adaptation loss structure. Object detection is chosen as task and PointPillars is chosen as base network. Performance of the model is evaluated by testing the trained model on target domain data and also testing it after fine-tuning with some target domain data.

Datasets The experiment uses Kitti [4] Dataset 3D object detection task's Velodyne point cloud data as the source domain and Stardust dataset as the target domain. Kitti Dataset has 3712 frames of training data and a similar number of testing data as well, Stardust Dataset has 5005 frames of training data and some number of testing data. Due to different sensor setting, point cloud data in these two datasets are different, which is shown in Figure 4.6. Based on the bird's eye view of two datasets, Kitti Dataset has denser point cloud data and also a wider field of view. This differences course significant performance decrease when applying model trained with one dataset on the other.

Training The experiments apply three different adaptation losses, which are MMD loss, CORAL loss and RevGrad loss. BSP loss is also added to encourage the model to have better discriminability on the target domain. All the different experiments follow the same pipeline:

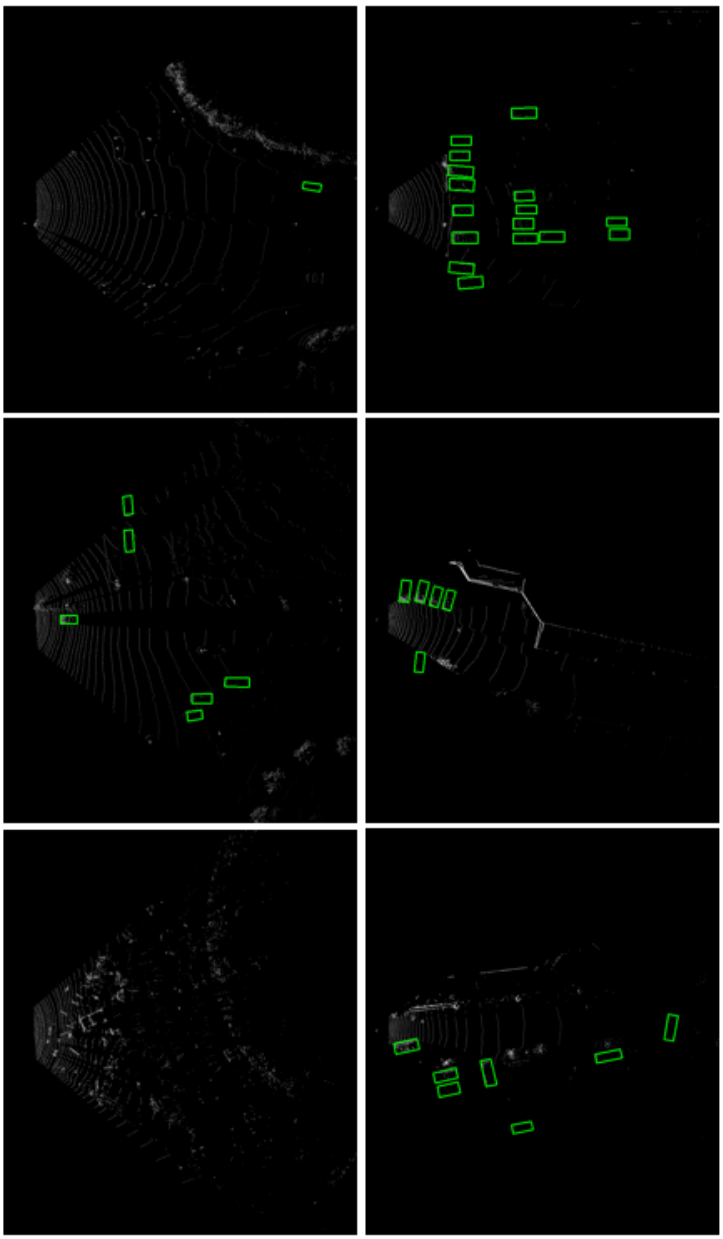

Figure 4.6: Bird's eye view of point cloud data from two different domains, Kitti and Stardust. The first column is from Kitti and the second column is from Stardust. Bounding box is ground truth annotation of cars.

a model that is trained on source domain for one epoch is used as a pre-trained model, all the parameters except the detection heads' and adaptation head's are adapted. A higher learning rate is set for the heads since they do not have pre-trained weights. A step learning rate schedule is applied to the training progress, which is shown in Figure 4.5.

Scenario	Train Set	Val Set	B-E %	B-M %	B-H %	3D-E %	3D-M %	3D-H %
1	kitti	kitti	89.47	79.41	78.55	75.03	63.93	58.00
2	stardust	stardust	88.09	77.79	72.30	67.34	53.19	45.67
3	kitti	stardust	0.00	0.00	0.34	0.00	0.00	0.00
4	stardust	kitti	41.97	40.76	40.51	2.10	4.33	4.50
5	kitti+10%stardust	stardust	59.88	54.40	45.70	27.24	25.03	20.77
6	stardust+10%kitti	kitti	77.84	73.73	70.83	33.86	34.19	31.80
7	kitti+10%stardust	kitti	75.24	63.48	60.29	54.48	40.80	38.00
8	stardust+10%kitti	stardust	88.03	76.67	68.61	57.61	46.73	39.86

Table 4.2: Results of PointPillars trained on source domain only or mixed dataset which contains both source and target domain data, and then tested on target domain. This table shows how large the gap is between source and target domain. Default Kitti evaluation metric is used and AP of both birds' eye view and 3D are reported. B represents Birds' Eye View, E represents easy, M represents moderate, H represents hard.

Evaluation The performance of domain adaptation is evaluated by testing the model with source data directly on the target domain. Then the performance of source domain model fine-tuned with target domain data is also tested on the target domain. The evaluation uses AP of bird's eye view images as evaluation metric and threshold is set to 0.70. The evaluation uses the same evaluation function used in Second. The evaluation first calculates the performance of source domain data on target domain without applying any methods, results of training on mixed datasets and testing on target domain is also presented as well in Table 4.2.

Experiments results on all three of the adaptation losses are reported. The experiments show that that CORAL adaptation loss can not improve adaptation performance for our model. This might relate to the sparsity of point cloud data. Figure 4.7 shows comparison between MMD loss and RevGrad loss.

Experiments show that domain adaptation is better than just fine-tuning on target domain data. Fine-tuning results of both AdaptivePointPillars and original PointPillars on target domain data is reported. Results is shown in Figure 4.8. Results of fine-tuning with 1, 5, 10 target domain data points and also one per cent to ten per cent target domain data are recorded. The results show that with limited target domain data (less than 1000 data points), our method is superior to fine-tuning, while with more target domain data, the performance of the two methods is similar. The superior can also be shown in AP which are shown in Table 4.3.

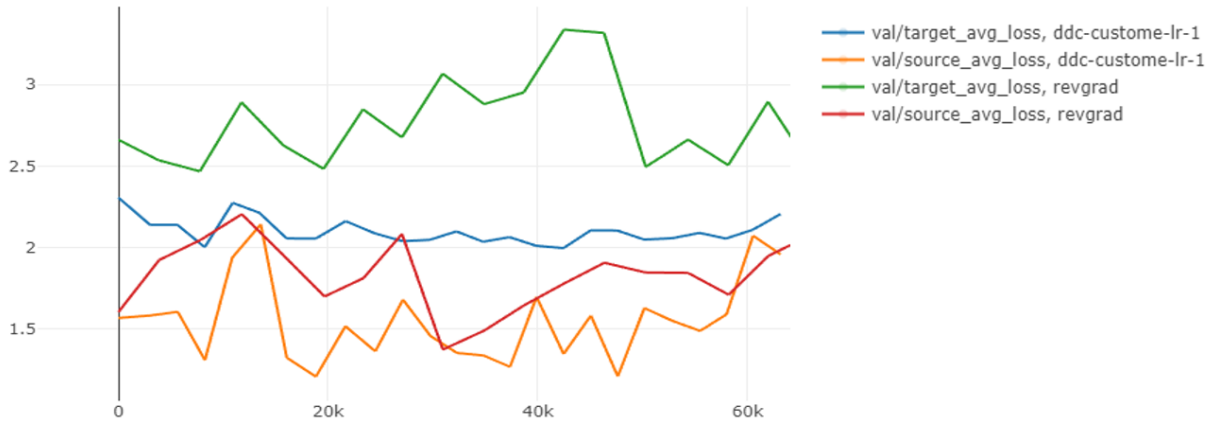

Figure 4.7: Source and Target domain loss of Adaptive PointPillars with MMD loss and RevGrad loss. ddc-custome-lr uses MMD adaptation loss and revgrad uses RevGrad adaptation loss.

	Adaptive PointPillars			Original PointPillars		
Num of Target Domain Data	Car AP (BEV, 0.7, 0.5, 0.5)					
	easy	medium	hard	easy	medium	hard
1	**14.76**	**11.89**	**11.44**	0.32	0.45	0.51
5	**17.33**	**13.43**	**11.99**	6.76	8.73	8.54
10	**30.75**	**20.59**	**15.95**	12.33	14.90	13.31
125	84.68	72.71	67.04	**85.60**	**73.96**	**67.15**
250	86.88	71.53	60.46	**88.05**	**74.20**	**63.20**
375	**89.01**	**79.40**	71.09	87.96	79.24	**72.25**
500	**89.13**	79.96	72.53	88.69	**80.38**	**73.57**
625	86.53	77.41	70.43	**88.18**	**80.68**	**73.20**
750	**88.52**	80.10	**74.13**	88.16	**80.16**	73.02
875	**88.99**	79.56	71.43	88.92	**80.83**	**73.85**
1000	**88.85**	78.11	72.24	88.81	**81.81**	**75.36**

Table 4.3: Fine-tuning Results

Results Discussion Our Adaptive PointPillars shows its superior adaptation performance. When target domain data is extremely limited (less than ten target domain data points), Adaptive PointPillars has an absolutely better performance compared to fine-tune, and with more target domain data, it also shows similar performance compared to fine-tune original PointPillars. More point cloud-specific modification need to be done on the proposed Adaptive PointPillars in order

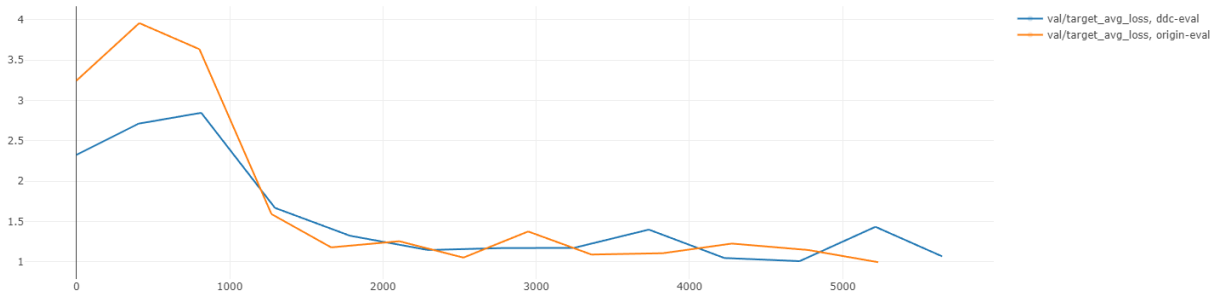

Figure 4.8: Results of fine-tuning with 1, 5, 10 target domain data points and also one percent to 10 percent target domain data points. The blue line is our Adaptive PointPillars and the orange line is original PointPillars.

to achieve better adaptation performance.

Chapter 5

Conclusions and Recommendations

5.1 Conclusions

This thesis mainly focuses on domain adaptation and its application for autonomous driving. The thesis first introduces some basic concepts of domain adaptation and how it can be useful for autonomous driving. Then the current research situation for domain adaptation and the current unexplored area is mentioned. The Methods section is categorized into domain adaptation for RGB images and domain adaptation for point cloud data. For RGB images domain adaptation, the proposed method uses semantic segmentation as the task and describe two different methods.

Generally speaking, Metric Guided Domain Adaptation directly trained task network with randomized source domain data. With the guarantee that using style transfer methods as a randomization method will barely change semantic information between the source image and corresponding randomized image, the trained task network can work on the target domain if target domain images can not be distinguished from randomized source images.

Style-agnostic Target Domain to Source Domain Generator, on the other way, can be considered as training a mapping function that can map the target domain image to its corresponding source domain edition. The mapping function was obtained by training a conditional adversarial network using randomized source domain images as domain A and source domain images as domain B. With a trained generator and task network, the domain adaptation could be done bypass through the target domain images to the generator and then passing through the generated fake source domain images to the task network trained directly on source domain images. These methods depend mainly on the quality of the conditional adversarial network.

When comparing these two proposed methods in two different experiment settings, the Style-

agnostic Target Domain to Source Domain Generator is more suitable for domain adaptation tasks with a less complicated source domain. At the same time, Metric Guided Domain Adaptation works better when the source domain is more complicated.

A baseline method for domain adaptation for point cloud data called Adaptive PointPillars was proposed, aiming to make domain adaptation for point cloud object detection. This method utilized MMD distance as a regularization term when training the original PointPillars network. A BSP penalty that will encourage the latent space to be more complicated during the domain adaptation progress was added to the whole loss function, not only focus on global feature alignment.

With the development of learning-based methods, autonomous driving becomes possible. However, the safety requirement of autonomous driving brings challenges to its perception module. A large amount of data is needed to cover all the potential traffic situations, weather, and lighting, which is nearly impossible even for companies with lots of labour and material resources. Domain adaptation aims to use all kinds of existing data more efficiently, which can be used to reduce the demand for collecting and annotating extra data. Also, domain adaptation can be applied to adapt the policy trained in a simulation environment to real-world situations, bringing unlimited access to any driving data. Data collected from the simulation environment also has ground truth annotation, and no extra labour force is needed for annotation.

Metric Guided Domain Randomization achieved similar performance with some state-of-the-art unsupervised domain adaptation methods. However, different from unsupervised domain adaptation settings where target domain images are accessible. The proposed method does not require any target domain information, including data and corresponding annotations. The proposed proposed Metric Guided Domain Randomization is able to improve mean intersection-over-union on target domain from 16.9 to 27.2 without using any target domain data or annotations. Another technique for RGB image domain adaptation was also utilized, which bases on CGAN. A comparison between these two different methods showed that the CGAN based method's performance decreased a lot as the source domain image becomes more complex, while Metric Guided Domain Adaptation, although performance was a little worse than CGAN based approach, achieved much better performance when the source domain images are more complex.

A baseline method designed for point cloud domain adaptation was proposed. Point Cloud domain adaptation is an unexplored area, with different characteristics like the sparsity of data and the disorder of data points, point cloud domain adaptation is somehow different from RGB image domain adaptation. The proposed method aims to fit point cloud's characteristics but is far from perfect. With access to more ten target domain data frames and fine-tuning will be as good as the proposed Adaptive PointPillars. More effects deserve to be devoted to this field.

5.2 Recommendations

The proposed Metric Guided Domain Randomization and Style-agnostic Target Domain to Source Domain Generator adopt policy from the source domain to target domain on the raw data level. Ideally, adaptation on the feature level will achieve better performance since the extracted feature is task-specific.

Adaptive PointPillars is a baseline method which simply adapt [14] to 3D object detection. Local feature alignment should be considered for better adaptation performance. Self-training based domain adaptation methods are worth trying since they do not focus on global feature alignment.

References

[1] Ekim Yurtsever, Jacob Lambert, Alexander Carballo, and Kazuya Takeda. A Survey of Autonomous Driving: Common Practices and Emerging Technologies. *IEEE Access*, 8:58443–58469, 2020. arXiv: 1906.05113.

[2] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, May 2015. Number: 7553 Publisher: Nature Publishing Group.

[3] H. Caesar, Varun Bankiti, A. Lang, Sourabh Vora, Venice Erin Liong, Q. Xu, A. Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11618–11628, 2020.

[4] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? The KITTI vision benchmark suite. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3354–3361. IEEE, 6 2012.

[5] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, Vijay Vasudevan, Wei Han, Jiquan Ngiam, Hang Zhao, Aleksei Timofeev, Scott Ettinger, Maxim Krivokon, Amy Gao, Aditya Joshi, Yu Zhang, Jonathon Shlens, Zhifeng Chen, and Dragomir Anguelov. Scalability in perception for autonomous driving: Waymo open dataset, 2019.

[6] Alexander Ratner, Stephen H. Bach, Henry R. Ehrenberg, Jason Alan Fries, Sen Wu, and Christopher Ré. Snorkel: Rapid training data creation with weak supervision. *CoRR*, abs/1711.10160, 2017.

[7] Amazon Web Services (AWS) - Cloud Computing Services.

[8] A. Gretton, AJ. Smola, J. Huang, M. Schmittfull, KM. Borgwardt, and B. Schölkopf. Covariate shift and local learning by distribution matching. In *Dataset Shift in Machine Learning*, pages 131–160. Biologische Kybernetik, Cambridge, MA, USA, 2009. 00000.

[9] Lei Zhang. Transfer Adaptation Learning: A Decade Survey. *CoRR*, abs/1903.04687, 2019. 00012.

[10] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An Open Urban Driving Simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, pages 1–16, 2017. 00504.

[11] Shital Shah, Debadeepta Dey, Chris Lovett, and Ashish Kapoor. AirSim: High-Fidelity Visual and Physical Simulation for Autonomous Vehicles. pages 621–635. Springer, Cham, 2018.

[12] S. J. Pan, I. W. Tsang, J. T. Kwok, and Q. Yang. Domain Adaptation via Transfer Component Analysis. *IEEE Transactions on Neural Networks*, 22(2):199–210, February 2011. 01543.

[13] J. Wang, Y. Chen, L. Hu, X. Peng, and P. S. Yu. Stratified Transfer Learning for Cross-domain Activity Recognition. In *2018 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pages 1–10, March 2018. 00042.

[14] Eric Tzeng, Judy Hoffman, Ning Zhang, Kate Saenko, and Trevor Darrell. Deep domain confusion: Maximizing for domain invariance. *CoRR*, abs/1412.3474, 2014.

[15] Mingsheng Long and Jianmin Wang. Learning Transferable Features with Deep Adaptation Networks. *CoRR*, abs/1502.02791, 2015. 01225.

[16] Yaroslav Ganin and Victor Lempitsky. *Unsupervised Domain Adaptation by Backpropagation*. 2014. 01235.

[17] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. Adversarial Discriminative Domain Adaptation. pages 7167–7176, 2017. 00395.

[18] Judy Hoffman, Eric Tzeng, Taesung Park, Jun-Yan Zhu, Phillip Isola, Kate Saenko, Alexei Efros, and Trevor Darrell. CyCADA: Cycle-Consistent Adversarial Domain Adaptation. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1989–1998, Stockholmsmässan, Stockholm Sweden, July 2018. PMLR. 00000.

[19] Xinyang Chen, Sinan Wang, Mingsheng Long, and Jianmin Wang. Transferability vs. Discriminability: Batch Spectral Penalization for Adversarial Domain Adaptation. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 1081–1090, Long Beach, California, USA, June 2019. PMLR. 00000.

[20] Amlan Kar, Aayush Prakash, Ming-Yu Liu, Eric Cameracci, Justin Yuan, Matt Rusiniak, David Acuna, Antonio Torralba, and Sanja Fidler. Meta-Sim: Learning to Generate Synthetic Datasets. *arXiv:1904.11621 [cs]*, April 2019. 00000 arXiv: 1904.11621.

[21] Wei Yang, Xiaolong Wang, Ali Farhadi, Abhinav Gupta, and Roozbeh Mottaghi. Visual Semantic Navigation using Scene Priors. *CoRR*, abs/1810.06543, 2018.

[22] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. *CoRR*, abs/1703.03400, 2017. 00000.

[23] Alon Farchy, Samuel Barrett, Patrick MacAlpine, and Peter Stone. Humanoid Robots Learning to Walk Faster: From the Real World to Simulation and Back. In *Proceedings of the 2013 International Conference on Autonomous Agents and Multi-agent Systems*, AAMAS '13, pages 39–46, Richland, SC, 2013. International Foundation for Autonomous Agents and Multiagent Systems.

[24] Paul Christiano, Zain Shah, Igor Mordatch, Jonas Schneider, Trevor Blackwell, Joshua Tobin, Pieter Abbeel, and Wojciech Zaremba. Transfer from Simulation to Real World through Learning Deep Inverse Dynamics Model. *arXiv:1610.03518 [cs]*, October 2016. arXiv: 1610.03518.

[25] Josiah P Hanna and Peter Stone. Grounded Action Transformation for Robot Learning in Simulation. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17)*, 2017.

[26] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Networks. 6 2014.

[27] Chao Lu, Fengqing Hu, Dongpu Cao, Jianwei Gong, and Yang Xing. Virtual - to - Real Knowledge Transfer for Driving Behaviour Recognition : Framework and a Case Study.

[28] Jingwei Zhang, Jost Tobias Springenberg, Joschka Boedecker, and Wolfram Burgard. Deep reinforcement learning with successor features for navigation across similar environments. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2371–2378. IEEE, 9 2017.

[29] S Barrett, ME Taylor, P Stone Ninth International Conference on, and undefined 2010. Transfer learning for reinforcement learning on a physical robot. *academia.edu*.

[30] Matthew E. Taylor and Peter Stone. Transfer Learning for Reinforcement Learning Domains: A Survey. *Journal of Machine Learning Research*, 10(Jul):1633–1685, 2009.

[31] Mark Cutler, Thomas J. Walsh, and Jonathan P. How. Real-World Reinforcement Learning via Multifidelity Simulators. *IEEE Transactions on Robotics*, 31(3):655–671, 6 2015.

[32] Igor Mordatch, Kendall Lowrey, and Emanuel Todorov. Ensemble-CIO: Full-body dynamic motion planning that transfers to physical humanoids. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5307–5314. IEEE, 9 2015.

[33] Kemin Zhou. ESSENTIALS OF ROBUST CONTROL. Technical report, 1999.

[34] Xiangyu Yue, Yang Zhang, Sicheng Zhao, Alberto Sangiovanni-Vincentelli, Kurt Keutzer, and Boqing Gong. Domain Randomization and Pyramid Consistency: Simulation-to-Real Generalization without Accessing Target Domain Data. *arXiv:1909.00889 [cs]*, September 2019. 00000 arXiv: 1909.00889.

[35] Mingsheng Long, Zhangjie Cao, Jianmin Wang, and Michael I. Jordan. Conditional Adversarial Domain Adaptation. *arXiv:1705.10667 [cs]*, December 2018. 00000 arXiv: 1705.10667.

[36] Geoffrey French, Michal Mackiewicz, and Mark Fisher. Self-ensembling for visual domain adaptation. *arXiv:1706.05208 [cs]*, September 2018. 00000 arXiv: 1706.05208.

[37] Judy Hoffman, Dequan Wang, Fisher Yu, and Trevor Darrell. FCNs in the Wild: Pixel-level Adversarial and Constraint-based Adaptation. *arXiv:1612.02649 [cs]*, December 2016. 00000 arXiv: 1612.02649.

[38] Nataniel Ruiz, Samuel Schulter, and Manmohan Chandraker. Learning To Simulate. *arXiv:1810.02513 [cs, stat]*, May 2019. 00011 arXiv: 1810.02513.

[39] Shai Ben-David, John Blitzer, Koby Crammer, and Fernando Pereira. Analysis of Representations for Domain Adaptation. page 8. 00825.

[40] Alexander Pashevich, Robin A. M. Strudel, Igor Kalevatykh, Ivan Laptev, and Cordelia Schmid. Learning to Augment Synthetic Images for Sim2real Policy Transfer. *arXiv:1903.07740 [cs, stat]*, March 2019. 00000 arXiv: 1903.07740.

[41] Barret Zoph, Ekin D. Cubuk, Golnaz Ghiasi, Tsung-Yi Lin, Jonathon Shlens, and Quoc V. Le. Learning Data Augmentation Strategies for Object Detection. *CoRR*, abs/1906.11172, 2019. 00000.

[42] Ekin D. Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V. Le. RandAugment: Practical automated data augmentation with a reduced search space. *arXiv:1909.13719 [cs]*, November 2019. 00000 arXiv: 1909.13719.

[43] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. *CoRR*, abs/1703.10593, 2017.

[44] Stephen James, Paul Wohlhart, Mrinal Kalakrishnan, Dmitry Kalashnikov, Alex Irpan, Julian Ibarz, Sergey Levine, Raia Hadsell, and Konstantinos Bousmalis. Sim-to-Real via Sim-to-Sim: Data-efficient Robotic Grasping via Randomized-to-Canonical Adaptation Networks. *arXiv:1812.07252 [cs]*, December 2018. 00004 arXiv: 1812.07252.

[45] S. J. Pan, I. W. Tsang, J. T. Kwok, and Q. Yang. Domain adaptation via transfer component analysis. *IEEE Transactions on Neural Networks*, 22(2):199–210, 2011.

[46] Jindong Wang, Yiqiang Chen, Lisha Hu, Xiaohui Peng, and Philip S. Yu. Stratified transfer learning for cross-domain activity recognition. *CoRR*, abs/1801.00820, 2018.

[47] M. Long, J. Wang, G. Ding, J. Sun, and P. S. Yu. Transfer feature learning with joint distribution adaptation. In *2013 IEEE International Conference on Computer Vision*, pages 2200–2207, 2013.

[48] Jindong Wang, Yiqiang Chen, Shuji Hao, Wenjie Feng, and Zhiqi Shen. Balanced distribution adaptation for transfer learning. *CoRR*, abs/1807.00516, 2018.

[49] Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Vaughan. A theory of learning from different domains. *Machine Learning*, 79:151–175, 2010. 00000.

[50] Mingsheng Long, Yue Cao, Jianmin Wang, and Michael Jordan. Learning transferable features with deep adaptation networks. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 97–105, Lille, France, 07–09 Jul 2015. PMLR.

[51] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. Generative adversarial networks. *ArXiv*, abs/1406.2661, 2014.

[52] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. Adversarial discriminative domain adaptation. *CoRR*, abs/1702.05464, 2017.

[53] Shuhao Cui, Shuhui Wang, Junbao Zhuo, Liang Li, Qingming Huang, and Qi Tian. Towards discriminability and diversity: Batch nuclear-norm maximization under label insufficient situations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3941–3950, 2020.

[54] Ming-Yu Liu, Thomas Breuel, and Jan Kautz. Unsupervised image-to-image translation networks. *CoRR*, abs/1703.00848, 2017.

[55] Xun Huang, Ming-Yu Liu, Serge J. Belongie, and Jan Kautz. Multimodal unsupervised image-to-image translation. *CoRR*, abs/1804.04732, 2018.

[56] Seunghyeon Kim, Jaehoon Choi, Taekyung Kim, and Changick Kim. Self-training and adversarial background regularization for unsupervised domain adaptive one-stage object detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 6092–6101, 2019.

[57] Mehran Khodabandeh, Arash Vahdat, Mani Ranjbar, and William G. Macready. A robust learning approach to domain adaptive object detection. *CoRR*, abs/1904.02361, 2019.

[58] Wouter M. Kouw and Marco Loog. A review of domain adaptation without target labels. *IEEE transactions on pattern analysis and machine intelligence*, 2019. 00000.

[59] Chaohui Yu, Jindong Wang, Yiqiang Chen, and Meiyu Huang. Transfer learning with dynamic adversarial adaptation network. In *2019 IEEE International Conference on Data Mining (ICDM)*, pages 778–786. IEEE, 2019.

[60] Yunhun Jang, Hankook Lee, Sung Ju Hwang, and Jinwoo Shin. Learning what and where to transfer. *CoRR*, abs/1905.05901, 2019.

[61] Ying Wei, Yu Zhang, and Qiang Yang. Learning to transfer. *CoRR*, abs/1708.05629, 2017.

[62] Kuniaki Saito, Kohei Watanabe, Yoshitaka Ushiku, and Tatsuya Harada. Maximum classifier discrepancy for unsupervised domain adaptation. *CoRR*, abs/1712.02560, 2017.

[63] Tao Wang, Xiaopeng Zhang, Li Yuan, and Jiashi Feng. Few-shot adaptive faster R-CNN. *CoRR*, abs/1903.09372, 2019.

[64] Jesper E. van Engelen and Holger H. Hoos. A survey on semi-supervised learning. *Machine Learning*, 109(2):373–440, February 2020.

[65] Martin Engelcke, Dushyant Rao, Dominic Zeng Wang, Chi Hay Tong, and Ingmar Posner. Vote3deep: Fast object detection in 3d point clouds using efficient convolutional neural networks. *CoRR*, abs/1609.06666, 2016.

[66] Bo Li, Tianlei Zhang, and Tian Xia. Vehicle detection from 3d lidar using fully convolutional network. *CoRR*, abs/1608.07916, 2016.

[67] Xiaozhi Chen, Huimin Ma, Ji Wan, Bo Li, and Tian Xia. Multi-view 3d object detection network for autonomous driving. *CoRR*, abs/1611.07759, 2016.

[68] Martin Simon, Stefan Milz, Karl Amende, and Horst-Michael Gross. Complex-yolo: Real-time 3d object detection on point clouds. *CoRR*, abs/1803.06199, 2018.

[69] Yin Zhou and Oncel Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. *CoRR*, abs/1711.06396, 2017.

[70] Alex H. Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. *CoRR*, abs/1812.05784, 2018.

[71] Jiquan Ngiam, Benjamin Caine, Wei Han, Brandon Yang, Yuning Chai, Pei Sun, Yin Zhou, Xi Yi, Ouais Alsharif, Patrick Nguyen, et al. Starnet: Targeted computation for object detection in point clouds. *arXiv preprint arXiv:1908.11069*, 2019.

[72] Ross Girshick and Ross. Fast R-CNN. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1440–1448. IEEE, 12 2015.

[73] Joseph Redmon and Ali Farhadi. YOLOv3: An Incremental Improvement. Technical report.

[74] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. SSD: Single Shot MultiBox Detector. pages 21–37. Springer, Cham, 2016.

[75] Xun Huang and Serge Belongie. Arbitrary Style Transfer in Real-time with Adaptive Instance Normalization. *arXiv:1703.06868 [cs]*, July 2017. 00454 arXiv: 1703.06868.

[76] Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The Unreasonable Effectiveness of Deep Features as a Perceptual Metric. *arXiv:1801.03924 [cs]*, April 2018. 00428 arXiv: 1801.03924.

[77] Jiang Wang, Yang Song, Thomas Leung, Chuck Rosenberg, Jingbin Wang, James Philbin, Bo Chen, and Ying Wu. Learning Fine-grained Image Similarity with Deep Ranking. *CoRR*, abs/1404.4661, 2014. 00719 _eprint: 1404.4661.

[78] Michael Gutmann and Aapo Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In Yee Whye Teh and Mike Titterington, editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence*

and Statistics, volume 9 of *Proceedings of Machine Learning Research*, pages 297–304, Chia Laguna Resort, Sardinia, Italy, May 2010. PMLR. 00420.

[79] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-to-Image Translation with Conditional Adversarial Networks. *arXiv:1611.07004 [cs]*, November 2018. 04155 arXiv: 1611.07004.

[80] Baochen Sun and Kate Saenko. Deep CORAL: Correlation Alignment for Deep Domain Adaptation. *CoRR*, abs/1607.01719, 2016. 00000.

[81] Muhammad Ghifary, W. Bastiaan Kleijn, and Mengjie Zhang. Domain adaptive neural networks for object recognition. *CoRR*, abs/1409.6041, 2014.

[82] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully Convolutional Networks for Semantic Segmentation. *arXiv:1411.4038 [cs]*, March 2015. 14349 arXiv: 1411.4038.

[83] Stephan R. Richter, Vibhav Vineet, Stefan Roth, and Vladlen Koltun. Playing for Data: Ground Truth from Computer Games. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *European Conference on Computer Vision (ECCV)*, volume 9906 of *LNCS*, pages 102–118. Springer International Publishing, 2016. 00547.

[84] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The Cityscapes Dataset for Semantic Urban Scene Understanding. *CoRR*, abs/1604.01685, 2016. 02497.

[85] Fisher Yu, Wenqi Xian, Yingying Chen, Fangchen Liu, Mike Liao, Vashisht Madhavan, and Trevor Darrell. BDD100k: A Diverse Driving Video Database with Scalable Annotation Tooling. *CoRR*, abs/1805.04687, 2018. 00236.

[86] Swami Sankaranarayanan, Yogesh Balaji, Arpit Jain, Ser Nam Lim, and Rama Chellappa. Learning from Synthetic Data: Addressing Domain Shift for Semantic Segmentation. *arXiv:1711.06969 [cs, stat]*, April 2018. 00091 arXiv: 1711.06969.

[87] Yang Zhang, Philip David, and Boqing Gong. Curriculum Domain Adaptation for Semantic Segmentation of Urban Scenes. *CoRR*, abs/1707.09465, 2017. 00144.

[88] Yang Zhang, Philip David, Hassan Foroosh, and Boqing Gong. A Curriculum Domain Adaptation Approach to the Semantic Segmentation of Urban Scenes. *IEEE transactions on pattern analysis and machine intelligence*, January 2019. 00000 arXiv: 1812.09953.

[89] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. *CoRR*, abs/1505.04597, 2015.

[90] Mehdi Mirza and Simon Osindero. Conditional Generative Adversarial Nets. *arXiv:1411.1784 [cs, stat]*, November 2014. 02620 arXiv: 1411.1784.

APPENDICES

Appendix A

PyTorch Implement of Three Adaptation Losses

Implements are from Jindong Wang's Transfer Learning reposity. [1]

A.1 Maximum Mean Discrepency

Listing A.1: MMD loss

```
class MMD_loss(nn.Module):
    def __init__(self, kernel_type='rbf', kernel_mul=2.0, kernel_num=5):
        super(MMD_loss, self).__init__()
        self.kernel_num = kernel_num
        self.kernel_mul = kernel_mul
        self.fix_sigma = None
        self.kernel_type = kernel_type

    def guassian_kernel(self, source, target, kernel_mul=2.0, \
                        kernel_num=5, fix_sigma=None):
        n_samples = int(source.size()[0]) + int(target.size()[0])
        total = torch.cat([source, target], dim=0)
        total0 = total.unsqueeze(0).expand(
            int(total.size(0)), int(total.size(0)), int(total.size(1)))
        total1 = total.unsqueeze(1).expand(
            int(total.size(0)), int(total.size(0)), int(total.size(1)))
```

[1] Transfer Learning, Jindong Wang https://github.com/jindongwang/transferlearning

```
        L2_distance = ((total0-total1)**2).sum(2)
        if fix_sigma:
            bandwidth = fix_sigma
        else:
            bandwidth = torch.sum(L2_distance.data) / (n_samples**2-n_samples)
        bandwidth /= kernel_mul ** (kernel_num // 2)
        bandwidth_list = [bandwidth * (kernel_mul**i)
                          for i in range(kernel_num)]
        kernel_val = [torch.exp(-L2_distance / bandwidth_temp)
                      for bandwidth_temp in bandwidth_list]
        return sum(kernel_val)

    def linear_mmd2(self, f_of_X, f_of_Y):
        loss = 0.0
        delta = f_of_X.float().mean(0) - f_of_Y.float().mean(0)
        loss = delta.dot(delta.T)
        return loss

    def forward(self, source, target):
        if self.kernel_type == 'linear':
            return self.linear_mmd2(source, target)
        elif self.kernel_type == 'rbf':
            batch_size = int(source.size()[0])
            kernels = self.guassian_kernel( \
                source, target, kernel_mul=self.kernel_mul, \
                kernel_num=self.kernel_num, fix_sigma=self.fix_sigma)
            with torch.no_grad():
                XX = torch.mean(kernels[:batch_size, :batch_size])
                YY = torch.mean(kernels[batch_size:, batch_size:])
                XY = torch.mean(kernels[:batch_size, batch_size:])
                YX = torch.mean(kernels[batch_size:, :batch_size])
                loss = torch.mean(XX + YY - XY - YX)
            torch.cuda.empty_cache()
            return loss
```

A.2 Correlation Alignment

Listing A.2: CORAL loss

```
def CORAL(source, target):
    d = source.size(1)
    ns, nt = source.size(0), target.size(0)
```

```
    # source covariance
    tmp_s = torch.ones((1, ns)).to(DEVICE) @ source
    cs = (source.t() @ source - (tmp_s.t() @ tmp_s) / ns) / (ns - 1)

    # target covariance
    tmp_t = torch.ones((1, nt)).to(DEVICE) @ target
    ct = (target.t() @ target - (tmp_t.t() @ tmp_t) / nt) / (nt - 1)

    # frobenius norm
    loss = (cs - ct).pow(2).sum().sqrt()
    loss = loss / (4 * d * d)

    return loss
```

A.3 RevGrad

Listing A.3: RevGrad loss

```
class ReverseLayerF(Function):

    @staticmethod
    def forward(ctx, x, alpha):
        ctx.alpha = alpha
        return x.view_as(x)

    @staticmethod
    def backward(ctx, grad_output):
        output = grad_output.neg() * ctx.alpha
        return output, None
```

Appendix B

Key Component of PyTorch Pix2Pix Implement

Borrow a lot from PyTorch-GAN reposity. [1]

Listing B.1: Key Component of PyTorch Pix2Pix Implement

```
for epoch in range(config.getint("train", "epoch"), \
                   config.getint("train", "n_epochs")):
    for i, batch in enumerate(data_loader):

        real_B, real_A = batch

        real_A = real_A.to(device)
        real_B = real_B.to(device)

        # Adversarial ground truths
        valid = torch.from_numpy(np.ones((real_A.size(0), *patch)))
        valid = valid.type(torch.float)
        valid = valid.to(device)
        fake = torch.from_numpy(np.zeros((real_A.size(0), *patch)))
        fake = fake.type(torch.float)
        fake = fake.to(device)
        # ------------------
        #  Train Generators
        # ------------------

        optimizer_G.zero_grad()
```

[1] PyTorch GAN https://github.com/eriklindernoren/PyTorch-GAN

```
        # GAN loss
        fake_B = generator(real_A)
        pred_fake, _ = discriminator(fake_B, real_A, spp=False)
        loss_GAN = criterion_GAN(pred_fake, valid)

        loss_pixel = criterion_pixelwise(fake_B, real_B)

        loss_G = loss_GAN + lambda_pixel * loss_pixel # + lambda_spp * loss_spp

        loss_G.backward()

        optimizer_G.step()

        # ---------------------
        #  Train Discriminator
        # ---------------------

        optimizer_D.zero_grad()

        # Real loss
        pred_real, _ = discriminator(real_B, real_A, spp=False)
        loss_real = criterion_GAN(pred_real, valid)

        # Fake loss
        pred_fake, _ = discriminator(fake_B.detach(), real_A, spp=False)
        loss_fake = criterion_GAN(pred_fake, fake)

        # Total loss
        loss_D = 0.5 * (loss_real + loss_fake)

        loss_D.backward()
        optimizer_D.step()
```