Train in Germany, Test in The USA: Making 3D Object Detectors Generalize

Yan Wang*[1] Xiangyu Chen*[1] Yurong You[1] Li Erran Li[2,3]
Bharath Hariharan[1] Mark Campbell[1] Kilian Q. Weinberger[1] Wei-Lun Chao[4]
[1]Cornell University [2]Scale AI [3]Columbia University [4]The Ohio State University
{yw763, xc429, yy785, bh497, mc288, kqw4}@cornell.edu erranlli@gmail.com chao.209@osu.edu

Abstract

In the domain of autonomous driving, deep learning has substantially improved the 3D object detection accuracy for LiDAR and stereo camera data alike. While deep networks are great at generalization, they are also notorious to overfit to all kinds of spurious artifacts, such as brightness, car sizes and models, that may appear consistently throughout the data. In fact, most datasets for autonomous driving are collected within a narrow subset of cities within one country, typically under similar weather conditions. In this paper we consider the task of adapting 3D object detectors from one dataset to another. We observe that naïvely, this appears to be a very challenging task, resulting in drastic drops in accuracy levels. We provide extensive experiments to investigate the true adaptation challenges and arrive at a surprising conclusion: the primary adaptation hurdle to overcome are differences in car sizes across geographic areas. A simple correction based on the average car size yields a strong correction of the adaptation gap. Our proposed method is simple and easily incorporated into most 3D object detection frameworks. It provides a first baseline for 3D object detection adaptation across countries, and gives hope that the underlying problem may be more within grasp than one may have hoped to believe. Our code is available at https://github.com/cxy1997/3D_adapt_auto_driving.

1. Introduction

Autonomous cars need to accurately detect and localize vehicles and pedestrians in 3D to drive safely. As such, the past few years have seen a flurry of interest on the problem of 3D object detection, resulting in large gains in accuracy on the KITTI benchmark [11, 14, 15, 16, 18, 19, 28, 29, 30, 31, 32, 33, 34, 37, 40, 41, 52, 53, 54, 51, 61, 62, 63, 64, 65, 68, 69]. However, in the excitement this has garnered, it has often been forgotten that KITTI is a fairly small (∼15K scenes) object detection dataset obtained from a narrow domain: it was collected using a fixed sensing apparatus by driving through a mid-sized German city and the German countryside, in clear weather, during the day. Thus, the 3D object detection algorithms trained on KITTI may have picked up all sorts of biases: they may expect the road to be visible or the sky to be blue. They may identify only certain brands of cars, and might have even over-fit to the idiosyncrasies of German drivers and pedestrians. Carrying

Figure 1: **Datasets.** We show frontal view images (left) and the corresponding LiDAR signals (right) from the bird's-eye view for five datasets: KITTI [18, 19], Argoverse [7], nuScenes [4], Lyft [25], and Waymo [3]. These datasets not only capture scenes at different geo-locations, but also use different LiDAR models, making generalizing 3D object detectors a challenging problem.

* Equal contributions

these biases over to a new environment in a different part of the world might cause the object detector to miss cars or pedestrians, with devastating consequences [1].

It is, therefore, crucial that we (a) understand the biases that our 3D object detectors are picking up before we deploy them in safety-critical applications, and (b) identify techniques to mitigate these biases. Our goal in this paper is to address both of these challenges.

Our first goal is to understand if any biases have crept into current 3D object detectors. For this, we leverage multiple recently released datasets with similar types of sensors to KITTI [18, 19] (cameras and LiDAR) and with 3D annotations, each of them collected in different cities [3, 4, 7, 25] (see Figure 1 for an illustration). Interestingly, they are also recorded with different sensor configurations (*i.e.*, the LiDAR and camera models as well as their mounting arrangements can be different). We first train two representative LiDAR-based 3D object detectors (PIXOR [63] and POINTRCNN [52]) on each dataset and test on the others. We find that when tested on a different dataset, 3D object detectors fail dramatically: a detector trained on KITTI performs **36 percent worse** on Waymo [3] compared to the one trained on Waymo. This indicates that the detector has indeed *over-fitted* to its training domain.

What domain differences are causing such catastrophic failure? One can think of many possibilities. There may be differences in low-level statistics of the images. The LiDAR sensors might have more or fewer beams, and may be oriented differently. But the differences can also be in the *physical world* being sensed. There may be differences in the number of vehicles, their orientation, and also their sizes and shapes. We present an extensive analysis of these potential biases that points to one major issue — statistical differences in the sizes and shapes of cars.

In hindsight, this difference makes sense. The best selling car in the USA is a 5-meter long truck (Ford F-series) [2], while the best selling car in Germany is a 4-meter long compact car (Volkswagen Golf[1]). Because of such differences, cars in KITTI tend to be smaller than cars in other datasets, a bias that 3D object detectors happily learn. As a counter to this bias, we propose an extremely simple approach that leverages aggregate statistics of car sizes (*i.e.*, mean) to correct for this bias, *in both the output annotations and the input signals*. Such statistics might be acquired from the department of motor vehicles, or car sales data. This single correction results in a massive improvement in cross-dataset performance, raising the 3D easy part average precision by 41.4 points and results in a much more robust 3D object detector.

Taken together, our contributions are two-fold:

- We present an extensive evaluation of the domain differences between self-driving car environments and how they impact 3D detector performance. Our results suggest a single core issue: size statistics of cars in different locations.
- We present a simple and effective approach to mitigate this issue by using easily obtainable aggregate statistics of car sizes, and show dramatic improvements in cross-dataset performance as a result.

Based on our results, we recommend that vision researchers and self-driving car companies alike be cognizant of such domain differences for large-scale deployment of 3D detection systems.

[1] https://www.best-selling-cars.com/germany/2019-q1-germany-best-selling-car-brands-and-models/

2. Related Work

We review 3D object detection for autonomous driving, and domain adaptation for 2D segmentation and detection in street scenes.

LiDAR-based detection. Most existing techniques of 3D object detection use LiDAR (sometimes with images) as the input signal, which provides accurate 3D points of the surrounding environment. The main challenge is thus on properly encoding the points so as to predict point labels or draw bounding boxes in 3D to locate objects. Frustum PointNet [41] applies PointNet [42, 43] to each frustum proposal from a 2D object detector; POINTRCNN [52] learns 3D proposals from PointNet++ features [43]. MV3D [11] projects LiDAR points into frontal and bird's-eye views (BEV) to obtain multi-view features; PIXOR [63] and LaserNet [37] show that properly encoding features in one view is sufficient to localize objects. VoxelNet [69] and PointPillar [30] encode 3D points into voxels and extracts features by 3D convolutions and PointNet. UberATG-ContFuse [34] and UberATG-MMF [33] perform continuous convolutions [56] to fuse visual and LiDAR features.

Image-based detection. While providing accurate 3D points, LiDAR sensors are notoriously expensive. A 64-line LiDAR (*e.g.*, the one used in KITTI [19, 18]) costs around $\$75,000$ (US dollars). As an alternative, researchers have also been investigating purely image-based 3D detection. Existing algorithms are largely built upon 2D object detection [45, 20, 35], imposing extra geometric constraints [6, 8, 38, 59] to create 3D proposals. [9, 10, 39, 60] apply stereo-based depth estimation to obtain 3D coordinates of each pixel. These 3D coordinates are either entered as additional input channels into a 2D detection pipeline, or used to extract hand-crafted features. The recently proposed pseudo-LiDAR [58, 44, 66] combined stereo-based depth estimation with LiDAR-based detection, converting the depth map into a 3D point cloud and processing it exactly as LiDAR signal. The pseudo-LiDAR framework has largely improved image-based detection, yet a notable gap is still remained compared to LiDAR. In this work, we

therefore focus on LiDAR-based object detectors.

Domain adaptation. (Unsupervised) domain adaptation has also been studied in autonomous driving scenes, but mainly for the tasks of 2D semantic segmentation [13, 22, 24, 36, 48, 49, 50, 55, 67, 73] and 2D object detection [5, 12, 21, 23, 26, 27, 46, 47, 57, 72, 71]. The common setting is to adapt a model trained from one labeled source domain (*e.g.*, synthetic images) to an unlabeled target domain (e.g., real images). The domain difference is mostly from the input signal (*e.g.*, image styles), and many algorithms have built upon adversarial feature matching and style transfer [17, 22, 70] to minimize the domain gap in the input or feature space. Our work contrasts these methods by studying 3D object detection. We found that, the output space (*e.g.*, car sizes) can also contribute to the domain gap; properly leveraging the statistics of the target domain can largely improve the model's generalization ability.

3. Datasets

We review KITTI [18, 19] and introduce the other four datasets used in our experiments: Argoverse [7], Lyft [25], nuScenes [4], and Waymo [3]. We focus on data related to 3D object detection. All the datasets provide ground-truth 3D bounding box labels for several kinds of objects. We summarize the five datasets in detail in Table 1.

KITTI. The KITTI object detection benchmark [18, 19] contains $7,481$ (left) images for training and $7,518$ images for testing. The training set is further separated into $3,712$ training and $3,769$ validation images as suggested by [9]. All the scenes are pictured around Karlsruhe, Germany in clear weather and day time. For each (left) image, KITTI provides its corresponding 64-beam Velodyne LiDAR point cloud and the right stereo image.

Argoverse. The Argoverse dataset [7] is collected around Miami and Pittsburgh, USA in multiple weathers and during different times of a day. It provides images from stereo cameras and another seven cameras that cover 360° information. It also provides 64-beam LiDAR point clouds captured by two 32-beam Velodyne LiDAR sensors stacked vertically. We extracted synchronized frontal-view images and corresponding point clouds from the original Argoverse dataset, with a timestamp tolerance of 51 ms between LiDAR sweeps and images. The resulting dataset we use contains $13,122$ images for training, $5,015$ images for validation, $4,168$ images for testing.

nuScenes. The nuScenes dataset [4] contains $28,130$ training and $6,019$ validation images. We treat the validation images as test images, and re-split and subsample the $28,130$ training images into $11,040$ training and $3,026$ validation images. The scenes are pictured around Boston, USA and Singapore in multiple weathers and during different times of a day. For each image, nuScenes provides the point cloud captured by a 32-beam roof LiDAR. It also provides images from another five cameras that cover 360° information.

Lyft. The Lyft Level 5 dataset [25] contains $18,634$ frontal-view images and we separate them into $12,599$ images for training, $3,024$ images for validation, $3,011$ images for testing. The scenes are pictured around Palo Auto, USA in clear weathers and during day time. For each image, Lyft provides the point cloud captured by a 40 (or 64)-beam roof LiDAR and two 40-beam bumper LiDAR sensors. It also provides images from another five cameras that cover 360° information and one long-focal-length camera.

Waymo. The Waymo dataset [3] contains $122,000$ training, $30,407$ validation, and $40,077$ test images and we subsample them into $12,000$, $3,000$, and $3,000$, respectively. The scenes are pictured at Phoenix, Mountain View, and San Francisco in multiple weathers and at multiple times of a day. For each image, Waymo provides the combined point cloud captured by five LiDAR sensors (one on the roof). It also provides images from another four cameras.

Data format. A non-negligible difficulty in conducting cross-dataset analysis lies in the differences of data formats. *Considering that most existing algorithms are developed using the KITTI format, we transfer all the other four datasets into its format.* See the Supplementary Material for details.

4. Experiments and Analysis

4.1. Setup

3D object detection algorithms. We apply two LiDAR-based models POINTRCNN [52] and PIXOR [63] to detect objects in 3D by outputting the surrounding 3D bounding boxes. PIXOR represents LiDAR point clouds by 3D tensors after voxelization, while POINTRCNN applies PointNet++ [43] to extract point-wise features. Both methods do not rely on images. We train both models on the five 3D object detection datasets. POINTRCNN has two subnetworks, the region proposal network (RPN) and region-CNN (RCNN), that are trained separately. The RPN is trained first, for 200 epochs with batch size 16 and learning rate 0.02. The RCNN is trained for 70 epochs with batch size 4 and learning rate 0.02. We use online ground truth boxes augmentation, which copies object boxes and inside points from one scene to the same locations in another scene. For PIXOR, we train it with batch size 4 and initial learning rate 5×10^{-5}, which will be decreased 10 times on the 50th and 80th epoch. We do randomly horizontal flip and rotate during training.

Metric. We follow KITTI to evaluate object detection in 3D and the bird's-eye view (BEV). We focus on the *Car* category, which has been the main focus in existing works. We report average precision (AP) with the IoU thresholds at 0.7: a car is correctly detected if the intersection over union (IoU) with the predicted 3D box is larger than 0.7. We denote AP for the 3D and BEV tasks by AP_{3D} and AP_{BEV}.

Table 1: Dataset overview. We focus on their properties related to frontal-view images, LiDAR, and 3D object detection. The dataset size refers to the number of synchronized (image, LiDAR) pairs. For Waymo and nuScenes, we subsample the data. See text for details.

Dataset	Size	LiDAR Type	Beam Angles	Object Types	Rainy Weather	Night Time
KITTI [18, 19]	$14,999$	1×64-beam	$[-24^\circ, 4^\circ]$	8	No	No
Argoverse [7]	$22,305$	2×32-beam	$[-26^\circ, 25^\circ]$	17	No	Yes
nuScenes [4]	$34,149$	1×32-beam	$[-16^\circ, 11^\circ]$	23	Yes	Yes
Lyft [25]	$18,634$	1×40 or $64 + 2 \times 40$-beam	$[-29^\circ, 5^\circ]$	9	No	No
Waymo [3]	$192,484$	$1 \times 64 + 4 \times 200$-beam	$[-18^\circ, 2^\circ]$	4	Yes	Yes

KITTI evaluates three cases: *Easy*, *Moderate*, and *Hard*. Specifically, it labels each ground truth box with four levels (0 to 3) of occlusion / truncation. The *Easy* case contains level-0 cars whose bounding box heights in 2D are larger than 40 pixels; the *Moderate* case contains level-$\{0, 1\}$ cars whose bounding box heights in 2D are larger than 25 pixels; the *Hard* case contains level-$\{0, 1, 2\}$ cars whose bounding box heights in 2D are larger than 25 pixels. The heights are meant to separate cars by their depths with respect to the observing car. Nevertheless, since different datasets have different image resolutions, such criteria might not be aligned across datasets. We thus replace the constraints of "larger than $40, 25$ pixels" by "within $30, 70$ meters". We further evaluate cars of level-$\{0, 1, 2\}$ within three depth ranges: $0-30$, $30-50$, and $50-70$ meters, following [63].

We mainly report and discuss results of PointRCNN on the *validation* set in the main paper. We report results of PIXOR in the Supplementary Material.

4.2. Results within each dataset

We first evaluate if existing 3D object detection models that have shown promising results on the KITTI benchmark can also be learned and perform well on newly released datasets. We summarize the results in Table 2: the rows are the source domains that a detector is trained on, and the columns are the target domains the detector is being tested on. The **bold** font indicates the within domain performance (*i.e.*, training and testing using the same dataset).

We see that PointRCNN works fairly well on the KITTI, Lyft, and Waymo datasets, for all the easy, moderate, and hard cases. The results get slightly worse on Argoverse, and then nuScenes. We hypothesize that this may result from the relatively poor LiDAR input: nuScenes has only 32 beams; while Argoverse has 64 beams, every two of them are very close due to the configurations that the signal is captured by two stacked LiDAR sensors.

We further analyze at different ranges in Table 2 (bottom). We see a drastic drop on Argoverse and nuScenes for the far-away ranges, which supports our hypothesis: with fewer beams, the far-away objects can only be rendered by very sparse LiDAR points and thus are hard to detect. We also see poor accuracies at $50-70$ meters on KITTI, which may result from very few labeled training instances there.

Overall, both 3D object detection algorithms work fairly well when being trained and tested using the same dataset, as long as the input sensor signal is of high quality and the labeled instances are sufficient.

4.3. Results across datasets

We further experiment with generalizing a trained detector across datasets. We indicate the best result per column and per setting by red fonts and the worst by blue fonts.

We see a clear trend of performance drop. For instance, the PointRCNN model trained on KITTI achieves only 45.2% AP_{BEV} (Moderate) on Waymo, lower than the model trained on Waymo by over 40%. The gap becomes even larger in AP_{3D}: the same KITTI model attains only 11.9% AP_{3D}, while the Waymo model attains 85.3%. We hypothesize that the car height is hard to get right. In terms of the target (test) domain, Lyft and Waymo suffer the least drop if the detector is trained from the other datasets, followed by Argoverse. KITTI and nuScenes suffer the most drop, which might result from their different geo-locations (one is from Germany and the other contains data from Singapore). The nuScenes dataset might also suffer from its relatively fewer beams in the input and other models may therefore not be able to apply. By considering different ranges, we also find that the deeper the range is, the bigger the drop is.

In terms of the source (training) domain, we see that the detector trained on KITTI seems to be the worst to transfer to others. In every 5×1 block that is evaluated on a single dataset in a single setting, the KITTI model is mostly outperformed by others. Surprisingly, nuScenes model can perform fairly well when being tested on the other datasets: the results are even higher than on its own. We thus have two arguments: The quality of sensors is more important in testing than in training; KITTI data (*e.g.*, car styles, time, and weather) might be too limited or different from others and therefore cannot transfer well to others. In the following subsections, we provide detailed analysis.

4.4. Analysis of domain idiosyncrasies

Table 2 and subsection 4.3 reveal drastic accuracy drops in generalizing 3D object detectors across datasets (domains). We hypothesize that there exist significant idiosyncrasies in each dataset. In particular, Figure 1 shows that the images and point clouds are quite different across datasets. One one hand, different datasets are collected by cars of dif-

Table 2: **3D object detection across multiple datasets** (evaluated on the validation sets). We report average precision (AP) of the *Car* category in bird's-eye view (AP_{BEV}) and 3D (AP_{3D}) at IoU = 0.7, using the POINTRCNN detector [52]. We report results at different difficulties (following the KITTI benchmark, but we replace the 40, 25, 25 pixel thresholds on 2D bounding boxes with 30, 70, 70 meters on object depths, for *Easy*, *Moderate*, and *Hard* cases, respectively) and different depth ranges (using the same truncation and occlusion thresholds as KITTI *Hard* case). The results show a significant performance drop in cross-dataset inference. We indicate the best generalization results per column and per setting by red fonts and the worst by blue fonts. We indicate in-domain results by bold fonts.

Setting	Source\Target	KITTI	Argoverse	nuScenes	Lyft	Waymo
Easy	KITTI	**88.0 / 82.5**	55.8 / 27.7	47.4 / 13.3	81.7 / 51.8	45.2 / 11.9
	Argoverse	69.5 / 33.9	**79.2 / 57.8**	52.5 / 21.8	86.9 / 67.4	83.8 / 40.2
	nuScenes	49.7 / 13.4	73.2 / 21.8	**73.4 / 38.1**	89.0 / 38.2	78.8 / 36.7
	Lyft	74.3 / 39.4	77.1 / 45.8	63.5 / 23.9	**90.2 / 87.3**	87.0 / 64.7
	Waymo	51.9 / 13.1	76.4 / 42.6	55.5 / 21.6	87.9 / 74.5	**90.1 / 85.3**
Moderate	KITTI	**80.6 / 68.9**	44.9 / 22.3	26.2 / 8.3	61.8 / 33.7	43.9 / 12.3
	Argoverse	56.6 / 31.4	**69.9 / 44.2**	27.6 / 11.8	66.6 / 42.1	72.3 / 35.1
	nuScenes	39.8 / 10.7	56.6 / 17.1	**40.7 / 21.2**	71.4 / 25.0	68.2 / 30.8
	Lyft	61.1 / 34.3	62.5 / 35.3	33.6 / 12.3	**83.7 / 65.5**	77.6 / 53.2
	Waymo	45.8 / 13.2	64.4 / 29.8	28.9 / 13.7	74.2 / 53.8	**85.9 / 67.9**
Hard	KITTI	**81.9 / 66.7**	42.5 / 22.2	24.9 / 8.8	57.4 / 34.2	41.5 / 12.6
	Argoverse	58.5 / 33.3	**69.9 / 42.8**	26.8 / 14.5	64.4 / 42.7	68.5 / 36.8
	nuScenes	39.6 / 10.1	53.3 / 16.7	**40.2 / 20.5**	67.7 / 25.7	66.9 / 29.0
	Lyft	60.7 / 33.9	62.9 / 35.9	30.6 / 11.7	**79.3 / 65.5**	77.0 / 53.9
	Waymo	46.3 / 12.6	61.6 / 29.0	28.4 / 14.1	74.1 / 54.5	**80.4 / 67.7**
0-30m	KITTI	**88.8 / 84.9**	58.4 / 34.7	47.9 / 14.9	77.8 / 54.2	48.0 / 14.0
	Argoverse	74.2 / 46.8	**83.3 / 63.3**	55.3 / 26.9	87.7 / 69.5	85.7 / 44.4
	nuScenes	50.7 / 13.9	73.7 / 26.0	**73.2 / 42.8**	89.1 / 43.8	79.8 / 43.4
	Lyft	75.1 / 45.2	81.0 / 54.0	61.6 / 25.4	**90.4 / 88.5**	88.6 / 70.9
	Waymo	56.8 / 15.0	80.6 / 48.1	57.8 / 24.0	88.4 / 76.2	**90.4 / 87.2**
30m-50m	KITTI	**70.2 / 51.4**	46.5 / 19.0	9.8 / 4.5	60.1 / 34.5	50.5 / 21.4
	Argoverse	33.9 / 11.8	**72.2 / 39.5**	9.5 / 9.1	65.9 / 39.1	75.9 / 42.1
	nuScenes	24.1 / 3.8	46.3 / 6.4	**17.1 / 4.1**	70.1 / 18.9	69.4 / 29.2
	Lyft	39.3 / 16.6	59.2 / 21.8	11.2 / 9.1	**83.8 / 62.7**	79.4 / 55.5
	Waymo	31.7 / 9.3	58.0 / 18.8	9.9 / 9.1	74.5 / 51.4	**87.5 / 68.8**
50m-70m	KITTI	**28.8 / 12.0**	9.2 / 3.0	1.1 / 0.0	33.2 / 9.6	27.1 / 12.0
	Argoverse	10.9 / 1.3	**29.9 / 6.9**	0.5 / 0.0	35.1 / 14.5	46.2 / 23.0
	nuScenes	6.5 / 1.5	15.2 / 2.3	**9.1 / 9.1**	41.8 / 5.3	37.9 / 15.2
	Lyft	13.6 / 4.6	23.1 / 3.9	1.1 / 0.0	**62.7 / 33.1**	54.6 / 27.5
	Waymo	5.6 / 1.8	26.9 / 5.6	0.9 / 0.0	50.8 / 21.3	**63.5 / 41.1**

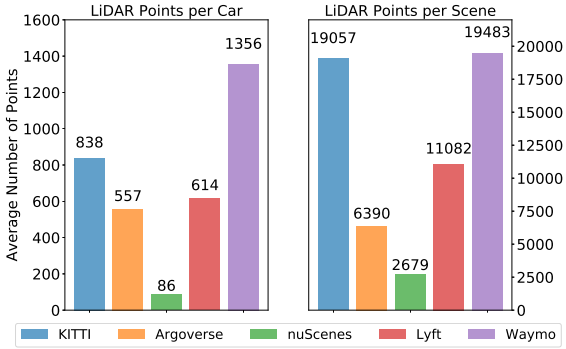

Figure 2: The average numbers of 3D points per car (left) and per scene (right). We only include points within the frontal-view camera view and cars whose depths are within 70 meters.

ferent sensor configurations. For example, nuScenes uses a single 32-beam LiDAR; the point clouds are thus sparser than the other datasets. On the other hand, these datasets are collected at different locations; the environments and the foreground object styles may also be different.

To provide a better understanding, we compute the average number of LiDAR points per scene and per car (using the ground-truth 3D bounding box) in Figure 2. We see a large difference: Waymo has ten times of points per car than nuScenes[2]. We further analyze the size of bounding boxes per car. Figure 3 shows the histograms of each dataset. We again see mismatches between different datasets: KITTI seems to have the smallest box sizes while Waymo has the largest. We conduct an analysis and find that most of the bounding boxes tightly contain the points of cars inside. We, therefore, argue that this difference of box sizes is related to the car styles captured in different datasets.

[2]We note that POINTRCNN applies point re-sampling so that every scene (in RPN) and object proposal (in RCNN) will have the same numbers of input points while PIXOR applies voxelization. Both operations can reduce but cannot fully resolve point cloud differences across domains.

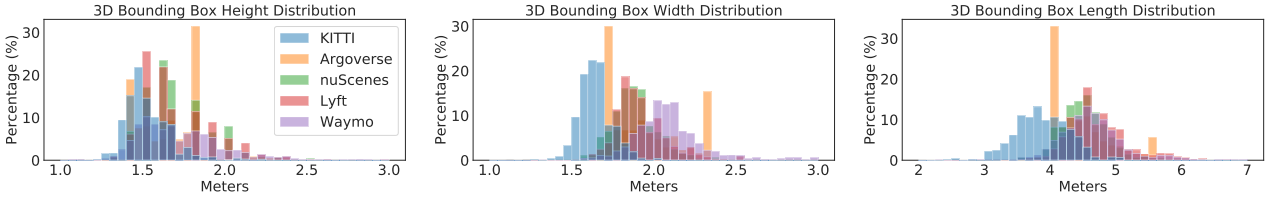

Figure 3: Car size statistics of different datasets.

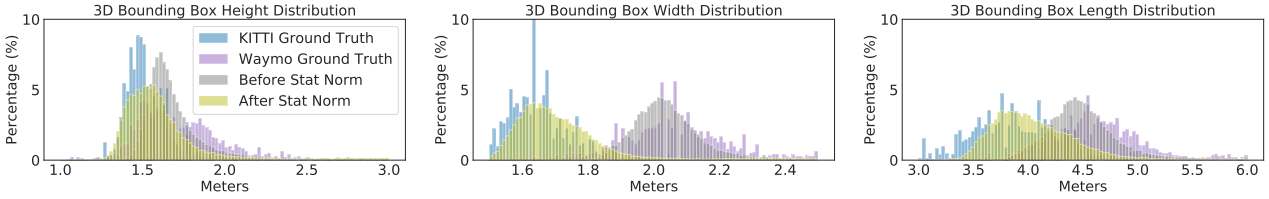

Figure 4: Sizes of detected bounding boxes before and after our Statistical Normalization (Stat Norm). The detector is trained on Waymo (w/o or w/ Stat Norm) and tested on KITTI. We also show the distribution of ground-truth box sizes in both datasets.

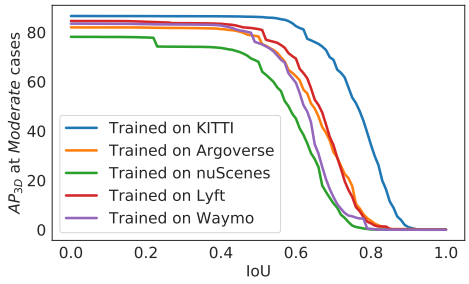

Figure 5: Car detection accuracy (AP_{3D} at *Moderate* cases) on KITTI, using POINTRCNN models trained on different datasets. We vary the IoU threshold from 0.0 to 1.0 (x-axis). The curves indicate that models trained on different datasets have similar detection abilities (converge at low IoU) but they differ in localization (diverge at high IoU).

4.5. Analysis of detector performance

So what are the idiosyncrasies that account for the majority of performance gap? There are two factors that can lead to an miss-detected car (*i.e.*, IoU < 0.7): the car might be entirely missed by the detector, or it is detected but poorly localized. To identify the main factor, we lower down the IoU threshold using KITTI as the target domain (see Figure 5). We observe an immediate increase in AP_{3D}, and the results become saturated when IoU is lower than 0.4. Surprisingly, POINTRCNN models trained from other datasets perform on a par with the model trained on KITTI. In other words, poor generalization resides primarily in localization.

We investigate one cause of mislocalization[3]: inaccurate box size. To this end, we replace the size of every detected car that has IoU > 0.2 to a ground-truth car with the corresponding ground-truth box size, while keeping its bottom center and rotation unchanged. We see an immediate performance boost in Table 3 (see the Supplementary Material for complete results across all pairs of datasets). In other words, the detector trained from one domain just cannot predict the car size right in the other domains. This observation correlates with our findings in Figure 3 that these datasets have different car sizes. By further analyzing the detected boxes (in Figure 4, we apply the detector trained from Waymo to KITTI), we find that the detector tends to predict box sizes that are similar to the ground-truth sizes in source domain, even though cars in the target domain are indeed physically smaller. We think this is because the detectors trained from the source data carry the learned bias to the target data.

[3]Mislocalization can result from wrong box centers, rotations, or sizes.

Table 3: Cross-dataset performance and gain (in parentheses) by assigning ground-truth box sizes to detected cars while keeping their centers and rotations unchanged. We report AP_{3D} of the *Car* category at IoU $= 0.7$, using POINTRCNN [52]. We show adaptation from KITTI to other datasets, and vice versa.

Setting	Dataset	From KITTI	To KITTI
Easy	Argoverse	65.7 (+38.0)	59.2 (+25.3)
	nuScenes	33.5 (+20.2)	63.9 (+50.5)
	Lyft	74.8 (+23.1)	58.4 (+19.0)
	Waymo	77.1 (+65.2)	78.2 (+65.1)
Moderate	Argoverse	50.9 (+28.6)	51.0 (+19.6)
	nuScenes	18.2 (+9.9)	47.3 (+36.6)
	Lyft	54.3 (+20.6)	49.4 (+15.1)
	Waymo	63.0 (+50.7)	60.6 (+47.4)
Hard	Argoverse	49.3 (+27.1)	52.5 (+19.2)
	nuScenes	17.7 (+8.9)	45.7 (+35.6)
	Lyft	53.0 (+18.8)	52.0 (+18.1)
	Waymo	59.1 (+46.5)	60.7 (+48.1)

5. Domain Adaptation Approaches

The poor performance due to mislocalization rather than misdetection opens the possibility of adapting a learned detector to a new domain with relatively smaller efforts. We investigate two scenarios: (1) a few labeled scenes (*i.e.*, point clouds with 3D box annotations) or (2) the car size

Table 4: **Improved 3D object detection across datasets** (evaluated on the validation sets). We report AP_{BEV}/ AP_{3D} of the *Car* category at IoU $= 0.7$, using POINTRCNN [52]. We investigate **(OT)** *output transformation* by directly adjusting the predicted box sizes, **(SN)** *statistical normalization*, and **(FS)** *few-shot fine-tuning* (with 10 labeled instances). We also include **(Direct)** directly applying the detectors trained on the source domain and **(Within)** applying the detectors trained on the target domain for comparison. We show adaption results from KITTI to other datasets, and vice versa. We mark the best result among Direct, OT, SN, and FS in red fonts, and worst in blue fonts.

		From KITTI (KITTI as the source; others as the target)					**To KITTI** (KITTI as the target; others as the source)				
Setting	Dataset	Direct	OT	SN	FS	Within	Direct	OT	SN	FS	Within
Easy	Argoverse	55.8 / 27.7	72.7 / 9.0	74.7 / 48.2	75.8 / 49.2	79.2 / 57.8	69.5 / 33.9	53.3 / 5.7	76.2 / 46.1	80.0 / 49.7	88.0 / 82.5
	nuScenes	47.4 / 13.3	55.0 / 10.4	60.8 / 23.9	54.7 / 21.7	73.4 / 38.1	49.7 / 13.4	75.4 / 31.5	83.2 / 35.6	83.8 / 58.7	88.0 / 82.5
	Lyft	81.7 / 51.8	88.2 / 23.5	88.3 / 73.3	89.0 / 78.1	90.2 / 87.3	74.3 / 39.4	71.9 / 4.7	83.5 / 72.1	85.3 / 72.5	88.0 / 82.5
	Waymo	45.2 / 11.9	86.1 / 16.2	84.6 / 53.3	87.4 / 70.9	90.1 / 85.3	51.9 / 13.1	64.0 / 3.9	82.1 / 48.7	81.0 / 67.0	88.0 / 82.5
Mod.	Argoverse	44.9 / 22.3	59.9 / 7.9	61.5 / 38.2	60.7 / 37.3	69.9 / 44.2	56.6 / 31.4	52.2 / 7.3	67.2 / 40.5	68.8 / 42.8	80.6 / 68.9
	nuScenes	26.2 / 8.3	30.8 / 6.8	32.9 / 16.4	28.7 / 12.5	40.7 / 21.2	39.8 / 10.7	58.5 / 27.3	67.4 / 31.0	67.2 / 45.5	80.6 / 68.9
	Lyft	61.8 / 33.7	70.1 / 17.8	73.7 / 53.1	74.2 / 53.4	83.7 / 65.5	61.1 / 34.3	60.8 / 5.6	73.6 / 57.9	73.9 / 56.2	80.6 / 68.9
	Waymo	43.9 / 12.3	69.1 / 13.1	74.9 / 49.4	75.9 / 55.3	85.9 / 67.9	45.8 / 13.2	54.9 / 3.7	71.3 / 47.1	66.8 / 51.8	80.6 / 68.9
Hard	Argoverse	42.5 / 22.2	59.3 / 9.3	60.6 / 37.1	59.8 / 36.5	69.9 / 42.8	58.5 / 33.3	53.5 / 8.6	68.5 / 41.9	66.3 / 43.0	81.9 / 66.7
	nuScenes	24.9 / 8.8	27.8 / 7.6	31.9 / 15.8	27.5 / 12.4	40.2 / 20.5	39.6 / 10.1	59.5 / 27.8	65.2 / 30.8	64.7 / 44.5	81.9 / 66.7
	Lyft	57.4 / 34.2	66.5 / 19.1	73.1 / 53.5	71.8 / 52.9	79.3 / 65.5	60.7 / 33.9	63.1 / 6.9	75.2 / 58.9	74.1 / 56.2	81.9 / 66.7
	Waymo	41.5 / 12.6	68.7 / 13.9	69.4 / 49.4	70.1 / 54.4	80.4 / 67.7	46.3 / 12.6	58.0 / 4.1	73.0 / 49.7	68.1 / 52.9	81.9 / 66.7
0-30	Argoverse	58.4 / 34.7	73.0 / 13.7	73.1 / 54.2	73.6 / 55.2	83.3 / 63.3	74.2 / 46.8	64.9 / 10.1	83.3 / 53.9	84.0 / 56.9	88.8 / 84.9
	nuScenes	47.9 / 14.9	56.2 / 13.9	60.0 / 29.2	54.0 / 23.6	73.2 / 42.8	50.7 / 13.9	74.6 / 36.6	83.6 / 42.8	81.2 / 59.8	88.8 / 84.9
	Lyft	77.8 / 54.2	88.4 / 27.5	88.8 / 75.4	89.3 / 77.6	90.4 / 88.5	75.1 / 45.2	74.8 / 9.1	87.4 / 73.6	87.5 / 73.9	88.8 / 84.9
	Waymo	48.0 / 14.0	87.7 / 22.2	87.1 / 60.1	88.7 / 74.1	90.4 / 87.2	56.8 / 15.0	71.3 / 4.4	85.7 / 59.0	84.8 / 71.0	88.8 / 84.9
30-50	Argoverse	46.5 / 19.0	56.1 / 5.4	61.5 / 31.5	59.0 / 29.9	72.2 / 39.5	33.9 / 11.8	35.1 / 9.1	48.9 / 25.7	47.9 / 23.8	70.2 / 51.4
	nuScenes	9.8 / 4.5	10.8 / 9.1	11.0 / 2.3	9.5 / 6.1	17.1 / 4.1	24.1 / 3.8	35.5 / 15.5	44.9 / 18.6	45.0 / 25.1	70.2 / 51.4
	Lyft	60.1 / 34.5	67.4 / 10.7	73.8 / 52.2	73.7 / 50.4	83.8 / 62.7	39.3 / 16.6	43.3 / 3.9	58.3 / 38.0	57.7 / 33.3	70.2 / 51.4
	Waymo	50.5 / 21.4	73.6 / 10.4	78.1 / 54.9	78.1 / 57.2	87.5 / 68.8	31.7 / 9.3	39.8 / 4.5	57.3 / 36.3	49.2 / 29.2	70.2 / 51.4
50-70	Argoverse	9.2 / 3.0	20.5 / 1.0	23.8 / 5.6	20.1 / 6.3	29.9 / 6.9	10.9 / 1.3	8.0 / 0.8	9.1 / 2.6	8.1 / 3.8	28.8 / 12.0
	nuScenes	1.1 / 0.0	1.5 / 1.0	3.0 / 2.3	3.3 / 1.2	9.1 / 9.1	6.5 / 1.5	7.8 / 5.1	9.4 / 5.1	12.9 / 5.7	28.8 / 12.0
	Lyft	33.2 / 9.6	41.3 / 6.8	49.9 / 22.2	46.8 / 19.4	62.7 / 33.1	13.6 / 4.6	12.7 / 0.9	21.1 / 6.7	17.5 / 8.0	28.8 / 12.0
	Waymo	27.1 / 12.0	42.6 / 4.2	46.8 / 25.1	45.2 / 24.3	63.5 / 41.1	5.6 / 1.8	7.7 / 1.1	14.4 / 5.7	10.5 / 4.8	28.8 / 12.0

statistics of the target domain are available. We argue that both scenarios are practical: we can simply annotate for every place a few labeled instances, or get the statistics from the local vehicle offices or car-selling websites. *In the main paper, we will mainly focus on training from KITTI and testing on the others, and vice versa.* We leave other results in the Supplementary Material.

Few-shot (FS) fine-tuning. In the first scenario where a few labeled scenes from the target domain are accessible, we investigate fine-tuning the already trained object detector with these few-shot examples. As shown in Table 4, using only 10 labeled scenes (average over five rounds of experiments) of the target domain, we can already improve the AP_{3D} by over 20.4% on average when adapting KITTI to other datasets and 24.4% on average when adapting other datasets to KITTI. Figure 6 further shows the performance by fine-tuning with different number of scenes. With merely 20 labeled target scenes, the adapted detector from Lyft and Waymo can already be on a par with that trained from scratch in the target domain with 500 scenes.

Statistical normalization (SN). For the second scenario where the target statistics (*i.e.*, average height, width, and length of cars) are accessible, we investigate modifying the already trained object detector so that its predicted box sizes can better match the target statistics. We propose a data modification scheme named *statistical normalization* by adjusting the source domain data, as illustrated in Figure 7. Specifically, we compute the difference of mean car sizes between the target domain (TD) and source domain (SD), $\Delta = (\Delta h, \Delta w, \Delta l) = (h_{\text{TD}}, w_{\text{TD}}, L_{\text{TD}}) - (h_{\text{SD}}, w_{\text{SD}}, L_{\text{SD}})$, where h, w, l stand for the height, width, and length, re-

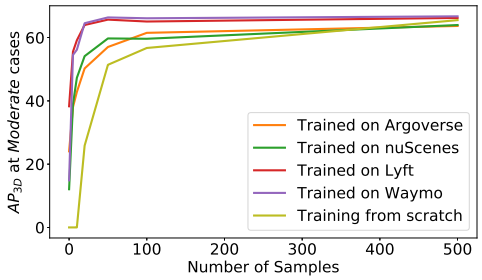

Figure 6: The few-shot fine-tuning performance on KITTI validation set with the model pre-trained on Argoverse, nuScenes, Lyft, and Waymo datasets. The x-axis indicates how many KITTI training images are used for fine-tuning. The y-axis marks AP_{3D} (moderate cases). *Scratch* denotes the model trained on the sampled KITTI training images with randomly initialized weights.

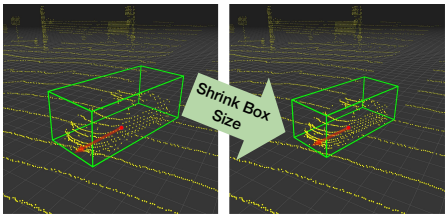

Figure 7: **Statistical Normalization (SN).** We shrink (or enlarge) the bounding box sizes (in the output space) and the corresponding point clouds (in the input space) in the training scenes of the source domain to match the *mean* statistics of the target domain. We fine-tune the detector with these modified source scenes.

spectively[4]. We then modify both the point clouds and the labels in the source domain with respect to Δ. For each annotated bounding box of cars, we adjust its size by adding $(\Delta h, \Delta w, \Delta l)$. We also crop the points inside the original box, scale up or shrink their coordinates to fit the adjusted bounding box size accordingly, and paste them back to the point cloud of the scene. By doing so, we generate new point clouds and labels whose car sizes are much similar to the target domain data. We then fine-tune the already trained model on the source domain with these data.

Surprisingly, with such a simple method that does not requires labeled target domain data, the performance is significantly improved (see Table 4) between KITTI and other datasets that obviously contain cars of different styles (*i.e.*, one in Germany, and others in the USA). Figure 4 and Figure 8 further analyze the prediction before and after statistical normalization. We see a clear shift of the histogram (predicted box) from the source to the target domain.

Output transformation (OT). We investigate an even simpler approach by directly adjusting the detector's prediction without fine-tuning — by adding $(\Delta h, \Delta w, \Delta l)$ to the predicted size. As shown in Table 4, this approach does not always improve but sometimes degrade the accuracy. This is because when we apply the source detector to the target domain, the predicted box sizes do slightly deviate from the source statistics to the target ones due to the difference of object sizes in the input signals (see Figure 4). Thus, simply adding $(\Delta h, \Delta w, \Delta l)$ may *over-correct* the bias. We hypothesize that by searching a suitable scale for addition or designing more intelligent output transformations can alleviate this problem and we leave them for future work.

Discussion. As shown in Table 4, statistical normalization largely improves over direct applying the source-domain detector. For some pairs of data sets (*e.g.*, from KITTI to Lyft, the AP_{BEV} after statistical normalization is encouraging, largely closing the gap to the **Within** performance.

Compared to domain adaptation on 2D images, there are more possible factors of domain gaps in 3D. While the box size difference is just one factor, we find addressing it to be highly effective in closing the gaps. This factor is rarely discussed in other domain adaptation tasks. We thus expect it and our solution to be valuable additions to the community.

[4]Here we obtain the target statistics directly from the dataset. We investigate using the car sales data online in the Supplementary Material.

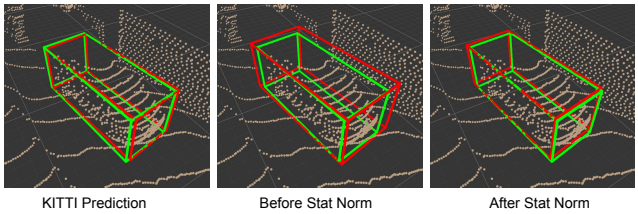

Figure 8: **Illustration of car prediction on KITTI w/o and w/ statistical normalization (Stat Norm).** The green boxes and red boxes indicate the ground truth and prediction, respectively. The box in the left image is predicted by POINTRCNN trained on KITTI. The middle image shows POINTRCNN that is pre-trained on Waymo and directly tested on KITTI. With statistical normalization, the model trained on Waymo only (with modified data) can accurately predict the bounding box shown in the right image.

6. Conclusion

In conclusion, in this paper we are the first (to our knowledge) to provide and investigate a standardized form of most widely-used 3D object detection datasets for autonomous driving. Although naïve adaptation across datasets is unsurprisingly difficult, we observe that, surprisingly, there appears to be a single dominant factor that explains a majority share of the adaptation gap: varying car sizes across different geographic regions. That car sizes play such an important role in adaptation ultimately makes sense. No matter if the detection is based on LiDAR or stereo cameras, cars are only observed from one side — and the depth of the bounding boxes must be estimated based on experience. If a deep network trained in Germany encounters an American Ford F-Series truck (with 5.3m length), it has little chance to correctly estimate the corresponding bounding box. It is surprising, however, that just matching the mean size of cars in the areas during fine-tuning already reduces this uncertainty so much. We hope that this publication will kindle interests in the exciting problem of cross-dataset domain adaptation for 3D object detection and localization, and that researchers will be careful to first apply simple global corrections before developing new computer vision algorithms to tackle the remaining adaptation gap.

Acknowledgments

This research is supported by grants from the National Science Foundation NSF (III-1618134, III-1526012, IIS-1149882, IIS-1724282, and TRIPODS-1740822), the Office of Naval Research DOD (N00014-17-1-2175), the Bill and Melinda Gates Foundation, and the Cornell Center for Materials Research with funding from the NSF MRSEC program (DMR-1719875). We are thankful for generous support by Zillow and SAP America Inc.

References

[1] https://en.wikipedia.org/wiki/Death_of_Elaine_Herzberg, 2018. 2

[2] https://www.motor1.com/features/280320/20-bestselling-vehicles-2018/, 2018. 2

[3] Waymo open dataset: An autonomous driving dataset, 2019. 1, 2, 3, 4

[4] Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. *arXiv preprint arXiv:1903.11027*, 2019. 1, 2, 3, 4

[5] Qi Cai, Yingwei Pan, Chong-Wah Ngo, Xinmei Tian, Lingyu Duan, and Ting Yao. Exploring object relation in mean teacher for cross-domain detection. In *CVPR*, 2019. 3

[6] Florian Chabot, Mohamed Chaouch, Jaonary Rabarisoa, Céline Teulière, and Thierry Chateau. Deep manta: A coarse-to-fine many-task network for joint 2d and 3d vehicle analysis from monocular image. In *CVPR*, 2017. 2

[7] Ming-Fang Chang, John W Lambert, Patsorn Sangkloy, Jagjeet Singh, Slawomir Bak, Andrew Hartnett, De Wang, Peter Carr, Simon Lucey, Deva Ramanan, and James Hays. Argoverse: 3d tracking and forecasting with rich maps. In *CVPR*, 2019. 1, 2, 3, 4

[8] Xiaozhi Chen, Kaustav Kundu, Ziyu Zhang, Huimin Ma, Sanja Fidler, and Raquel Urtasun. Monocular 3d object detection for autonomous driving. In *CVPR*, 2016. 2

[9] Xiaozhi Chen, Kaustav Kundu, Yukun Zhu, Andrew G Berneshawi, Huimin Ma, Sanja Fidler, and Raquel Urtasun. 3d object proposals for accurate object class detection. In *NIPS*, 2015. 2, 3

[10] Xiaozhi Chen, Kaustav Kundu, Yukun Zhu, Huimin Ma, Sanja Fidler, and Raquel Urtasun. 3d object proposals using stereo imagery for accurate object class detection. *TPAMI*, 40(5):1259–1272, 2018. 2

[11] Xiaozhi Chen, Huimin Ma, Ji Wan, Bo Li, and Tian Xia. Multi-view 3d object detection network for autonomous driving. In *CVPR*, 2017. 1, 2

[12] Yuhua Chen, Wen Li, Christos Sakaridis, Dengxin Dai, and Luc Van Gool. Domain adaptive faster r-cnn for object detection in the wild. In *CVPR*, 2018. 3

[13] Yuhua Chen, Wen Li, and Luc Van Gool. Road: Reality oriented adaptation for semantic segmentation of urban scenes. In *CVPR*, 2018. 3

[14] Yilun Chen, Shu Liu, Xiaoyong Shen, and Jiaya Jia. Fast point r-cnn. In *ICCV*, 2019. 1

[15] Yilun Chen, Shu Liu, Xiaoyong Shen, and Jiaya Jia. Dsgn: Deep stereo geometry network for 3d object detection. In *CVPR*, 2020. 1

[16] Xinxin Du, Marcelo H Ang Jr, Sertac Karaman, and Daniela Rus. A general pipeline for 3d detection of vehicles. In *ICRA*, 2018. 1

[17] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *JMLR*, 17(1):2096–2030, 2016. 3

[18] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013. 1, 2, 3, 4

[19] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *CVPR*, 2012. 1, 2, 3, 4

[20] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *ICCV*, 2017. 2

[21] Zhenwei He and Lei Zhang. Multi-adversarial faster-rcnn for unrestricted object detection. In *ICCV*, 2019. 3

[22] Judy Hoffman, Eric Tzeng, Taesung Park, Jun-Yan Zhu, Phillip Isola, Kate Saenko, Alexei A Efros, and Trevor Darrell. Cycada: Cycle-consistent adversarial domain adaptation. In *ICML*, 2018. 3

[23] Han-Kai Hsu, Chun-Han Yao, Yi-Hsuan Tsai, Wei-Chih Hung, Hung-Yu Tseng, Maneesh Singh, and Ming-Hsuan Yang. Progressive domain adaptation for object detection. In *WACV*, 2020. 3

[24] Haoshuo Huang, Qixing Huang, and Philipp Krahenbuhl. Domain transfer through deep activation matching. In *ECCV*, 2018. 3

[25] R. Kesten, M. Usman, J. Houston, T. Pandya, K. Nadhamuni, A. Ferreira, M. Yuan, B. Low, A. Jain, P. Ondruska, S. Omari, S. Shah, A. Kulkarni, A. Kazakova, C. Tao, L. Platinsky, W. Jiang, and V. Shet. Lyft level 5 av dataset 2019. urlhttps://level5.lyft.com/dataset/, 2019. 1, 2, 3, 4

[26] Mehran Khodabandeh, Arash Vahdat, Mani Ranjbar, and William G Macready. A robust learning approach to domain adaptive object detection. In *ICCV*, 2019. 3

[27] Taekyung Kim, Minki Jeong, Seunghyeon Kim, Seokeon Choi, and Changick Kim. Diversify and match: A domain adaptive representation learning paradigm for object detection. In *CVPR*, 2019. 3

[28] Hendrik Königshof, Niels Ole Salscheider, and Christoph Stiller. Realtime 3d object detection for automated driving using stereo vision and semantic information. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, 2019. 1

[29] Jason Ku, Melissa Mozifian, Jungwook Lee, Ali Harakeh, and Steven Waslander. Joint 3d proposal generation and object detection from view aggregation. In *IROS*, 2018. 1

[30] Alex H Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *CVPR*, 2019. 1, 2

[31] Buyu Li, Wanli Ouyang, Lu Sheng, Xingyu Zeng, and Xiaogang Wang. Gs3d: An efficient 3d object detection framework for autonomous driving. In *CVPR*, 2019. 1

[32] Peiliang Li, Xiaozhi Chen, and Shaojie Shen. Stereo r-cnn based 3d object detection for autonomous driving. In *CVPR*, 2019. 1

[33] Ming Liang, Bin Yang, Yun Chen, Rui Hu, and Raquel Urtasun. Multi-task multi-sensor fusion for 3d object detection. In *CVPR*, 2019. 1, 2

[34] Ming Liang, Bin Yang, Shenlong Wang, and Raquel Urtasun. Deep continuous fusion for multi-sensor 3d object detection. In *ECCV*, 2018. 1, 2

[35] Tsung-Yi Lin, Piotr Dollár, Ross B Girshick, Kaiming He, Bharath Hariharan, and Serge J Belongie. Feature pyramid networks for object detection. In *CVPR*, 2017. 2

[36] Yawei Luo, Liang Zheng, Tao Guan, Junqing Yu, and Yi Yang. Taking a closer look at domain shift: Category-level adversaries for semantics consistent domain adaptation. In *CVPR*, 2019. 3

[37] Gregory P Meyer, Ankit Laddha, Eric Kee, Carlos Vallespi-Gonzalez, and Carl K Wellington. Lasernet: An efficient probabilistic 3d object detector for autonomous driving. In *CVPR*, 2019. 1, 2

[38] Arsalan Mousavian, Dragomir Anguelov, John Flynn, and Jana Košecká. 3d bounding box estimation using deep learning and geometry. In *CVPR*, 2017. 2

[39] Cuong Cao Pham and Jae Wook Jeon. Robust object proposals re-ranking for object detection in autonomous driving using convolutional neural networks. *Signal Processing: Image Communication*, 53:110–122, 2017. 2

[40] Alex D Pon, Jason Ku, Chengyao Li, and Steven L Waslander. Object-centric stereo matching for 3d object detection. In *ICRA*, 2020. 1

[41] Charles R Qi, Wei Liu, Chenxia Wu, Hao Su, and Leonidas J Guibas. Frustum pointnets for 3d object detection from rgb-d data. In *CVPR*, 2018. 1, 2

[42] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *CVPR*, 2017. 2

[43] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *NIPS*, 2017. 2, 3

[44] Rui Qian, Divyansh Garg, Yan Wang, Yurong You, Serge Belongie, Bharath Hariharan, Mark Campbell, Kilian Q Weinberger, and Wei-Lun Chao. End–end pseudo-lidar for image-based 3d object detection. In *CVPR*, 2020. 2

[45] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NIPS*, 2015. 2

[46] Adrian Lopez Rodriguez and Krystian Mikolajczyk. Domain adaptation for object detection via style consistency. In *BMVC*, 2019. 3

[47] Kuniaki Saito, Yoshitaka Ushiku, Tatsuya Harada, and Kate Saenko. Strong-weak distribution alignment for adaptive object detection. In *CVPR*, 2019. 3

[48] Kuniaki Saito, Kohei Watanabe, Yoshitaka Ushiku, and Tatsuya Harada. Maximum classifier discrepancy for unsupervised domain adaptation. In *CVPR*, 2018. 3

[49] Fatemeh Sadat Saleh, Mohammad Sadegh Aliakbarian, Mathieu Salzmann, Lars Petersson, and Jose M Alvarez. Effective use of synthetic data for urban scene semantic segmentation. In *ECCV*. Springer, 2018. 3

[50] Swami Sankaranarayanan, Yogesh Balaji, Arpit Jain, Ser Nam Lim, and Rama Chellappa. Learning from synthetic data: Addressing domain shift for semantic segmentation. In *CVPR*, 2018. 3

[51] Shaoshuai Shi, Chaoxu Guo, Li Jiang, Zhe Wang, Jianping Shi, Xiaogang Wang, and Hongsheng Li. Pv-rcnn: Point-voxel feature set abstraction for 3d object detection. In *CVPR*, 2020. 1

[52] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. Pointrcnn: 3d object proposal generation and detection from point cloud. In *CVPR*, 2019. 1, 2, 3, 5, 6, 7

[53] Shaoshuai Shi, Zhe Wang, Jianping Shi, Xiaogang Wang, and Hongsheng Li. From points to parts: 3d object detection from point cloud with part-aware and part-aggregation network. *TPAMI*, 2020. 1

[54] Weijing Shi and Ragunathan Rajkumar. Point-gnn: Graph neural network for 3d object detection in a point cloud. In *CVPR*, 2020. 1

[55] Yi-Hsuan Tsai, Wei-Chih Hung, Samuel Schulter, Kihyuk Sohn, Ming-Hsuan Yang, and Manmohan Chandraker. Learning to adapt structured output space for semantic segmentation. In *CVPR*, 2018. 3

[56] Shenlong Wang, Simon Suo, Wei-Chiu Ma3 Andrei Pokrovsky, and Raquel Urtasun. Deep parametric continuous convolutional neural networks. In *CVPR*, 2018. 2

[57] Tao Wang, Xiaopeng Zhang, Li Yuan, and Jiashi Feng. Few-shot adaptive faster r-cnn. In *CVPR*, 2019. 3

[58] Yan Wang, Wei-Lun Chao, Divyansh Garg, Bharath Hariharan, Mark Campbell, and Kilian Q. Weinberger. Pseudo-lidar from visual depth estimation: Bridging the gap in 3d object detection for autonomous driving. In *CVPR*, 2019. 2

[59] Yu Xiang, Wongun Choi, Yuanqing Lin, and Silvio Savarese. Subcategory-aware convolutional neural networks for object proposals and detection. In *WACV*, 2017. 2

[60] Bin Xu and Zhenzhong Chen. Multi-level fusion based 3d object detection from monocular images. In *CVPR*, 2018. 2

[61] Zhenbo Xu, Wei Zhang, Xiaoqing Ye, Xiao Tan, Wei Yang, Shilei Wen, Errui Ding, Ajin Meng, and Liusheng Huang. Zoomnet: Part-aware adaptive zooming neural network for 3d object detection. 2020. 1

[62] Yan Yan, Yuxing Mao, and Bo Li. Second: Sparsely embedded convolutional detection. *Sensors*, 18(10):3337, 2018. 1

[63] Bin Yang, Wenjie Luo, and Raquel Urtasun. Pixor: Real-time 3d object detection from point clouds. In *CVPR*, 2018. 1, 2, 3, 4

[64] Zetong Yang, Yanan Sun, Shu Liu, and Jiaya Jia. 3dssd: Point-based 3d single stage object detector. In *CVPR*, 2020. 1

[65] Zetong Yang, Yanan Sun, Shu Liu, Xiaoyong Shen, and Jiaya Jia. Std: Sparse-to-dense 3d object detector for point cloud. In *ICCV*, 2019. 1

[66] Yurong You, Yan Wang, Wei-Lun Chao, Divyansh Garg, Geoff Pleiss, Bharath Hariharan, Mark Campbell, and Kilian Q Weinberger. Pseudo-lidar++: Accurate depth for 3d object detection in autonomous driving. In *ICLR*, 2020. 2

[67] Yang Zhang, Philip David, and Boqing Gong. Curriculum domain adaptation for semantic segmentation of urban scenes. In *ICCV*, 2017. 3

[68] Yin Zhou, Pei Sun, Yu Zhang, Dragomir Anguelov, Jiyang Gao, Tom Ouyang, James Guo, Jiquan Ngiam, and Vijay Vasudevan. End-to-end multi-view fusion for 3d object detection in lidar point clouds. In *CoRL*, 2019. 1

[69] Yin Zhou and Oncel Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. In *CVPR*, 2018. 1, 2

[70] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *ICCV*, 2017. 3

[71] Xinge Zhu, Jiangmiao Pang, Ceyuan Yang, Jianping Shi, and Dahua Lin. Adapting object detectors via selective cross-domain alignment. In *CVPR*, 2019. 3

[72] Chenfan Zhuang, Xintong Han, Weilin Huang, and Matthew R Scott. ifan: Image-instance full alignment networks for adaptive object detection. In *AAAI*, 2020. 3

[73] Yang Zou, Zhiding Yu, BVK Vijaya Kumar, and Jinsong Wang. Unsupervised domain adaptation for semantic segmentation via class-balanced self-training. In *ECCV*, 2018. 3