Leveraging Simulation for Computer Vision

Rawal Khirodkar

May, 2019

CMU-RI-TR-19-11

The Robotics Institute

Carnegie Mellon University

Pittsburgh, Pennsylvania 15213

Thesis Committee:

Kris Kitani, *Chair*

Deva Ramanan

David Held

Lerrel Pinto

Submitted in partial fulfillment of the requirements for the degree of Master of Science in Robotics

Abstract

A large amount of labeled data is required to train deep neural networks. The process of data annotation on such a large scale is expensive and time-consuming, especially for complex vision tasks like object pose estimation, semantic segmentation. A promising alternative in this regard, is to use simulation to generate labeled synthetic data. However, a network trained solely on synthetic data may not generalize to real data. This is the problem of *domain gap*, when the training data distribution differs from the test data distribution.

There are two popular solutions to reducing *domain gap*, domain adaptation (DA) and domain randomization (DR). The recently proposed DR has shown promise in control based problems and warrants a study in computer vision. This thesis aims to provide a scientific study of DR with a special focus on its applications in computer vision.

The thesis is composed of three parts.

In the first part, we provide *an empirical analysis of DR* in the context of object detection and pose estimation. Another key contribution as part of our analysis is a semi-automatic pipeline to build 3D simulators for surveillance scenes.

In the second part, we provide *a theoretical analysis of DR*. The highlights here include its generalization error bound, a comparison with DA and its limitations.

Finally, in the third part, we propose *adversarial domain randomization* (ADR) as a solution to DR's limitations. We evaluate ADR along with DR on image classification, object detection, object pose estimation on real world datasets. Our evaluations showcase ADR as a more data efficient approach than DR.

Acknowledgements

Journeys in life are rarely made alone, and the one that this thesis represent is no different.

First, I want to thank my advisor Kris Kitani, for providing an atmosphere where I could pursue any and all of my ideas, and for teaching me to aim high and think big. There were moments when you were more confident in my abilities than I was - thank you for that. I want to thank Donghyun for being the mentor figure I needed. I would like to thank Han Zhao along with Aashi Manglik, both of whom very graciously agreed to go over proofs presented in this thesis. I would like to thank my labmates, Tanya Marwah, Xiaofang Wang, Shashank Tripathi, Yan Xu for the numerous brainstorm sessions.

In addition, I am lucky to have friends who I know will last a lifetime: Akshita Mittel, Nilesh Kulkarni, Siva Chaitanya, Ishan Nigam, Akshat Agrawal, Sumit Kumar. They are the reason behind my numerous memories at CMU. Finally, at the risk of sounding pretentious, and with the cognizance that I never fully can, I want to acknowledge the roles played by the three people closest to me: my mother, Kiran Khirodkar (who gifted me my love for books), my father, Sheshrao Khirodkar (who, amongst other things is responsible for my love for computers), my elder sister, Sneha Khirodkar (who was always the smarter sibling, but didn't know it).

A few words rarely ever capture the true significance of people, but we try.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Overview

Simulation plays an important role in fields like robotics [56], physics [3], biology [8]. We believe, it seems to be a promising solution to an important problem in computer vision, *data annotation*. Many state-of-the-art systems in computer vision are based on deep neural networks. These systems are notoriously known to require large scale training data often of the order of millions of samples. A natural question therefore arises, how much data is enough? [25] did an important study which shows that the performance of these state-of-the-art systems does not plateau even after training on billions of samples. A manual annotation process on such a scale can be expensive and time-consuming, especially for complex vision tasks like semantic segmentation where labels are difficult to specify. It is, thus, natural to explore simulation for generating synthetic data as a cheaper and quicker alternative to manual annotation.

However, while use of simulation makes data annotation easier, it results in the problem of *domain gap*. This problem arises when the training data distribution - *source domain* differs from the test data distribution - *target domain*. In this case, learners trained on synthetic data would suffer from high generalization error. There are two popular solutions to reducing domain gap, (1) the well studied paradigm of domain adaptation (DA) [50] and (2) the recently proposed domain randomization (DR) [47]. DR has shown promise in control based problems and warrants a study in computer vision. This thesis aims to provide a scientific study of DR as a tool to leverage simulation for computer vision.

1.2 Domain Randomization

Given a vision task, we can always find *critical features* in data most useful for the task. For example, shape of the car is more useful for car detection than the color of the car. The underlying assumption when using simulation for training is that we approximate these *critical features* from real data as closely as possible in synthetic data. Under this assumption, these features are also called as *domain invariant features*, features which are common to real data (target domain) and synthetic data (source domain).

The central idea of DR is to generate synthetic data by randomizing over features which are not *critical* or *domain-invariant*. Therefore, a model trained on such varied synthetic data generalizes to real-world without any further fine-tuning. The central idea of DR is to generate synthetic data by randomizing over features which are not *critical* or *domain-invariant*.

1.3 Contributions of this thesis

In this thesis, we take steps towards understanding the advantages and limitations of DR both empirically and theoretically. Specifically:

1. In Chapter 2, we empirically analyse DR in the context of object detection and pose estimation in surveillance setting. We propose a semi-automatic pipeline based on Unreal Engine [37] to model surveillance scenes in 3D capable of performing DR. Our analysis includes an ablative study which examines the effect of different randomizations like appearance randomization, size randomization, shape randomization, light augmentation and distractors.

2. In Chapter 3, we theoretically analyse DR. Specifically, 1) we provide a bound for its generalization error, 2) we show that randomization results in a lower error bound than not randomizing at all, 3) we highlight key limitations of DR.

3. In Chapter 4, as a solution to DR's limitations, we develop Adversarial Domain Randomization (ADR), a more data efficient variant of DR.

1.4 Related Publications

Most ideas described in this thesis have appeared (or are under review) in the following publications:

- R. Khirodkar, D. Yoo and K. Kitani. Domain Randomization for Scene-Specific Car Detection and Pose Estimation. IEEE Winter Conference on Applications of Computer Vision (WACV), 2019.

- (under review) R. Khirodkar, D. Yoo and K. Kitani. Adversarial Domain Randomization. IEEE International Conference on Computer Vision (ICCV), 2019.

Chapter 2

Domain Randomization: An Empirical Analysis

2.1 Introduction

For our analysis, we choose the problem of car detection and pose estimation in surveillance videos. The setup is as follows, consider the scenario in which a surveillance system is installed in a novel location and an image-based car detection and pose estimation model must be trained. A common approach is to frame this as a supervised learning problem and re-use models pre-trained on large visual datasets such as COCO [21], KITTI [10], PASCAL3D+ [53]. However, the data distribution of these off-the-shelf datasets can vary significantly from the in-focus surveillance setting. For instance, the camera elevations of a close circuit camera are different from a vehicle mounted camera, a representative case of the KITTI dataset. Fine-tuning a pre-trained model using a small scene-specific dataset is often performed to ensure good performance during inference. We would like to highlight two major shortcomings of this approach, (1) obtaining a new dataset for each new surveillance location is not scalable due to expensive human hours spent in gathering these annotations, (2) another problem is overfitting to this small domain specific dataset during training. We provide an alternative solution to address these shortcomings using a synthetic data generation pipeline which is scalable and robust to data variance.

Our goal is to develop a method for training a scene-specific car pose estimation and detection model without real annotations. This is an ill posed zero-instance learning problem, however, we do have access to three important priors here: (1) 3D scene geometry, (2) camera parameters and finally (3) priors on the geometry and appearance of the object of interest (such as cars) available in the form rich3D CAD models. We model this domain specific knowledge in the form of a synthetic data generation pipeline to compensate for the absence of real training data. In this pipeline, we model the scene in 3D using state-of-the-art rendering engine Unreal Engine 4. An upside of using synthetic data is that we are capable of generating pixel-level accurate annotations like instance segmentation, depth map, category labels, 3D object locations and pose annotations. However, an underlying challenge here is that using synthetic data as the sole source of supervision can result into poor performance during inference on real data.

This is the aforementioned problem of '*domain gap*' [5,31,33,50]. The model trained on synthetic data can over-fit to it, not generalizing well to real data. We use domain randomization (DR) as a solution to bridge the domain gap. A key advantage of DR is that it provides a solution for domain gap in the data generation phase rather than the algorithm phase. This allows us to modularize our solution nicely into two modules, data generation and learning model. Only the data generation module addresses domain gap, which provides us flexibility of making different design choices for our learning model. We provide an end-to-end framework capable of supporting multiple learning models which is an important attribute of a scalable system.

The contributions of this analysis are: (1) synthetic data generation pipeline, we encode known information like 3D scene geometry, camera parameters, object shape and appearance into rich annotated data, (2) a supportive ablation study for DR, we use DR to generate diverse synthetic data to span the space of realistic images. We conduct an ablation study to examine the effect of individual randomization components like texture randomization, light augmentation and distractors, (3) evaluations on diverse datasets, we benchmark our approach on VIRAT [29], UA-DETRAC [24], EPFL Cars Datasets [30] for the task of car detection and pose estimation.

2.2 Related Work

Our analysis is related to object detection and pose estimation, synthetic data for computer vision, domain adaptation and domain randomization.

3D Models for Object Detection and Pose Estimation Car detection and pose estimation is a well studied problem in the literature(see e.g., [28], [27], [40]). The classical problem of 6 DoF pose estimation of an object instance from a single 2D image has been considered previously as a purely geometric problem known as the *perspective n-point problem (PnP)*. Several closed form and iterative solutions assuming correspondences between 2D keypoints in the image and a 3D model of the object can be found in [18].

Synthetic Data for Computer Vision Dhome *et al.* used synthetic models to recognize objects from a single image [26]. For pedestrian detection, computer generated pedestrian images were used to train classifiers [11]. 3D simulation has been used for multi-view car detection [4] [54] [13]. Sun and Saenko [44] trained 2D object detector with synthetic data generated by 3D simulation.

Domain Adaptation Ganin and Lempitsky proposed a domain adaptation method where the

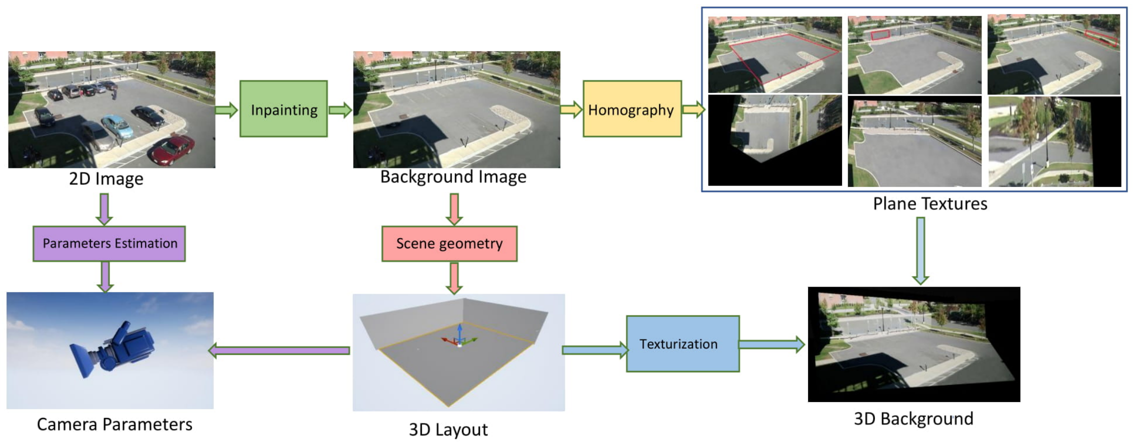


Figure 2.1: 3D background modeling from a single image with camera parameter estimation.

learned feature are invariant to the domain shift [55]. [52] suggested a multichannel autoencoder to reduce the domain gap between real and synthetic data. SimGAN [42] used domain adaption for training eye gaze estimation systems on synthetic eye images. They solved the domain shift problem of synthetic images using a GAN based refiner that converts the synthetic images to the refined images. The refined images have similar noise distribution to real eye images.

Domain Randomization [39] used DR to fly a quadrotor through indoor environments. [7] trained an agent to play Doom and generalize to unseen game levels. [47], [14], [36], [35] used DR for grasping objects. Tremblay *et al.* performed car detection with DR [48]. [45] proposed object orientation estimation for industrial part shapes that is solely trained on synthetic views rendered from a 3D model. They used DR to reduce the gap between synthetic data and real data.

2.3 Method

Our goal is to detect all the cars and estimate their orientations in a specific scene. We assume that scene geometry and camera parameters are given, and use that information along with a rich library of 3D CAD models to generate annotated synthetic data. As alluded to earlier, the main challenge which must be addressed is how to ensure generalization of our network trained on synthetic data to unseen real data. To address this challenge, we employ DR in the data generation step to create synthetic data with enough variations such that the trained network does not over-fit to the synthetic data distribution, thus generalizing to real data during inference.

In this section, we first describe our scene-specific data generation methodology followed by details about DR.

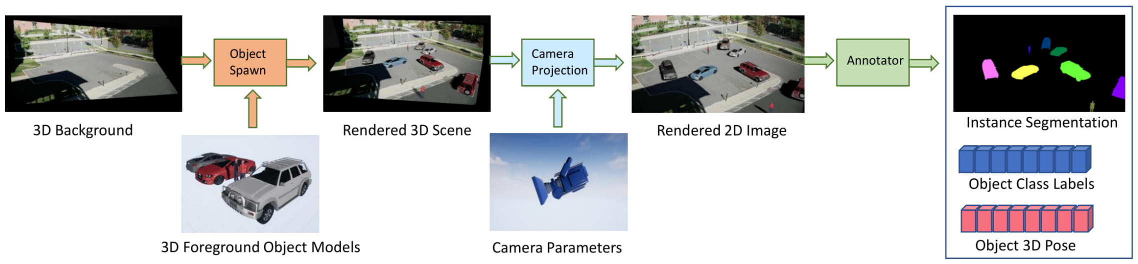


Figure 2.2: Labeled synthetic data generated using Unreal Engine.

2.3.1 Synthetic Data Generation

We use the state-of-the-art rendering platform Unreal Engine 4 for data generation. The first step is to encode the known scene geometry and camera parameters in our model with minimal user supervision. Figure 2.1 shows a schematic overview of this step. Here, we assume access to one image of the scene from the surveillance camera. First foreground objects are removed from this image using in-painting (or background extraction from video) giving us the scene's background image. A textureless 3D layout is constructed from the known scene geometry. The layout is then textured using homographic projections of scene's background image into different views giving us an approximate box-shaped 3D model of the background. The extrinsic camera parameters are estimated using 2D to 3D point correspondences provided by the user, we also perform grid search over an initial estimate of camera intrinsics if the knowledge of these exact parameters are not known. The scale transformation of an unit measure in the real scene to the synthetic scene is provided by the user. Finally, these assets, namely the camera and 3D textured layout represent the known camera parameters and scene geometry respectively.

We now proceed to encode rich priors about the foreground objects appearing in the scene using a large collection of 3D CAD models (see Figure 2.2). 3D models in standard mesh representations are very high dimensional but interpretable and finite dimensional. By virtue of design, this gives us excellent control over the object's characteristic properties like scale, shape, texture. We now proceed to render foreground objects with desirable properties in the 3D layout. This flexibility is critical for our DR setup. After the completion of rendering, we use Unreal Engine's ray tracing to generate pixel level accurate ground truth labels like instance segmentation, class labels and pose annotations.

2.3.2 Domain Randomization

The key idea here is to generate enough variations in the image space to avoid model overfitting. Also, the variations using DR make the model robust to object appearances and light condition.

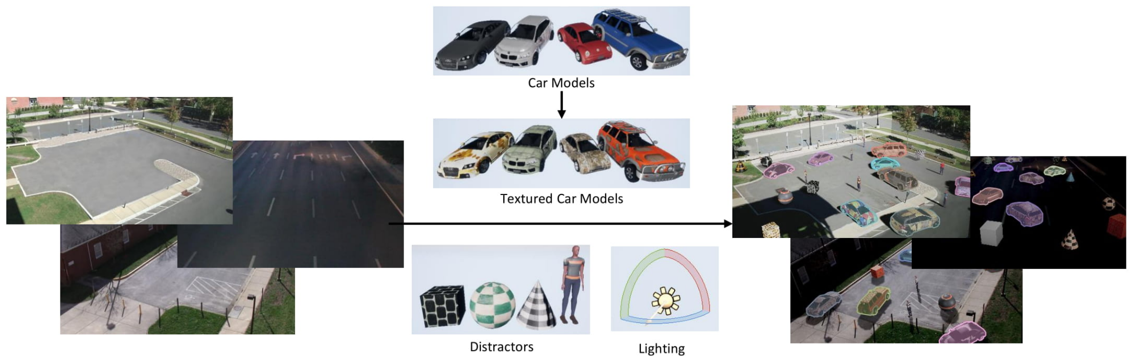


Figure 2.3: Labeled synthetic data generation using domain randomization.

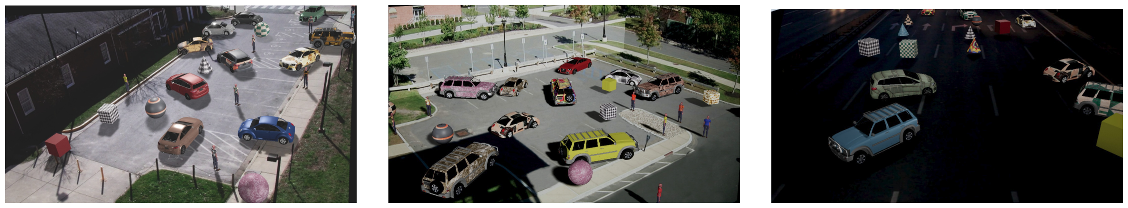


Figure 2.4: Scenes from VIRAT and UA-DETRAC dataset generated using domain randomization.

Concretely, we introduce randomization by varying the following aspects of the scene.

- **Content Variation**: A random number of objects are placed randomly in the 3D background with varying orientation. To achieve shape and size variations we use an array of 5 different types of car models, the model's dimensions are randomly perturbed to provide more fine geometric variations. We also place 3D distractor objects in the scene like pedestrians, cones, spheres at varying dimensions to model the appearance of uninteresting objects in the scene.

- **Style Variation**: The style variation is achieved by randomly varying the color, texture of all the objects. We use a texture bank of 50 textures for this purpose. Furthermore, we apply light augmentations to capture the varying shadow conditions and time of day changes, we also varying the lightning conditions by changing the number of sources along with the location, orientation and luminosity. The generated image also undergoes random changes in contrast and brightness to model slight appearance changes of the background.

2.4 Detection and Pose Estimation Network

For the learning model, our method adopts the Faster-RCNN [38]/Network-on-Convolution meta-architecture. The network consists of a shared backbone feature extractor for the full-image, followed by region-wise sub-networks (heads) that predict the car's yaw angle with respect to the camera coordinate system in addition to traditional 2D bounding box and class label.

Our initial experiments indicated framing the problem of pose estimation as a classification task more favorably than as a regression task. Classification over-parametrizes the problem, and thus allows the network more flexibility to learn the task. As, the pose angles are bounded by $[-\pi, \pi]$, we quantize this space into 36 bins.

For our backbone network, we use the Feature Pyramid Network setup with ResNet101 architecture. Figure 2.5 illustrates our model. We define a multi-task loss function (L) for sampled region proposal as follows:

$L = L_{\text{class}} + L_{\text{bbox}} + L_{\text{pose}}$

where L_{class} is the object classification loss, L_{bbox} is the bounding box regression loss as defined in [38]. L_{pose} is the pose label classification loss.

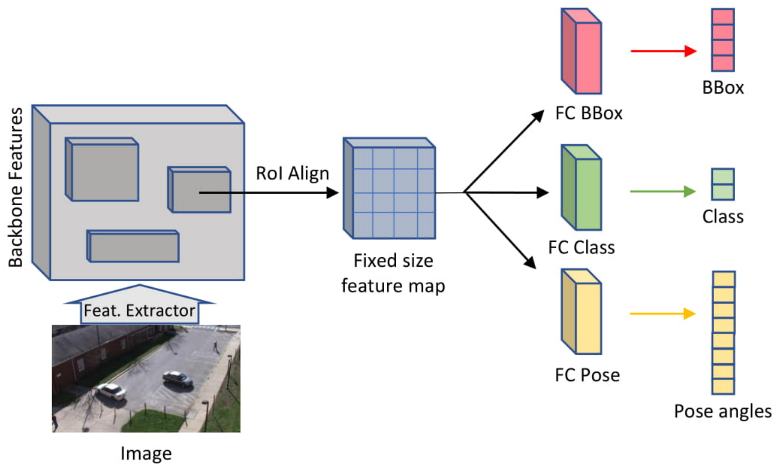


Figure 2.5: Car detection and pose estimation using modified Faster-RCNN.

2.5 Experiments

We analyse DR on three datasets VIRAT, EPFL-Car and UA-DETRAC for the tasks of car detection and pose estimation.

- VIRAT dataset is designed to be realistic, natural and challenging for video surveillance domains in terms of its background clutter, diversity in scenes. The captured video are recorded at 25 frames per seconds (fps) at the resolution of 1920×1080 pixels.

- EPFL-Car dataset contains 20 sequences of cars as they rotate by 360 degrees. The dataset contains images with variation in background, and slight changes in camera elevation recorded at 376×250 pixels.

- UA-DETRAC is a challenging real-world multi-object detection benchmark consisting of 10 hours of videos captured at 24 different locations. The videos are recorded at 25 frames per seconds (fps), with resolution of 960×540 pixels.

We perform evaluations on two scenes from VIRAT dataset, entire EPFL-Car dataset and two scenes from UA-DETRAC dataset. 10,000 synthetic images are rendered for each scene at the resolution of 1440×810 pixels for each variation in the ablation study with 80:20 split as the training and validation set. Furthermore, 8000 images per scene (VIRAT), 2300 images (EPFL-Car), 2000 images per scene (UA-DETRAC) constitutes our real data for evaluations. A random 70:20:10 split of the real images per scene makes up our training, validation and test set respectively.

Note that to capture the background and camera elevation changes in the EPFL-Car dataset, we also randomly perturb the camera parameters and background textures during data generation.

For all our experiments, we compare six models.

- SS-scratch, initialized with random weights and trained on scene specific synthetic data without DR.
- SS-finetune, initialized with COCO and fine-tuned on scene specific synthetic data without DR.
- SS+DR-scratch, initialized with random weights and trained on scene specific domain randomized data.
- SS+DR-finetune, initialized with COCO and fine-tuned on scene specific domain randomized data.
- Real-0.5, initialized with COCO and fine-tuned on 50% real training data.
- Real, initialized with COCO and fine-tuned on the entire real training data.

Without DR: We use 4 car models with fixed dimensions, 7 colors, constant lighting conditions without contrast or brightness augmentation. We do not use cubes, spheres or cones as distractors, however pedestrians are still rendered.

To ensure consistency, we use similar model architecture as reported in section 2.4, same hyper-parameters and training regime. We also compare our models against the baseline of an off-the-shelf pre-trained network for the car detection and pose estimation.

Model	VIRAT 1		VIRAT 2		EPFL-Car		UA-DETRAC Night		UA-DETRAC Street	
	AP@0.5	AP@0.75	AP@0.5	AP@0.75	AP@0.5	AP@0.75	AP@0.5	AP@0.75	AP@0.5	AP@0.75
COCO	92.9	**75.6**	85.5	43.0	55.7	**28.5**	62.9	34.5	78.3	**51.8**
SS-scratch	54.2	37.8	49.1	18.9	31.5	0.4	51.0	20.4	36.7	12.8
SS-finetune	80.3	39.5	54.6	19.2	38.1	3.5	52.5	21.9	38.3	15.0
SS+DR-scratch	98.8	56.4	81.7	34.3	48.6	7.4	78.7	40.6	52.1	29.6
SS+DR-finetune	**98.9**	62.6	**97.6**	**48.8**	**76.8**	22.9	**82.8**	**56.7**	**87.3**	51.2
Real-0.5	94.3	82.1	87.5	57.4	62.3	48.1	82.3	67.1	84.0	44.2
Real	98.9	84.8	98.1	61.8	82.7	60.8	90.1	73.3	92.0	65.1

Table 2.1: Detection results tested on VIRAT, EPFL-Car, UA-DETRAC dataset.

2.5.1 Object Detection

The pre-trained model we use in detection evaluations is a car detector trained on 5010 car images from the COCO dataset (referred as COCO henceforth). We report average precision for bounding box regression at IoU={0.5, 0.75}.

Quantitative Analysis: Table 2.1 shows the detection results tested on VIRAT, EPFL-Car, and UA-DETRAC dataset. Our SS+DR-finetune achieves the best performance in the most cases and outperforms Real-0.5. We hypothesize this is due to overfitting on limited real data which is avoided by design in DR. Furthermore, our model trained only on synthetic data is competitive to the model trained on the entire real data at IoU=0.5.

SS+DR-finetune outperforms the off-shelf baseline COCO by more than 6% and this margin only increases when the scene distribution differs considerably from the off-shelf dataset's distribution.

Qualitative Analysis: Figure 2.6 shows qualitative detection results, in this section we compare COCO, SS-finetune, and SS+DR-finetune. COCO fails in cases of severe object occlusion (first row), unusual shadows (first row, second row) and object truncation (second row, third row). Even though we explicitly model object occlusions and truncation in our synthetic data, SS-finetune fails to generalize to real data. This is because the deep network latches on to the synthetic dataset's biases, our SS+DR-finetune does not suffer from any of these shortcomings.

Furthermore, we found that the bounding boxes of SS+DR-finetune are more accurate than SS-finetune. We argue that this is because SS-finetune overfits to the height-to-width distribution of bounding boxes in the synthetic dataset. DR helps mitigates this problem by randomizing the dimensions and shapes of the 3D models during data generation, thus directly providing enough variance in the bounding box distribution. This regularizes SS+DR-finetune.

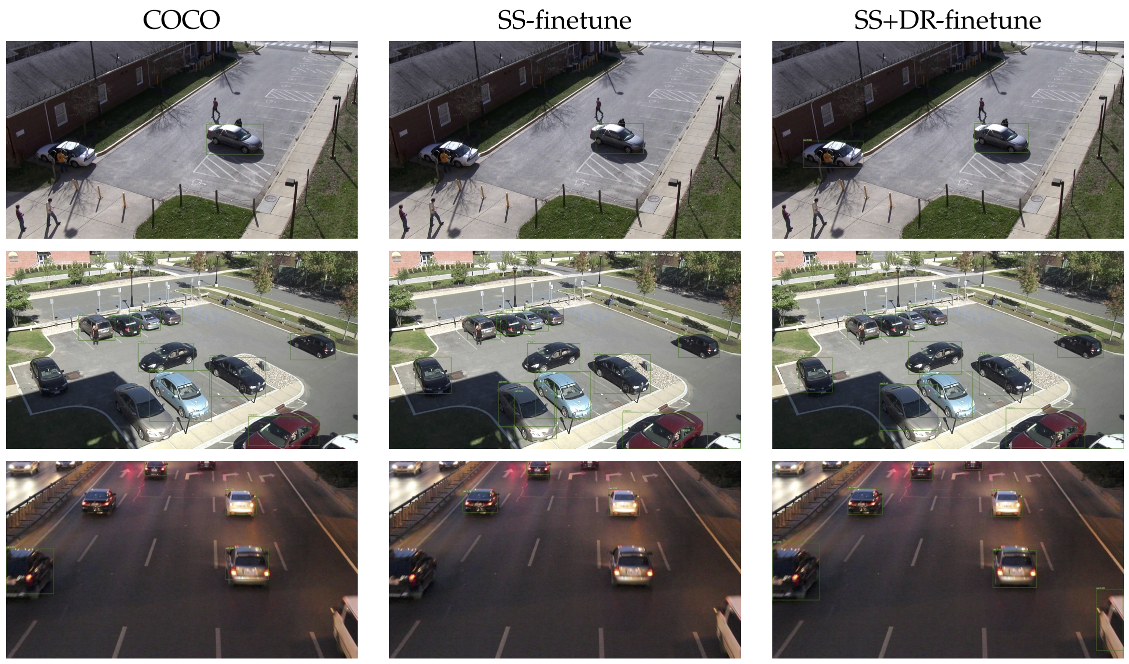


Figure 2.6: Qualitative analysis of detection on VIRAT 1, VIRAT 2 and UA-DETRAC Night.

2.5.2 Pose Estimation

Similar to COCO for detections, we benchmark our models against Deep3DBox [27] trained on KITTI dataset for 3D bounding box estimation. To ensure consistency, we use ground truth detections as our region proposals for all our experiments and extract features using this proposals for the task of pose estimation.

Quantitative Analysis: Table 2.2 shows the pose estimation results, we benchmark our results on EPFL-Car and UA-DETRAC datasets. EPFL-Car already has car pose annotations, we manually annotate two scenes from UA-DETRAC with pose annotations. We report pose classification accuracy $\text{Acc}_{10^o} \uparrow$ at 10^o degree quantization and median error angle MedErr$\downarrow$.

Our SS+DR-finetune achieves the best results that are competitive to Real-0.5 and Real in all the cases even though we don't use any real images for training.

Qualitative Analysis: Figure 2.7 shows qualitative pose results, in this section we compare Deep3DBox-KITTI, SS-finetune, and SS+DR-finetune on images from EPFL-Car and UA-DETRAC Street scene. Note we assume ground truth bounding boxes as input for this analysis.

The camera perspective is significantly different for KITTI and EPFL-Car dataset, this results in poor performance of Deep3DBox on EPFL-Car images. However, SS-finetune performs better

Model	EPFL-Car		UA-DETRAC Night		UA-DETRAC Street	
	$\text{Acc}_{10^o}\uparrow$	MedErr$\downarrow$	$\text{Acc}_{10^o}\uparrow$	MedErr$\downarrow$	$\text{Acc}_{10^o}\uparrow$	MedErr$\downarrow$
Deep3DBox-KITTI	0.67	18.3^o	0.41	28.0^o	0.58	17.7^o
SS-scratch	0.35	46.2^o	0.18	63.1^o	0.45	38.3^o
SS-finetune	0.66	16.1^o	0.51	31.8^o	0.72	23.0^o
SS+DR-scratch	0.72	10^o	0.74	21.6^o	0.61	$\mathbf{10.4^o}$
SS+DR-finetune	**0.86**	$\mathbf{6.6^o}$	**0.83**	$\mathbf{14.2^o}$	**0.87**	13.2^o
Real-0.5	0.91	5.4^o	0.93	11^o	0.96	5.8^o
Real	0.96	6.1^o	0.97	10.4^o	0.97	3.4^o

Table 2.2: Pose estimation results tested on EPFL-Car and UA-DETRAC datasets.

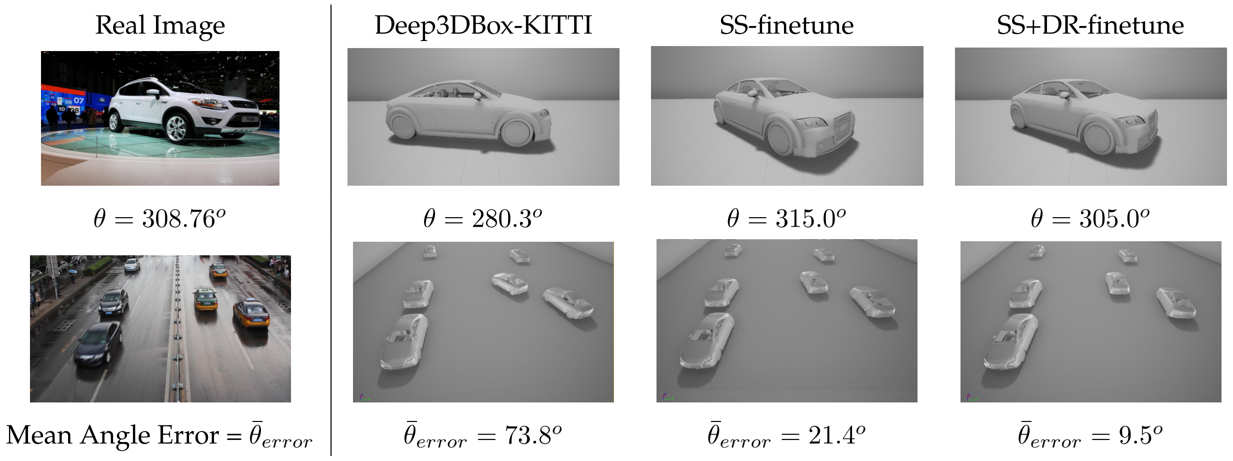

Real Image
Deep3DBox-KITTI
SS-finetune
SS+DR-finetune
$\theta = 308.76^o$
$\theta = 280.3^o$
$\theta = 315.0^o$
$\theta = 305.0^o$
Mean Angle Error = $\bar{\theta}_{error}$
$\bar{\theta}_{error} = 73.8^o$
$\bar{\theta}_{error} = 21.4^o$
$\bar{\theta}_{error} = 9.5^o$

Figure 2.7: Qualitative analysis of pose estimation on EPFL-Car and UA-DETRAC

than Deep3DBox in our analysis, as we assume ground truth bounding box proposals, this is due to the task specific network learning appearance invariant features for pose prediction.

On the UA-DETRAC dataset, the camera perspective is much more similar to KITTI dataset, however the car appearance is significantly different from KITTI. Notice the top right yellow car was flipped by Deep3DBox as it could not distinguish between the head and the tail. Our model SS+DR-finetune performs consistently well across these evaluations.

2.5.3 Ablation Study

We conduct ablation studies to examine the effects of textures (T), light augmentation (LA), geometrical variations to object shapes (G), addition of distractors (D), and full randomization(T + LA + D + G).

We compare (1) LA + G + D, no textures, the object models are rendered without texture vari-

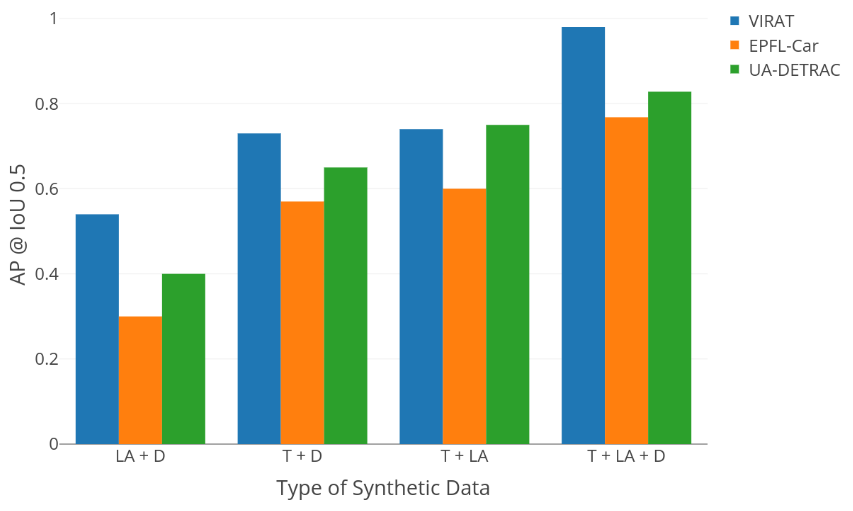


Figure 2.8: An ablation study for detection across VIRAT, EPFL-Car and UA-DETRAC.

ations with a fixed color scheme, (2) T + G + D, no light augmentations, the lighting conditions are held constant, no contrast or brightness changes introduced, (3) T + LA + D: no geometrical variations, every object has fixed dimension, (4) T + LA + G: no distractors, (5) T + LA + D + G, full randomization.

Figure 2.8 shows the comparison chart of the bounding boxes results at IoU=0.5 for the aforementioned randomization conditions for 3 datasets. Removing textures (LA + G + D) results in the most performance drop followed by removing geometrical variations for the task of object detection. Thus, we conclude that varying object appearance and shape and size are two critical components for object detection.

Figure 2.9 shows the comparison chart of the pose estimation results on EPFL-Cars dataset. We assume ground truth bounding box proposals. Removing geometric variations results in the most performance drop followed by the object appearance. The pose estimation head is learning appearance invariant features and is more sensitive to geometry of the object.

2.6 Discussion

To conclude, scene specific treatment along with DR is promising solution for annotation less setups in surveillance. In this empirical analysis, we proposed a framework to generate rich synthetic annotations which incorporate prior knowledge about the scene. Furthermore, using DR we

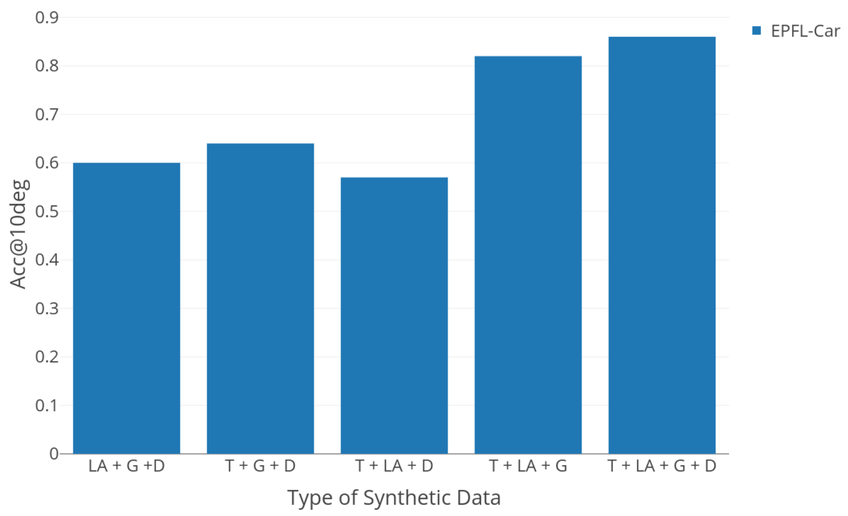


Figure 2.9: An ablation study for pose estimation on EPFL-Car.

ensure any learning algorithm trained on such kind of data would generalize to real data during inference. We performed studies to analyse the effect of each individual randomization components. Compelling results are demonstrated on VIRAT, EPFL-Car and UA-DETRAC datasets for the detection and pose estimation.

Chapter 3

Domain Randomization: A Theoretical Analysis

3.1 Introduction

We ask following key questions: how does DR compensate for the lack of information on target domain? The answer lies in the design of the simulator used by DR for randomization. The simulator is encoded with target domain knowledge (priors on shape of objects) beforehand. This acts as a sanity check between the labels of source domain and target domain (cars should not be labeled as trucks). Another key design decision involves which simulation parameters are supposed to be randomized. This plays an important role in deciding the domain invariant features which help the learner in generalizing to the target domain. Clearly, these domain invariant features are task dependent, as they contain critical information to solve the task. in car detection, the shape of a car is a critical feature and has to be preserved in source domain, it is therefore acceptable to randomize the car's appearance by adding textures in DR. However, when the task changes from car detection to a red car detection, car's appearance becomes a critical feature and can no longer be randomized in DR. We believe it is important to understand such characteristics of the DR algorithm. Therefore, as a step in this direction, in section **??**, we provide theoretical analysis of DR. Concretely our analysis answers the following questions about DR: 1) how does it make the learner focus on domain invariant features? 2) what is its generalization error bound? 3) in presence of unlabeled target data, can we use it with DA? 4) what are its limitations?

We analyze DR using generalization error bounds from multiple source domain adaptation [1]. The key insight is to view DR as a learning problem from multiple source domains where each source domain represents a particular subset of the data space. For example, consider the space of all car images. The images containing a car of a particular make form a subset of that space which we refer to as a single source domain. If one were to generate random images across different car makes, as we do in DR, this can be interpreted as combining data from multiple source domains.

In this section, we first introduce the preliminaries in 3.2 followed by a formal definition of DR algorithm in 3.3. Lastly, we draw parallels between DR and multiple source domain adaptation in 3.4 where we also show that the generalization error bound for DR is better than the bound for data generation without randomization.

3.2 Preliminaries

The notation introduced below is based on the theoretical model for DA using multiple sources for binary classification [1]. The analysis here is limited to binary classification for simplicity but can be extended to other tasks as long as the triangle inequality holds [2].

A *domain* is defined as a tuple $\langle \mathcal{D}, f \rangle$ where: (1) $\mathcal{D}$ is a distribution over input space $\mathcal{X}$ and (2) $f : \mathcal{X} \mapsto \mathcal{Y}$ is a labeling function, $\mathcal{Y}$ being the output space which is $[0, 1]$ for binary classification. N source domains are denoted as $\{\langle \mathcal{D}_i, f_i \rangle\}|_{i=1}^{N}$ and the target domain is denoted as $\langle \mathcal{D}_T, f_T \rangle$. A *hypothesis* is a binary classification function $h : \mathcal{X} \mapsto \{0, 1\}$. The *error* (sometimes called *risk*) of a hypothesis h w.r.t a labeling function f under distribution $\mathcal{D}$ is defined as $\epsilon(h, f, \mathcal{D}) := \mathbb{E}_{x \sim \mathcal{D}}\big[|h(x) - f(x)|\big]$. We denote the error of hypothesis h on target domain as $\epsilon_T(h) = \epsilon(h, f_T, \mathcal{D}_T)$ and on i^{th} source domain as $\epsilon_i(h) = \epsilon(h, f_i, \mathcal{D}_i)$. As common notation in computational learning theory, we use $\epsilon_T(h)$ and $\hat{\epsilon}_T(h)$ to denote the true error and empirical error on the target domain. Similarly, $\epsilon_i(h)$ and $\hat{\epsilon}_i(h)$ are defined for i^{th} source domain.

Multiple Source Domain Problem. Our goal is to learn a hypothesis h from hypothesis class $\mathcal{H}$ which minimizes $\epsilon_T(h)$ on the target domain $\langle \mathcal{D}_T, f_T \rangle$ by only using labeled samples from N source domains $\{\langle \mathcal{D}_i, f_i \rangle\}|_{i=1}^{N}$.

α-Source Domain. We combine N source domains into a single source domain denoted as α-source domain where α helps us control the contribution of each source domain during training. We denote by Δ the simplex of $\mathbb{R}^N$, $\Delta = \{\alpha : \alpha_i \geq 0 \wedge \sum_{i=1}^{N} \alpha_i = 1\}$. Any $\alpha \in \Delta$ forms an α-source domain. The error of a hypothesis h on this source domain (α-error) is denoted as $\epsilon_\alpha(h) = \sum_{i=1}^{N} \alpha_i \epsilon_i(h)$. The α input distribution is denoted as $\mathcal{D}_\alpha = \sum_{i=1}^{N} \alpha_i \mathcal{D}_i$.

The multiple source domain problem can now be reduced to training on labeled samples from α-source domain for various values of $\alpha \in \Delta$. We use $\mathcal{D}_\alpha$ to sample inputs from $\mathcal{X}$, which are then labeled using all the labeling functions $\{\langle f_i \rangle\}|_{i=1}^{N}$. We learn a hypothesis h to minimize the empirical α-error, $\hat{\epsilon}_\alpha(h) = \sum_{i=1}^{N} \alpha_i \hat{\epsilon}_i(h)$.

Generalization Error Bound. Let $\hat{h}_\alpha = \operatorname{argmin}_h \hat{\epsilon}_\alpha(h)$ and $h^*_\alpha = \operatorname{argmin}_h \epsilon_\alpha(h)$ $\hat{h}_\alpha$ and h^*_α minimize the empirical and true α-error respectively. We are interested in bounding the generalization error $\epsilon_T(\hat{h}_\alpha)$ for empirically optimal hypothesis $\hat{h}_\alpha$. However, using Hoeffding's inequality [41], it can be shown that minimum empirical error $\hat{\epsilon}_\alpha(\hat{h}_\alpha)$ converges uniformly to the minimum true error $\epsilon_\alpha(h^*_\alpha)$ without loss of generality $\hat{h}_\alpha$ converges to h^*_α given large number of samples. To simplify our analysis we instead bound generalization error $\epsilon_T(h^*_\alpha)$ for true optimal hypothesis h^*_α (the bound for $\hat{h}_\alpha$ is provided in appendix).

Following the proof for Th. 5 (*A bound using combined divergence*) in [1], we provide a bound for generalization error $\epsilon_T(h^*_\alpha)$ below.

Theorem 1. *(Based on Th.5 [1]) Consider the optimal hypothesis on target domain $h^*_T = \text{argmin}_h\, \epsilon_T(h)$ and on α-source domain $h^*_\alpha = \text{argmin}_h\, \epsilon_\alpha(h)$. If $\gamma_\alpha = \min_h\{\epsilon_T(h) + \epsilon_\alpha(h)\}$, then*

$$\epsilon_T(h^*_\alpha) \leq \epsilon_T(h^*_T) + 2\gamma_\alpha + d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_\alpha, \mathcal{D}_T)$$

Remarks: Proof in appendix. $\epsilon_T(h^*_T)$ is the minimum true error possible for the target domain (clearly, $\epsilon_T(h^*_T) \leq \epsilon_T(h^*_\alpha)$). γ_α represents the minimum error using both the target domain and α-source domain jointly. Intuitively, it represents the agreement between all the labeling functions involved (target domain and all source domains) γ_α would be large, if these labeling functions label an input differently. Lastly, $d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_\alpha, \mathcal{D}_T)$ is the $\mathcal{H}\Delta\mathcal{H}$ divergence between input distributions $\mathcal{D}_\alpha$ and $\mathcal{D}_T$. In summary, generalization on target domain $\epsilon_T(h^*_\alpha)$ depends on: (1) difficulty of task on target domain $\epsilon_T(h^*_T)$; (2) labeling consistency between target domain and source domains γ_α; (3) similarity of input distribution between target and source domains $d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_\alpha, \mathcal{D}_T)$.

3.3 Domain Randomization

DR addresses the multiple source domain problem by modeling various source domains using a simulator with randomization. The simulator is a generative module which produces labeled data (x, y). In practice, DR uses an accurate simulator which internally encodes the knowledge about the target domain as a target labeling function $f_T(x)$, such that $y = f_T(x)$. Concretely, let Θ be the rendering parameter space and $\mathcal{D}_\Theta$ be a probability distribution over Θ. We denote the simulator as a function $g : \Theta \mapsto \mathcal{X} \times \mathcal{Y}$ such that $g(\theta) = \{x, y\}$ where $\theta \sim \mathcal{D}_\Theta$. Simply put, a simulator g takes a set of parameters θ and generates an image x and its label y. In general, the DR algorithm generates data by randomly (uniformly) sampling θ from Θ $\mathcal{D}_\Theta$ is set to $\mathcal{U}_\Theta$, an uniform distribution over Θ (refer Alg. 1). The algorithm outputs a hypothesis $\hat{h}$ which empirically minimizes the loss $\ell(\hat{h}(x_i), y_i)$ over M data samples. Note, in our analysis, we set $\ell(\hat{h}(x), y)$ to be $|\hat{h}(x) - y|$. LEARNER-UPDATE is the parameter update of $\hat{h}$ using loss $\ell(\hat{h}(x_i), y_i)$.
The objective function optimized by these steps can be written as follows:

$$\min_{h\in\mathcal{H}} \mathop{\mathbb{E}}_{\theta\sim\mathcal{U}_\Theta} \left[\ell\Big(h\big(g(\theta)_x\big), g(\theta)_y\Big)\right] \tag{1}$$

Algorithm 1 Domain Randomization

```
    Input: $g, M$
    Output: $\hat{h}$
1: for $i \in \{1, 2, \ldots, M\}$ do
2:     $\theta \sim \mathcal{U}_\Theta$
3:     $\{x, y\} = g(\theta)$
4:     $\hat{h} = \text{LEARNER-UPDATE}(\hat{h}, x, y)$
```

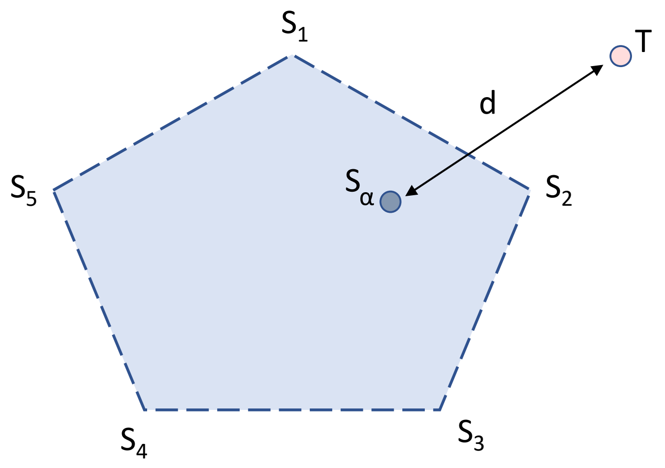


Figure 3.1: A visualization of the domains using simplex $\Delta(N = 5)$. The corners of Δ are the source domains $S_1, \ldots, S_5$ and the point T is the target domain. Any interior point $S_\alpha \in \Delta$ is the upper bound for generalization error $\epsilon_T(h^*_\alpha)$ and the point T is $\epsilon_T(h^*_T)$. The distance between S_α and T is $d = 2\gamma_\alpha + d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_\alpha, \mathcal{D}_T)$. The data generated using DR is the centre of Δ.

3.4 DR as $\bar{\alpha}$-Source Domain

We interpret data generated using DR as labeled data from an α-source domain, specifically $\alpha = [\frac{1}{N}, \ldots, \frac{1}{N}]$ (referred as $\bar{\alpha}$ hereafter). This captures equal contribution by each sample during training according to Alg 1. Using Th. 1, we can bound the generalization error for $\bar{\alpha}$-source domain (DR) by $\epsilon_T(h^*_T) + 2\bar{\gamma} + d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_{\bar{\alpha}}, \mathcal{D}_T)$ where $\bar{\gamma} = \min_h\{\epsilon_T(h) + \frac{1}{N}\sum_{i=1}^{N}\epsilon_i(h)\}$.

We now compare DR with data generation without randomization or variations. The later is same as choosing only one source domain for training, denoted as α_i-source domain where α_i is a one-hot N-vector indicating domain $i \in \{1, \ldots, N\}$. The generalization error bound for α_i-source domain would be $\epsilon_T(h^*_T) + 2\gamma_i + d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_i, \mathcal{D}_T)$ where $\gamma_i = \min_h\{\epsilon_T(h) + \epsilon_i(h)\}$.

Refer to Fig. 3.1 for a visualization of generalization error of S_α as a distance measure from target domain, we define $d = 2\gamma_\alpha + d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_\alpha, \mathcal{D}_T)$ as the upper bound for $|\epsilon_T(h^*_\alpha) - \epsilon_T(h^*_T)|$. We wish to find an optimal α which minimizes this distance measure the point in Δ closest to the target domain (α_2 in Fig. 3.1). However, when no unlabeled target data is available it is best to choose the centre of Δ ($\alpha = \bar{\alpha}$) as our source domain. We prove this in lemma 5, which states that the distance of target domain from the centre of Δ is less than the average distance from the corners of Δ.

Lemma 2. *For $i \in \{1, \dots, N\}$, let $\alpha_i = [0, ..\underset{i^{th}}{1}..0]$, $\gamma_i = \min_h\{\epsilon_T(h) + \epsilon_i(h)\}$ and $\bar{\alpha} = [\frac{1}{N}, \frac{1}{N} \dots, \frac{1}{N}]$, $\bar{\gamma} = \min_h\{\epsilon_T(h) + \frac{1}{N}\sum_{i=1}^{N}\epsilon_i(h)\}$, then*

$$\frac{1}{N}\sum_{i=1}^{N}\Big(2\gamma_i + d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_i, \mathcal{D}_T)\Big) \geq 2\bar{\gamma} + d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_{\bar{\alpha}}, \mathcal{D}_T)$$

Remarks: Proof provided in appendix follows from the convexity of distance measure $2\gamma_i + d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_i, \mathcal{D}_T)$ with application of Jensen's inequality [19]. Using this lemma, the corollary 2.1 states that in the absence of unlabeled target data, in expectation DR (centre of Δ) is superior to data generation without randomization (any other point in Δ).

Corollary 2.1. *The generalization error bound for $\bar{\alpha}$-source domain (DR) is smaller than the expected generalization error bound of a single source domain (expectation over a uniform choice of source domain).*

3.5 Discussion

DR is a powerful method that bridges the gap between real and synthetic data. There is a need to analyze such an important technique, our work is the first step in this direction. We theoretically show that DR is superior to synthetic data generation without randomization. We also identify DR's requirement of a lot of data for generalization.

Chapter 4

Adversarial Domain Randomization

4.1 Introduction

In our previous analysis, we identify a key limitation: DR requires a lot of data to be effective. This limitation mainly arises from DR's simple strategy of using a uniform distribution for simulation parameter randomization. As a result, DR often generates samples which the learner has already seen and is good at. This wastes valuable computational resources during data generation. We can view this as an exploration problem in the space of simulation parameters. A uniform sampling strategy in this space clearly does not guarantee sufficient exploration in image space. We address this limitation in here by proposing Adversarial Domain Randomization (ADR), a data efficient variant of DR. ADR generates adversarial samples with respect to the learner during training. We implement ADR as a policy whose action space is the quantized simulation parameter space. At each iteration, the policy is updated using policy gradients to maximize learner's loss on generated data whereas the learner is updated to minimize this loss. As a result, we observe ADR frequently generates samples like truncated and occluded objects for object detection and confusing classes for image classification.

Finally, we go back to the question posed previously, can we use DR with DA when unlabeled target data is available? Our reinforcement learning framework for ADR easily extends to incorporate DA's feature discrepancy minimization in form of reward for the policy. We thus incentivize the policy to generate data which is consistent with target data. As a result, we no longer need to manually encode domain knowledge in the design of the simulator. In summary, the contributions of our work are as follows:

- **Adversarial Domain Randomization**: As a solution to DR's limitations, we propose a more data efficient variant of DR, namely ADR.

- **Evaluations on Real Datasets for Diverse Tasks**: We benchmark our approach on image classification and object detection on real-world datasets like Syn2Real [34], VIRAT [29].

4.2 Method

We modify DR's objective (eq.1) by making a pessimistic (adversarial) assumption about $\mathcal{D}_\Theta$ instead of assuming it to be stationary and uniform. By making this adversarial assumption, we force the learned hypothesis to be robust to adversarial variations occurring in the target domain. This type of worst case modeling is especially desirable when annotated target data is not available for rare scenarios.

The resulting min-max objective function is as follows:

$$\min_{h \in \mathcal{H}} \max_{\mathcal{D}_\Theta} \mathop{\mathbb{E}}_{\theta \sim \mathcal{D}_\Theta} \left[\ell\Big(h\big(g(\theta)_x\big), g(\theta)_y \Big) \right] \tag{2}$$

4.2.1 ADR via Policy Gradient Optimization

This adversarial objective function is a zero-sum two player game between SIMULATOR (g) and LEARNER (h). The SIMULATOR selects a distribution $\mathcal{D}_\Theta$ for data generation and the LEARNER chooses $h \in \mathcal{H}$ which minimizes loss on the data generated from $\mathcal{D}_\Theta$. The Nash equilibrium of this game corresponds to the optima of the min-max objective, which we find by using reinforcement learning with policy gradients [46]. The SIMULATOR's action $\mathcal{D}_\Theta$ is modeled as the result of following the policy π_ω with parameters ω. g samples θ according to π_ω, which is then converted into labeled data (x, y). The LEARNER's action h is optimized to minimize loss $\ell(h(x), y)$. The reward r_θ for policy π_ω is set to this loss. Specifically, we maximize the objective $J(\omega)$ by incrementally updating π_ω, where

$$J(\omega) = \mathop{\mathbb{E}}_{\theta \sim \pi_\omega} [r(\theta)] \qquad \text{where} \quad r(\theta) = \ell\Big(h(g(\theta)_x), g(\theta)_y \Big).$$

We use REINFORCE [51] to obtain gradients for updating ω using an unbiased empirical estimate of $\nabla_\omega J(\omega)$

$$\hat{J}(\omega) = \frac{1}{M} \sum_{i=1}^{m} \nabla_\omega \log(\pi_\omega(\theta))[r(\theta) - b] \tag{3}$$

where b is a baseline computed using previous rewards and M is the data size. Both SIMULATOR (π_ω) and LEARNER (h) are trained alternately according to Alg. 2 shown pictorially in Fig. 4.1.

4.2.2 DA as ADR with Unlabeled Target Data

In the original formulation of the ADR problem above, our task was to generate a multi-source data that would be useful for any target domain. An easier variant of the problem exists where we

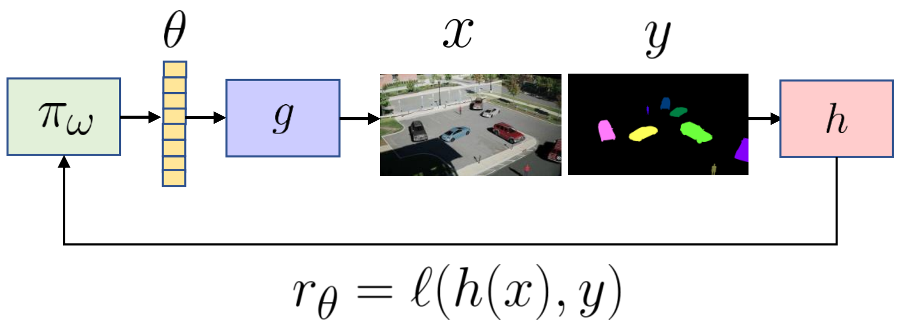


Figure 4.1: π_ω is the policy with parameter ω, g is the simulator which takes θ as input and generates a labeled data sample x, y. The learner h is trained to minimize $\ell(h(x), y)$ on the data sample. π_ω is rewarded to maximize the learner's loss.

Algorithm 2 Adversarial Domain Randomization

```
    Input: $g$, $M$
    Output: $\hat{h}$
1: for $i \in \{1, 2, \ldots, M\}$ do
2:     $\theta_1 \sim \pi_\omega$
3:     $\{x_1, y_1\} = g(\theta_1)$
4:     $\hat{h} = \text{LEARNER-UPDATE}(\hat{h}, x_1, y_1)$
5:
6:     $\theta_2 \sim \pi_\omega$
7:     $\{x_2, y_2\} = g(\theta_2)$
8:     $r = \ell(\hat{h}(x_2), y_2)$
9:     $\pi_\omega = \text{SIMULATOR-UPDATE}(\pi_\omega, r, \theta)$        ▷ Using eq. 3
```

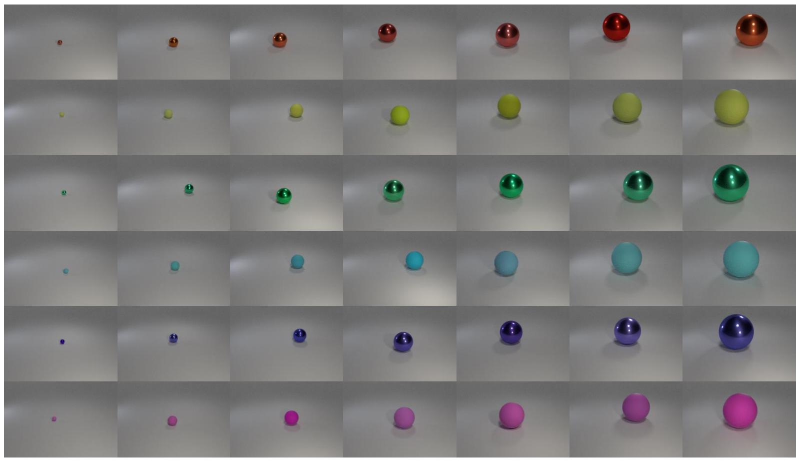
do have access to unlabeled target data. As mentioned before, this falls under the DA paradigm.

To use unlabeled target data, similar to [49] we introduce a domain classifier D which empirically computes $d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_\alpha, \mathcal{D}_T)$. D takes $\phi_h(x)$ as input where ϕ_h is a function which extracts feature from input x using h. D classifies $\phi_h(x)$ into either from target domain (label 1) or source domain (label 0). The reward function for π_ω is modified to incorporate this distance measure as

$$r(\theta) = \ell\Big(h(g(\theta)_x), g(\theta)_y\Big) + w_1 \log D\Big(\phi_h(g(\theta)_x)\Big)$$

where w_1 is a hyper-parameter. This new reward encourages the policy π_ω to fool D, which makes the simulator g generate synthetic data which looks similar to target data.

However, it is plausible that due to simulator's limitations, we might never be able generate data that looks exactly like target data the simplex Δ corresponding to g might be very far from


(a) Target domain: sphere images

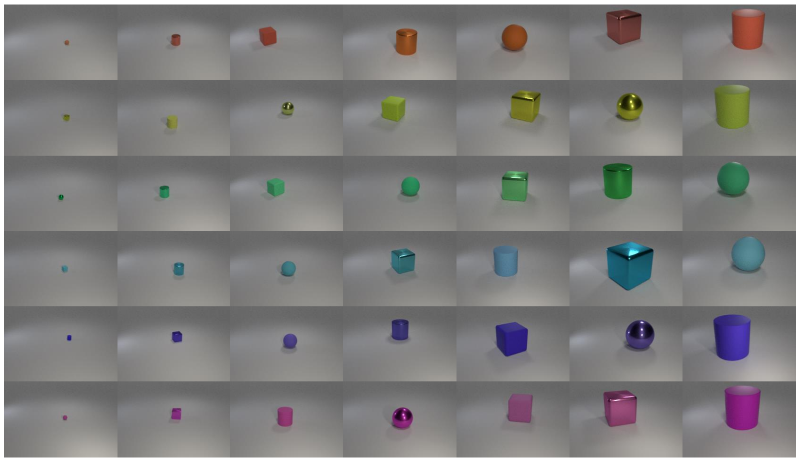

(b) Source domain: sphere, cube, cylinder images

Figure 4.2: Image classification (6 classes) for CLEVR.

point T. In this case, we also modify h's loss as $\ell\Big(h(g(\theta)_x), g(\theta)_y\Big) + w_2 \log D\Big(\phi_h(g(\theta)_x)\Big)$ (w_2 is a hyper-parameter). As a result, h extracts features $\phi_h(x)$ which are domain invariant. This allows both g and h to minimize distance measure from the target domain.

4.3 Experiments

We evaluate ADR in three settings, (1) *perfect simulation* for CLEVR [15] (T inside Δ), (2) *imperfect simulation* for Syn2Real [34](T outside and far from Δ), (3) *average simulation* for VIRAT [29] (T outside but close to Δ). We perform image classification for the first two settings and object detection for the third setting.

4.3.1 Image Classification

CLEVR

We use Blender3D to build a simulator using assets provided by [15]. The simulator generates images containing exactly one object and labels them into six categories according to the color of the object. The input space $\mathcal{X}$ consists of images with resolution of 480×320 and the output space $\mathcal{Y}$ is $\{$red, yellow, green, cyan, purple, magenta$\}$. Here, $\theta \in \Theta$ corresponds to [color, shape, material, size]. Specifically, 6 colors, 3 shapes (sphere, cube, cylinder), 2 materials (rubber, metal) and 6 sizes. Other parameters like lighting, camera pose are randomly sampled.

As a toy target domain, we generate 5000 images consisting only of spheres. Refer Fig. 6.1 for visualizations of target and source domain.

ADR Setup: The policy π_ω consists of $|\text{color}| \times |\text{shape}| \times |\text{material}| \times |\text{size}| = 252$ parameters representing a multinomial distribution over Θ, initialized as a uniform distribution. The learner h is implemented as ResNet18 followed by a fully connected layer for classification which is trained end-to-end. The domain classifier D is a small fully connected network accepting 512 dimensional feature vector extracted from conv5 layer of ResNet18 as input.

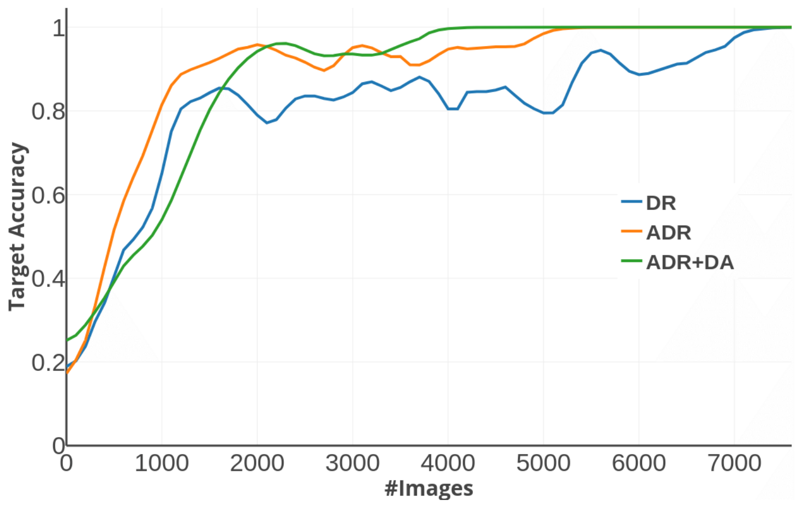
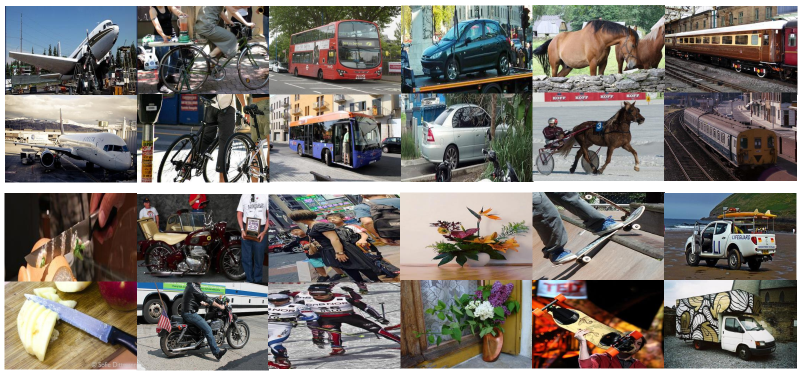


Figure 4.3: Effect of data size on DR, ADR and ADR+DA, target classification accuracy on CLEVR averaged over 10 independent runs.


(a) Target domain

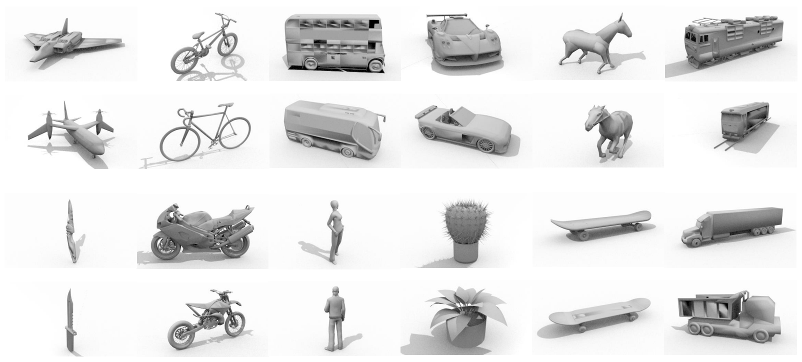

(b) Source domain

Figure 4.4: Image classification (12 classes) for Syn2Real.

Results: We compare the target classification accuracy of DR, ADR and ADR+DA with the number of training images in Fig. 4.3. Please note that for ADR+DA, we independently generated 1000 unlabeled images from the target domain. ADR eventually learns to generate images containing an object of small size (first and second column in Fig 6.1). On the other hand, DR keeps on occasionally generating images with large objects, such images are easier for the learner to classify due to a large number of pixels corresponding to the color of object. Interestingly ADR + DA performs the best, the domain classifier learns to discriminate generated images on the basis of the shape being sphere or not. This encourages the policy π_ω to generate images with spheres (images similar to target domain).

#Images	10k	25k	50k	100k	All(150k)
DR	8.1	10.0	13.8	23.5	28.1 [34]
ADR	**15.3**	**18.6**	**24.9**	**31.1**	**35.9**

Table 4.1: Effect of data size on DR and ADR, target classification accuracy for Syn2Real

Syn2Real

We use 1,907 3D models by [34] in Blender3D to build a simulator for Syn2Real image classification. In our experiment, we use the validation split of Syn2Real as the target domain. The split contains 55,388 real images from the Microsoft COCO dataset [21] for 12 classes. The input space $\mathcal{X}$ consists of images with resolutions 384×216 and the output space $\mathcal{Y}$ are the 12 categories. Here $\theta \in \Theta$ corresponds to [image class]. Other parameters like camera elevation, lighting and object pose are randomly sampled after quantization. Refer Fig. 6.2 for visualizations of target and source domain. There are a total of 152,397 possible values of simulation parameters, each corresponds to an unique image in the source domain.

ADR Setup: π_ω consists of $|\text{category of image}| = 12$ parameters representing a multinomial distribution over Θ, initialized as a uniform distribution. The learner h is AlexNet [17] initialized with weights learned on ImageNet [6]. The hyperparameters are similar to [34].

Results: Table 1 compares target classification of DR and ADR with varying data size. We observe that ADR focuses on confusing classes like (1) car and bus, (2) bike and motorbike by trading off samples for easier classes like plant and train. Note, we also include the case when every possible image variation (152,397) is used to train the learner h (All-150k). In this case, ADR reduces to hard negative mining over image classes which performs better than the baseline (Source) [34].

We also study the effect of ADR with unlabeled target data and various DA methods like ADA [49], Deep-CORAL [43], DAN [23], SE [9] with same setup as [34] except for ADA, we use a domain classifier with ResNet18 architecture. Refer Table 4.2 for per class performance of ADR-150k with various DA methods. Note, we use all the target images without labels for adaptation.

4.3.2 Object Detection

We use the Unreal Engine 4 based simulator by [16] for the VIRAT dataset [29]. VIRAT contains surveillance videos of parking lots. The simulator models the parking lot and the surveillance camera in a 3D world. The randomization process uses a texture bank of 100 textures with 10 cars, 5 person models and geometric distractors (cubes, cones, spheres) along with varying lighting

DA Method	aero	bike	bus	car	horse	knife	mbke	prsn	plant	skbrd	train	truck	mean
All Source [34]	53	3	50	52	27	14	27	3	26	10	64	4	28.1
ADA	68	41	63	34	57	45	74	30	57	24	63	15	47.6
D-CORAL [34]	76	31	60	35	45	48	55	28	56	28	60	19	45.5
DAN [34]	71	47	67	31	61	49	72	36	64	28	70	19	51.6
SE [34]	97	87	84	64	95	96	92	82	96	92	87	54	85.5
ADR	49	12	52	56	38	25	31	14	34	32	59	29	35.9
ADA + ADR	73	50	60	38	59	51	79	37	60	41	69	35	54.3
D-CORAL + ADR	78	56	71	48	64	59	77	45	68	49	70	55	61.6
DAN + ADR	87	60	73	40	59	56	68	43	72	39	68	51	59.7
SE + ADR	94	85	88	72	89	93	91	88	93	86	84	75	86.4
Real [34]	94	83	83	86	93	91	90	86	94	88	87	65	87.2

Table 4.2: Effect of unlabeled target data with ADR on Syn2Real

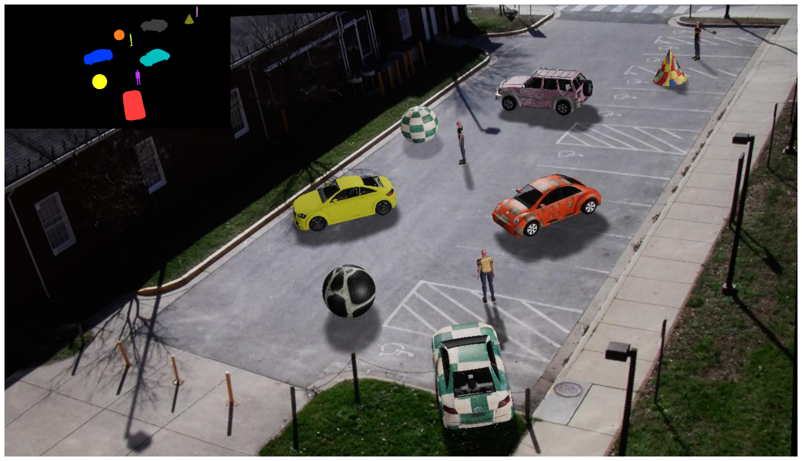
conditions, contrast and brightness. In our experiment, we use 50,000 images from two scenes of the dataset as the target domain. The input space $\mathcal{X}$ consists of images with resolution 1920×1080 and the output space $\mathcal{Y}$ is the space of car bounding boxes (atmost 20) present in the image. Here $\theta \in \Theta$ is a list of object attributes in the image. These attributes specify the location and type of the object in the image. Fig. **??**, 4.6 shows labeled samples from source domain.

ADR Setup: We divide the scene ground plane into 45 rectangular cells (refer Fig. 4.7). In each cell, we place three kinds of objects (car, person or distractor). The policy π_ω consists of $|\text{cells}| \times |\text{types of objects}| = 135$ parameters representing the object spawn probability the policy π_ω decides which object is placed where in the scene. To include variable number of objects in the image, we randomly sample n = number of objects ($2 \leq n \leq 12$) and invoke π_ω n times. Other parameters like lighting, texture, car model, pose, size of the object are randomly sampled. The reward for policy π_ω is computed per cell and is negative of the IoU of the bounding box predicted by the learner h.

The learner h is implemented as Faster-RCNN [38] with RoI-Align and ResNet101 with feature pyramid network [20] architecture as the backbone for all our evaluations.

#Images	1k	10k	25k	50k	100k
DR	20.6	32.1	43.7	54.9	75.8
ADR	**31.4**	**43.8**	**56.0**	**78.2**	**88.6**

Table 4.3: Effect of data size on DR and ADR, Faster-RCNN's AP at 0.7 IoU on VIRAT.


(a) Domain Randomization

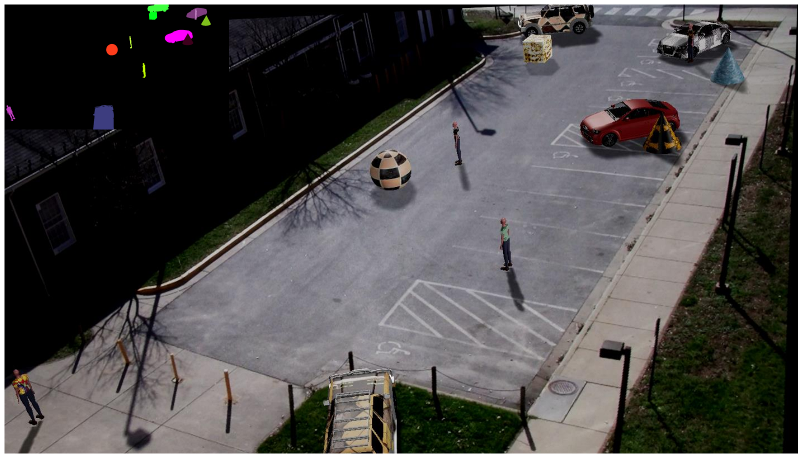
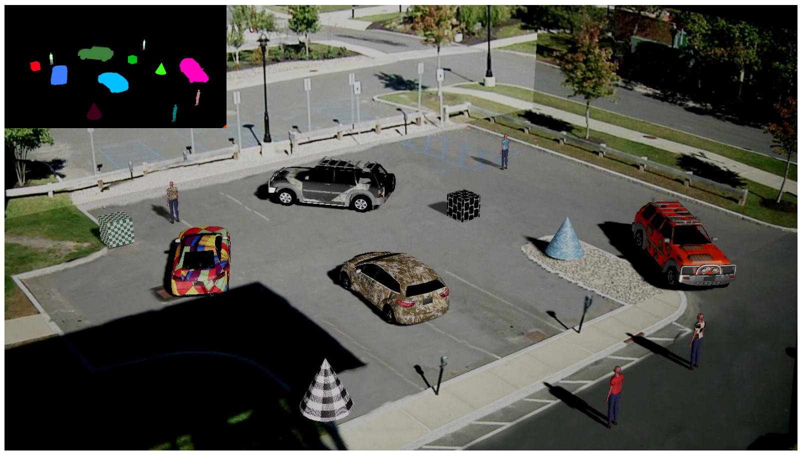

(b) Adversarial Domain Randomization

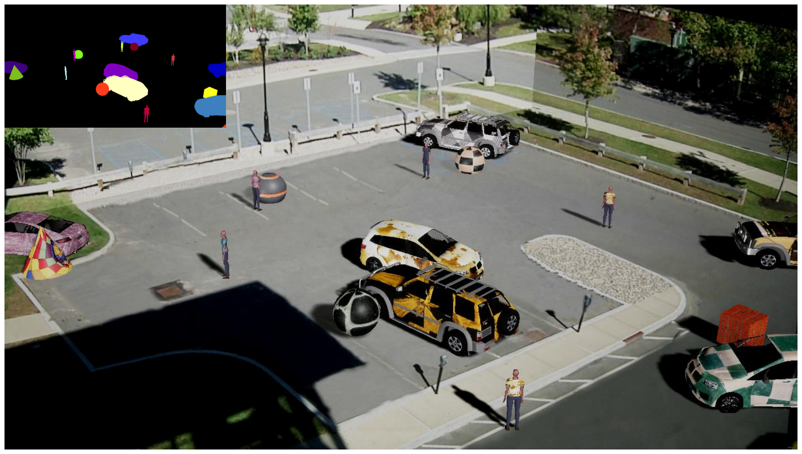
Figure 4.5: A comparison of source data for VIRAT1 from DR and ADR along with ground truth instance segmentation map.

Method	AP @ 0.7
COCO	80.2
ADA	84.7
ADR-100k + ADA	**93.6**
Real	98.1

Table 4.4: Effect of unlabeled target data with ADR on VIRAT.



(a) Domain Randomization



(b) Adversarial Domain Randomization

Figure 4.6: A comparison of source data for VIRAT2 from DR and ADR along with ground truth instance segmentation map.

Results: Table 4.3 compares performance of DR and ADR on target data along with the size of synthetic data. ADR outperforms DR consistently by generating informative samples for object detection containing object occlusions and truncations. We compare data samples generated by DR and ADR in Fig. 4.6 along with a visualization of π_ω learned by ADR in Fig. 4.7, π_ω is shown as a heat-map (warmer colors indicate higher object spawn probability). ADR learns to place objects far from the camera, thus making them difficult for the learner to detect. Fig. 4.8 shows examples of car detection from Faster-RCNN trained on data generated by ADR, affirming that our method performs well under severe truncations and occlusions.

We also evaluate ADR with 5,000 unlabeled target images (not in the test set). Refer Table 4.4 for

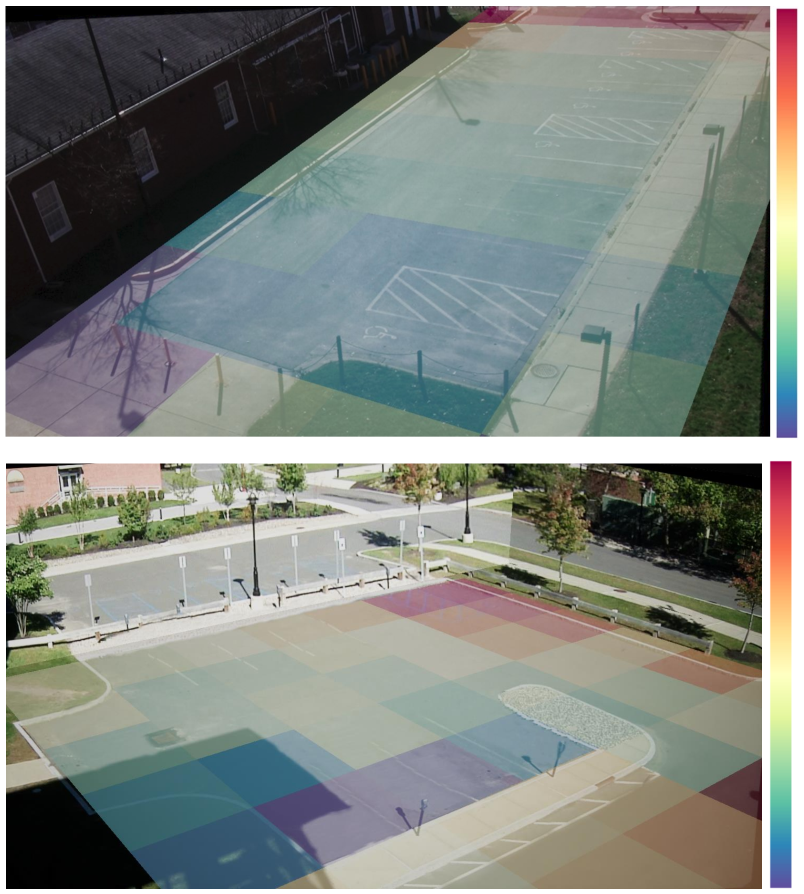


Figure 4.7: Object spawn probability (π_ω) visualized as a heat-map. The warmer colors indicate higher probability which correspond to small/truncated/occluded objects in the image.

comparison of (1) Faster-RCNN trained on Microsoft-COCO dataset (COCO), (2) ADA [49] with ResNet18 as domain classifier, (3) ADR + ADA with 100k source images and (4) Faster-RCNN trained on target images from VIRAT (Real). Using unlabeled data with synthetic data boosts performance from (for DR 75.8 to 84.7, for ADR 88.6 to 93.6). However, the combination of labeled synthetic data and unlabeled real data still performs worse than labeled real data. We provide more analysis in supplementary material.

4.4 Discussion

As an alternative to DR, we proposed ADR, which generates adversarial samples with respect to the learner during training. Our framework easily incorporates any non-differential simulator and

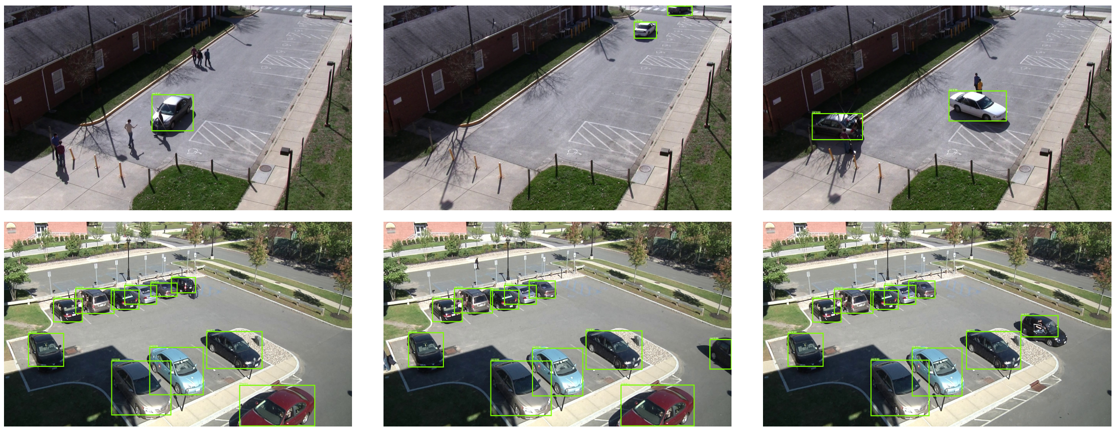


Figure 4.8: Object detection performance of Faster-RCNN trained on 100k images generated by ADR

also makes use of unlabeled target data when available. Furthermore, our evaluations show that ADR outperforms DR using less data for image classification and object detection on real datasets like VIRAT, Syn2Real.

Chapter 5

Future Work

Simulation is a natural solution to collect and label data cheaply and on a large scale. However, in this thesis we assumed access to a simulator beforehand. We believe there are two key problems, (1) building a simulator and (2) bridging the reality gap. This thesis focuses on the latter part. We now wish to focus on the first problem. The underlying research question here is - can we model the real world into 3D using data? This requires automating the entire workflow of making a 3D scene, for example we can take notes from the standard practices in the gaming industry. The standard workflow consists of (1) 3D object modeling, (2) adding material, (3) adding texture and (4) world building. The gaming industry has already automated material and texture addition and it the 3D object modeling and world building which needs a data based solution. Our first approach to these problems would be to look into procedural modeling. [32] used procedural modeling to built cities, we wish to extend this approach to real world scenes using learning based approaches.

Apart from this, we also would like to point key unsolved theoretical problems in bridging the reality gap. For instance, generalization error bounds for complex vision tasks like object detection, semantic segmentation and pose estimation. Finally, we wish to apply domain randomization to video based tasks like activity detection and recognition.

Chapter 6

Appendix

6.1 Derivation of Generalization Error Bound

Lemma 3. *(Based on Lemma 3 [1]) For any hypotheses $h_1, h_2 \in \mathcal{H}$,*

$$|\epsilon_\alpha(h_1, h_2) - \epsilon_T(h_1, h_2)| \leq \frac{1}{2} d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_\alpha, \mathcal{D}_T)$$

Proof. By the definition of $\mathcal{H}\Delta\mathcal{H}$ divergence, $d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_\alpha, \mathcal{D}_T)$

$$\begin{aligned}
&= 2 \sup_{h,h' \in \mathcal{H}} |\Pr_{x\sim\mathcal{D}_\alpha}[h(x) \neq h'(x)] - \Pr_{x\sim\mathcal{D}_T}[h(x) \neq h'(x)]| \\
&= 2 \sup_{h,h' \in \mathcal{H}} |\epsilon_\alpha(h, h') - \epsilon_T(h, h')| \\
&\geq 2|\epsilon_\alpha(h_1, h_2) - \epsilon_T(h_1, h_2)|
\end{aligned}$$

□

Lemma 4. *(Based on Th.5 [1]) For fixed $h \in \mathcal{H}$ and α, if $\gamma_\alpha = \min_h\{\epsilon_T(h) + \epsilon_\alpha(h)\}$, then*

$$|\epsilon_\alpha(h) - \epsilon_T(h)| \leq \gamma_\alpha + \frac{1}{2} d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_\alpha, \mathcal{D}_T)$$

Proof. $|\epsilon_\alpha(h) - \epsilon_T(h)|$

$$\begin{aligned}
&\leq |\epsilon_\alpha(h) - \epsilon_\alpha(h, h^*)| + |\epsilon_\alpha(h, h^*) - \epsilon_T(h, h^*)| \\
&+ |\epsilon_T(h, h^*) - \epsilon_T(h)| \\
&\leq \epsilon_\alpha(h^*) + |\epsilon_\alpha(h, h^*) - \epsilon_T(h, h^*)| + \epsilon_T(h^*) \\
&\leq \gamma_\alpha + \frac{1}{2} d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_\alpha, \mathcal{D}_T)
\end{aligned}$$

□

Lemma 5. *For $i \in \{1, 2, ..N\}$, let $\alpha_i = [0, ..\underset{i^{th}}{1}..0]$, $\gamma_i = \min_h\{\epsilon_T(h) + \epsilon_i(h)\}$, $\bar{\alpha} = [\frac{1}{N}, \frac{1}{N}..., \frac{1}{N}]$, $\bar{\gamma} = \min_h\{\epsilon_T(h) + \frac{1}{N}\sum_{i=1}^{N}\epsilon_i(h)\}$, then*

$$\frac{1}{N}\sum_{i=1}^{N}\Big(2\gamma_i + d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_i, \mathcal{D}_T)\Big) \geq 2\bar{\gamma} + d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_{\bar{\alpha}}, \mathcal{D}_T)$$

Proof. Let the input source distributions $\mathcal{D}_1, \mathcal{D}_2, \ldots \mathcal{D}_N$ be represented by density functions $p_1, p_2, \ldots p_N$. Similarly, let the input target distribution $\mathcal{D}_T$ be represented by the density function q. We can now represent each ith source domain by two functions p_i and f_i.

Consider an arbitrary source domain S represented by functions p and f. We show that $\gamma = \min_h\{\epsilon_T(h) + \epsilon_S(h)\}$ and $d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_S, \mathcal{D}_T)$ is convex in p and f.

Clearly, $\epsilon_S(h)$ is convex in p and f. Therefore, γ is convex in p and f. We now show that the $\mathcal{H}-$ divergence $d_{\mathcal{H}}(\mathcal{D}_S, \mathcal{D}_T)$ is convex in p for all $\mathcal{H}$. $d_{\mathcal{H}\Delta\mathcal{H}} = d_{\mathcal{H}'}$ such that $\mathcal{H}' = \mathcal{H}\Delta\mathcal{H}$.

$\forall\mathcal{H}$, $d_{\mathcal{H}}(p, q)$ is convex in $\forall q$, $\forall p_1, p_2$, $\lambda \in [0, 1]$.
$d_{\mathcal{H}}(\lambda p_1 + (1-\lambda)p_2, q)$

$$\begin{aligned}
&= \text{sup}_{A\in\mathcal{H}^{-1}}|(\lambda p_1 + (1-\lambda)p_2)(A) - q(A)| \\
&= |(\lambda p_1 + (1-\lambda)p_2)(A^*) - q(A^*)| \\
&\leq \lambda|p_1(A^*) - q(A^*)| + (1-\lambda)|p_2(A^*) - q(A^*)| \\
&\leq \lambda\text{sup}_{A\in\mathcal{H}^{-1}}|p_1(A) - q(A)| + (1-\lambda)\text{sup}_{A\in\mathcal{H}^{-1}}|p_2(A) - q(A)| \\
&\leq \lambda d_{\mathcal{H}}(p_1, q) + (1-\lambda)d_{\mathcal{H}}(p_2, q)
\end{aligned}$$

The lemma follows from using Jensen' inquality $f(\mathbb{E}[X]) \leq \mathbb{E}[f(X)]$ where $f = \gamma_S + d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_S, \mathcal{D}_T)$ is a convex function and X is random variable represents choice over N source domains. $\square$

Theorem 6. *(Based on Th.5 [1]) Consider the optimal hypothesis on target domain $h_T^* = \text{argmin}_h\,\epsilon_T(h)$ and on α-combination of source domains $h_\alpha^* = \text{argmin}_h\,\epsilon_\alpha(h)$. If $\gamma_\alpha = \min_h\{\epsilon_T(h) + \epsilon_\alpha(h)\}$, then*

$$\epsilon_T(h_\alpha^*) \leq \epsilon_T(h_T^*) + 2\gamma_\alpha + d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_\alpha, \mathcal{D}_T))$$

Proof.

$$
\begin{aligned}
|\epsilon_\alpha(h_T^*) - \epsilon_T(h_T^*)| &\le \gamma_\alpha + \frac{1}{2} d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_\alpha, \mathcal{D}_T) && \text{L4} \\
\epsilon_\alpha(h_T^*) &\le \epsilon_T(h_T^*) + \gamma_\alpha + \frac{1}{2} d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_\alpha, \mathcal{D}_T) \\
\epsilon_\alpha(h_\alpha^*) &\le \epsilon_T(h_T^*) + \gamma_\alpha + \frac{1}{2} d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_\alpha, \mathcal{D}_T) \\
&\because \epsilon_\alpha(h_\alpha^*) \le \epsilon_\alpha(h_T^*) \\
|\epsilon_\alpha(h_\alpha^*) - \epsilon_T(h_\alpha^*)| &\le \gamma_\alpha + \frac{1}{2} d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_\alpha, \mathcal{D}_T) && \text{L4} \\
\epsilon_T(h_\alpha^*) &\le \epsilon_\alpha(h_\alpha^*) + \gamma_\alpha + \frac{1}{2} d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_\alpha, \mathcal{D}_T) \\
\epsilon_T(h_\alpha^*) &\le \epsilon_T(h_T^*) + 2\gamma_\alpha + d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_\alpha, \mathcal{D}_T)
\end{aligned}
$$

□

Corollary 6.1. *The generalization error bound for an equal weighted combination of source domains (DR) is smaller than the expected generalization error bound of a single source domain (expectation over an uniform choice of source domain).*

Proof. For $i \in \{1, 2, ..N\}$, let $h_i^* = \operatorname{argmin}_h \epsilon_i(h)$ then using $\alpha = \alpha_i = [0, ..\underset{i^{\text{th}}}{1}..0]$ in Th.6, we have

$$
\begin{aligned}
\epsilon_T(h_i^*) &\le \epsilon_T(h_T^*) + 2\gamma_i + d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_i, \mathcal{D}_T)) \\
\mathbb{E}_{i\sim\mathcal{U}_N}[\epsilon_T(h_i^*)] &\le \mathbb{E}_{i\sim\mathcal{U}_N}[\epsilon_T(h_T^*) + 2\gamma_i + d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_i, \mathcal{D}_T)] \\
\frac{1}{N}\sum_{i=1}^{N} \epsilon_T(h_i^*) &\le \epsilon_T(h_T^*) + \frac{1}{N}\sum_{i=1}^{N}\Big(2\gamma_i + d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_i, \mathcal{D}_T)\Big)
\end{aligned}
$$

Similarly, for an equal weighted combination of source domains, let $\bar{h}^* = \frac{1}{N}\operatorname{argmin}_h \sum_{i=1}^{N} \epsilon_i(h)$ then using $\alpha = \bar{\alpha} = [\frac{1}{N}, ..\frac{1}{N}]$ in Th. 6, we have

$$
\epsilon_T(\bar{h}^*) \le \epsilon_T(h_T^*) + 2\bar{\gamma} + d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_{\bar{\alpha}}, \mathcal{D}_T)
$$

The relationship between the upper bounds follows from L5. □

6.2 CLEVR: Color Classification

Figure 6.1 shows the probability of generation over the most critical rendering dimension for this toy problem i.e size. ADR's policy π_ω effectively focuses on harder (smaller) objects to achieve better performance on target data. Each iteration corresponds to

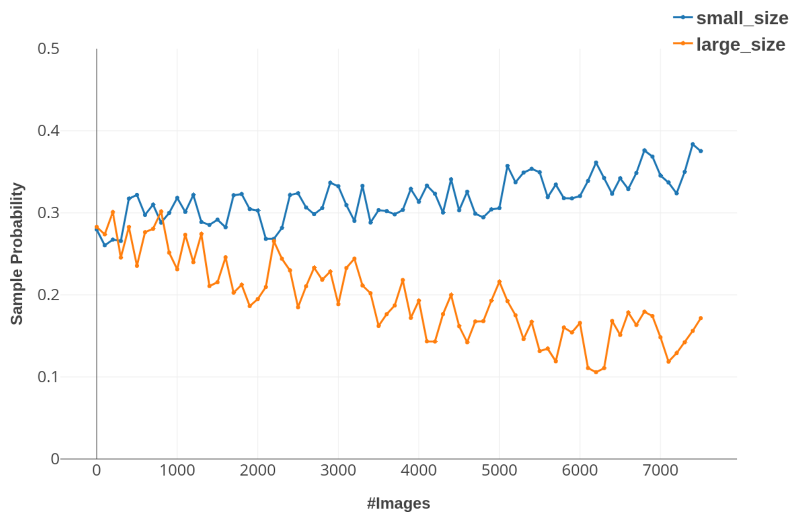


Figure 6.1: ADR sample probability over size dimension according to π_ω averaged over 10 runs. small-size are the two smallest sizes and large-size are the two largest sizes.

6.3 Syn2Real: Object Classification

Figure 6.2 visualizes sample generation probability over all the 12 classes after generating 100k images. Our proposed method focuses on confusing group of classes like {bike, motorbike} and {bus, car, train, truck } rather than generating visually distinct classes like {aero, horse, knife, person, plant, skbrd}.

6.4 VIRAT: Depth Estimation

In this section, we demonstrate the utility of our approach on a challenging use-case. We perform monocular depth estimation on VIRAT dataset using synthetic data. Please note that the VIRAT dataset consists only of RGB videos and no depth maps. Furthermore, depth estimation in an outdoor setting is useful as it is an appearance invariant feature (similar to optical flow) which are known to be helpful in activity recognition. ADR shows that using simulation, monocular outdoor depth estimation is possible without installing costly depth sensors.

The input space $\mathcal{X}$ consists of images with resolution 1920×1080 and the output space $\mathcal{Y}$ is the space of depth images of resolution 454×256, where each pixel corresponds to the density quantized in 80 bins. Here $\theta \in \Theta$ (similar to object detection) is a list of object attributes in the image. These attributes specify the location and type of the object in the image. Fig. 6.3 shows labeled samples from source domain.

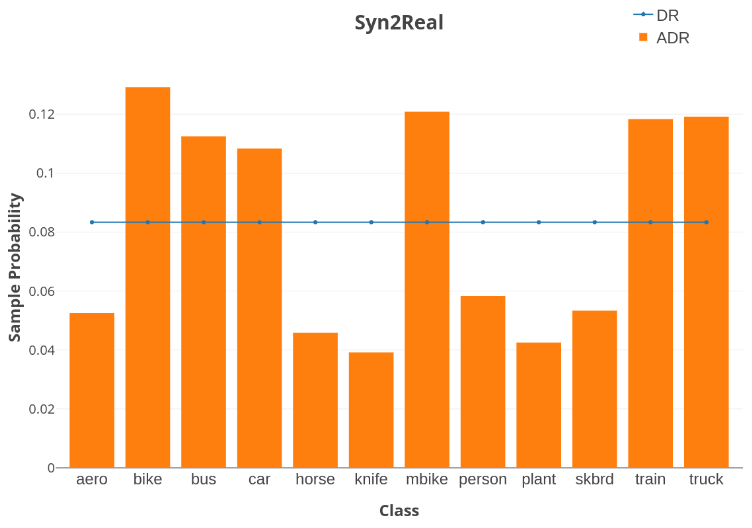


Figure 6.2: ADR π_ω after generating 100k images on Syn2Real Image Classification task. ADR increases the sample generation probability of confusing classes like {bike, mbike} and {bus, car, truck, train} whereas easier classes like {knife, plant, aero} see a decrease in generation probability.

ADR Setup: The setup is similar to object detection but we do not use distractors for this task and only choose between person or car object. The reward is again computed per cell and is the negative of the average cross entropy loss using the learner h's predictions.

The learner h is implemented as an FCN [22] with a ResNet101 [12] as the backbone architecture. h classifies each pixel into one of the bins at a coarse resolution of 454×256. The batch size is set to 2 along with SGD optimizer with a linearly decaying learning rate. We train a separate model for each scene in VIRAT.

Results: Figure 6.4 shows qualitative results for monocular depth estimation. h very quickly learns the depth profile of the background as it is almost constant for all the source domain. π_ω attempts to make it harder for h to predict the depth profile of the foreground object by minimizing its pixels in the image. This results into increasing the spawn probability of cells away from the camera. As a result, our task model trained using ADR even accurately predicts the depth profile of small foreground objects like people.

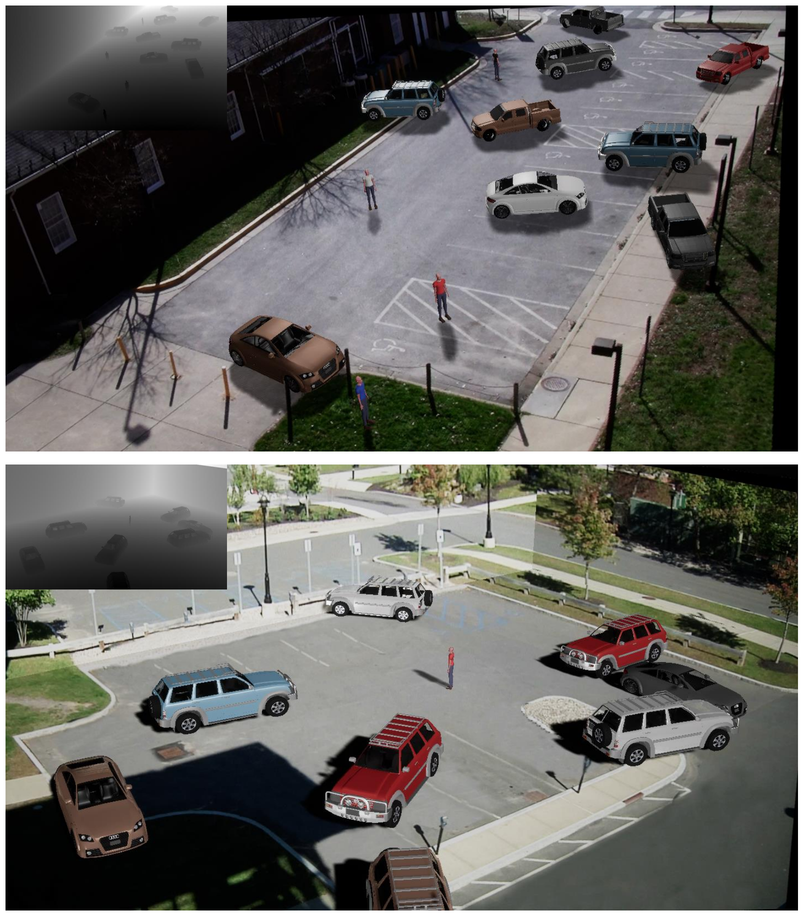


Figure 6.3: Source data for VIRAT scenes generated using ADR with the ground truth depth map.

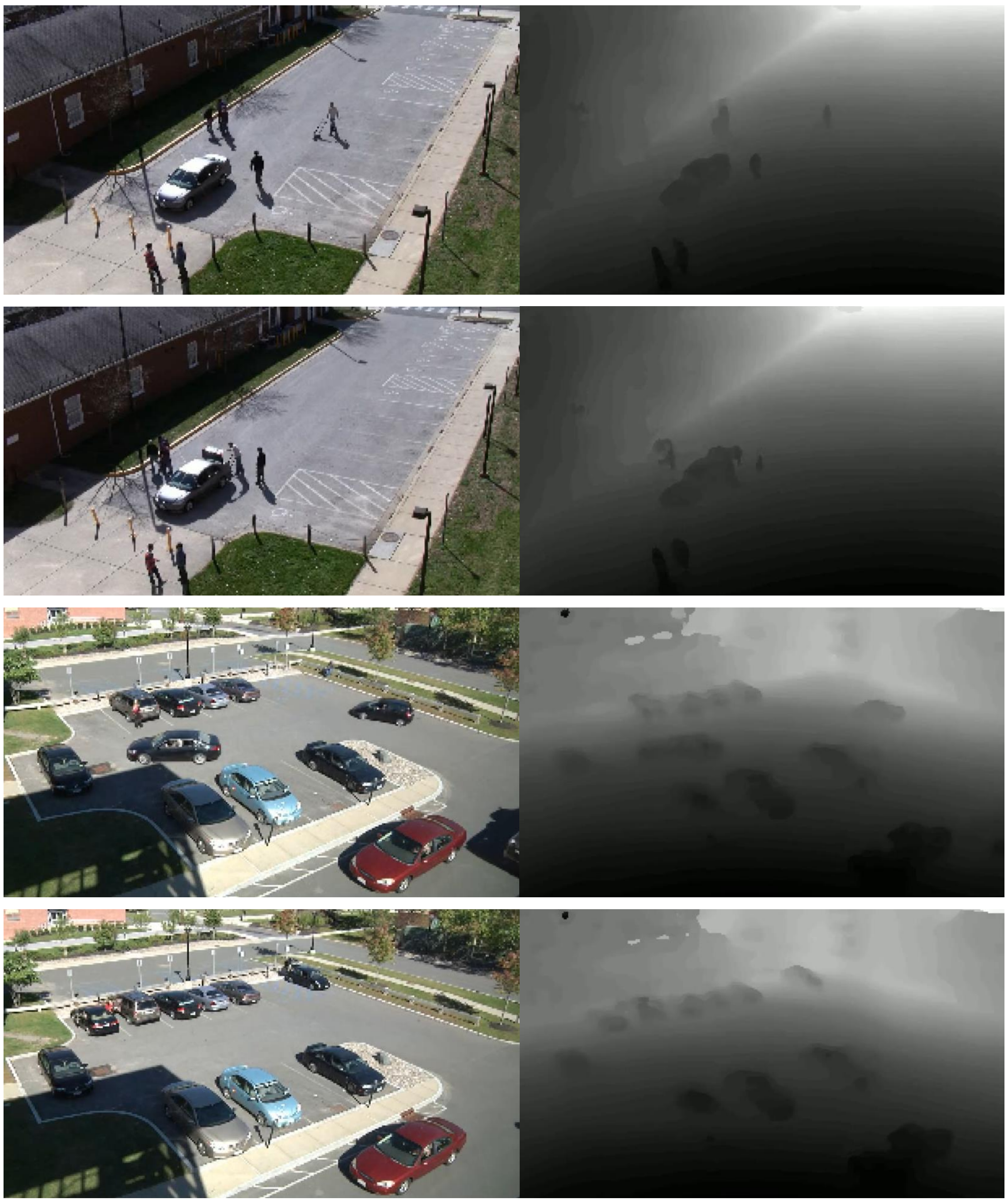


Figure 6.4: Depth estimation output from an FCN trained on data generated by ADR for VIRAT

Bibliography

[1] S. Ben-David, J. Blitzer, K. Crammer, A. Kulesza, F. Pereira, and J. W. Vaughan. A theory of learning from different domains. *Machine learning*, 79(1-2):151–175, 2010.

[2] S. Ben-David, J. Blitzer, K. Crammer, and F. Pereira. Analysis of representations for domain adaptation. In *Advances in neural information processing systems*, pages 137–144, 2007.

[3] C. K. Birdsall and A. B. Langdon. *Plasma physics via computer simulation*. CRC press, 2004.

[4] P. G. Bojan Pepik, Michael Stark and B. Schiele. Teaching 3d geometry to deformable part models. *CVPR*, 2012.

[5] G. Csurka. Domain adaptation for visual applications: A comprehensive survey. *arXiv preprint arXiv:1702.05374*, 2017.

[6] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[7] A. Dosovitskiy and V. Koltun. Learning to act by predicting the future. *arXiv preprint arXiv:1611.01779*, 2016.

[8] R. O. Dror, R. M. Dirks, J. Grossman, H. Xu, and D. E. Shaw. Biomolecular simulation: a computational microscope for molecular biology. *Annual review of biophysics*, 41:429–452, 2012.

[9] G. French, M. Mackiewicz, and M. Fisher. Self-ensembling for domain adaptation. *arXiv preprint arXiv:1706.05208*, 2017.

[10] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. Vision meets robotics: The KITTI dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013.

[11] H. Hattori, N. Lee, V. N. Boddeti, F. Beainy, K. M. Kitani, and T. Kanade. Synthesizing a scene-specific pedestrian detector and pose estimator for static video surveillance. *International Journal of Computer Vision*, 2018.

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

[13] M. Hejrati and D. Ramanan. Analysis by synthesis: 3d object recognition by object reconstruction. *CVPR*, 2014.

[14] S. James, A. J. Davison, and E. Johns. Transferring end-to-end visuomotor control from simulation to real world for a multi-stage task. *arXiv preprint arXiv:1707.02267*, 2017.

[15] J. Johnson, B. Hariharan, L. van der Maaten, L. Fei-Fei, C. L. Zitnick, and R. Girshick. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, pages 1988–1997. IEEE, 2017.

[16] R. Khirodkar, D. Yoo, and K. M. Kitani. Domain randomization for scene-specific car detection and pose estimation. *arXiv preprint arXiv:1811.05939*, 2018.

[17] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[18] V. Lepetit, F. Moreno-Noguer, and P. Fua. Epnp: An accurate o (n) solution to the pnp problem. *International journal of computer vision*, 81(2):155, 2009.

[19] J. Liao and A. Berg. Sharpening jensen's inequality. *The American Statistician*, pages 1–4, 2018.

[20] T.-Y. Lin, P. Dollár, R. B. Girshick, K. He, B. Hariharan, and S. J. Belongie. Feature pyramid networks for object detection. In *CVPR*, volume 1, page 4, 2017.

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

[22] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[23] M. Long, Y. Cao, J. Wang, and M. I. Jordan. Learning transferable features with deep adaptation networks. *arXiv preprint arXiv:1502.02791*, 2015.

[24] S. Lyu, M.-C. Chang, D. Du, L. Wen, H. Qi, Y. Li, Y. Wei, L. Ke, T. Hu, M. Del Coco, et al. Ua-detrac 2017: Report of avss2017 & iwt4s challenge on advanced traffic monitoring. In *Advanced Video and Signal Based Surveillance (AVSS), 2017 14th IEEE International Conference on*, pages 1–7. IEEE, 2017.

[25] D. Mahajan, R. Girshick, V. Ramanathan, K. He, M. Paluri, Y. Li, A. Bharambe, and L. van der Maaten. Exploring the limits of weakly supervised pretraining. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 181–196, 2018.

[26] A. Y. Michel Dhome and J.-M. Lavest. Determination of the pose of an articulated object from a single perspective view. *BMVC*, 1993.

[27] A. Mousavian, D. Anguelov, J. Flynn, and J. Košecká. 3d bounding box estimation using deep learning and geometry. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, pages 5632–5640. IEEE, 2017.

[28] Y. Movshovitz-Attias, Y. Sheikh, V. N. Boddeti, and Z. Wei. 3d pose-by-detection of vehicles via discriminatively reduced ensembles of correlation filters. In *BMVC*, 2014.

[29] S. Oh, A. Hoogs, A. Perera, N. Cuntoor, C.-C. Chen, J. T. Lee, S. Mukherjee, J. Aggarwal, H. Lee, L. Davis, E. Swears, X. Wang, Q. Ji, K. Reddy, M. Shah, C. Vondrick, H. Pirsiavash, D. Ramanan, J. Yuen, A. Torralba, B. Song, A. Fong, A. Roy-Chowdhury, and M. Desai. A large-scale benchmark dataset for event recognition in surveillance video. *IEEE Comptuer Vision and Pattern Recognition (CVPR)*, 2011.

[30] M. Ozuysal, V. Lepetit, and P. Fua. Pose estimation for category specific multiview object localization. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

[31] S. J. Pan, Q. Yang, et al. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2010.

[32] Y. I. Parish and P. Müller. Procedural modeling of cities. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 301–308. ACM, 2001.

[33] V. M. Patel, R. Gopalan, R. Li, and R. Chellappa. Visual domain adaptation: A survey of recent advances. *IEEE signal processing magazine*, 32(3):53–69, 2015.

[34] X. Peng, B. Usman, K. Saito, N. Kaushik, J. Hoffman, and K. Saenko. Syn2real: A new benchmark forsynthetic-to-real visual domain adaptation. *arXiv preprint arXiv:1806.09755*, 2018.

[35] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. *arXiv preprint arXiv:1710.06537*, 2017.

[36] L. Pinto, M. Andrychowicz, P. Welinder, W. Zaremba, and P. Abbeel. Asymmetric actor critic for image-based robot learning. *arXiv preprint arXiv:1710.06542*, 2017.

[37] W. Qiu and A. Yuille. Unrealcv: Connecting computer vision to unreal engine. In *European Conference on Computer Vision*, pages 909–916. Springer, 2016.

[38] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[39] F. Sadeghi and S. Levine. Cad2rl: Real single-image flight without a single real image. *arXiv preprint arXiv:1611.04201*, 2016.

[40] S. Savarese and L. Fei-Fei. 3d generic object categorization, localization and pose estimation. 2007.

[41] R. J. Serfling. Probability inequalities for the sum in sampling without replacement. *The Annals of Statistics*, pages 39–48, 1974.

[42] A. Shrivastava, T. Pfister, O. Tuzel, J. Susskind, W. Wang, and R. Webb. Learning from simulated and unsupervised images through adversarial training. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[43] B. Sun, J. Feng, and K. Saenko. Return of frustratingly easy domain adaptation. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[44] B. Sun and K. Saenko. From virtual to reality: Fast adaptation of virtual object detectors to real domains. *BMVC*, 2014.

[45] M. Sundermeyer, Z. Marton, M. Durner, and R. Triebel. Implicit 3d orientation learning for 6d object detection from rgb images. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 699–715, 2018.

[46] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, pages 1057–1063, 2000.

[47] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*, pages 23–30. IEEE, 2017.

[48] J. Tremblay, A. Prakash, D. Acuna, M. Brophy, V. Jampani, C. Anil, T. To, E. Cameracci, S. Boochoon, and S. Birchfield. Training deep networks with synthetic data: Bridging the reality gap by domain randomization. *arXiv preprint arXiv:1804.06516*, 2018.

[49] E. Tzeng, J. Hoffman, K. Saenko, and T. Darrell. Adversarial discriminative domain adaptation. In *Computer Vision and Pattern Recognition (CVPR)*, volume 1, page 4, 2017.

[50] M. Wang and W. Deng. Deep visual domain adaptation: A survey. *Neurocomputing*, 2018.

[51] R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

[52] A. Z. L. S. X. Zhang, Y. Fu and G. Agam. Learning classifiers from synthetic data using a multichannel autoencoder. *arXiv:1503.03163*, 2015.

[53] Y. Xiang, R. Mottaghi, and S. Savarese. Beyond pascal: A benchmark for 3d object detection in the wild. In *Applications of Computer Vision (WACV), 2014 IEEE Winter Conference on*, pages 75–82. IEEE, 2014.

[54] Z. W. Yair Movshovitz-Attias, Vishnu Naresh Boddeti and Y. Sheikh. 3d pose-by-detection of vehicles via discriminatively reduced ensembles of correlation filters. *BMVC*, 2014.

[55] V. L. Yaroslav Ganin. Unsupervised domain adaptation by backpropagation. *ICML*, 2015.

[56] L. Žlajpah. Simulation in robotics. *Mathematics and Computers in Simulation*, 79(4):879–897, 2008.