SqueezeSegV2: Improved Model Structure and Unsupervised Domain Adaptation for Road-Object Segmentation from a LiDAR Point Cloud

Bichen Wu*, Xuanyu Zhou*, Sicheng Zhao*, Xiangyu Yue, Kurt Keutzer
UC Berkeley
{bichen, xuanyu_zhou, schzhao, xyyue, keutzer}@berkeley.edu

Abstract— **Earlier work demonstrates the promise of deep-learning-based approaches for point cloud segmentation; however, these approaches need to be improved to be practically useful. To this end, we introduce a new model SqueezeSegV2 that is more robust to dropout noise in LiDAR point clouds. With improved model structure, training loss, batch normalization and additional input channel, SqueezeSegV2 achieves significant accuracy improvement when trained on real data. Training models for point cloud segmentation requires large amounts of labeled point-cloud data, which is expensive to obtain. To sidestep the cost of collection and annotation, simulators such as GTA-V can be used to create unlimited amounts of labeled, synthetic data. However, due to domain shift, models trained on synthetic data often do not generalize well to the real world. We address this problem with a domain-adaptation training pipeline consisting of three major components: 1) learned intensity rendering, 2) geodesic correlation alignment, and 3) progressive domain calibration. When trained on real data, our new model exhibits segmentation accuracy improvements of 6.0-8.6% over the original SqueezeSeg. When training our new model on synthetic data using the proposed domain adaptation pipeline, we nearly double test accuracy on real-world data, from 29.0% to 57.4%. Our source code and synthetic dataset will be open-sourced.**

I. INTRODUCTION

Accurate, real-time, and robust perception of the environment is an indispensable component in autonomous driving systems. For perception in high-end autonomous vehicles, LiDAR (Light Detection And Ranging) sensors play an important role. LiDAR sensors can directly provide distance measurements, and their resolution and field of view exceed those of radar and ultrasonic sensors [1]. LiDAR sensors are robust under almost all lighting conditions: day or night, with or without glare and shadows [2]. As such, LiDAR-based perception has attracted significant research attention.

Recently, deep learning has been shown to be very effective for LiDAR perception tasks. Specifically, Wu et al. proposed SqueezeSeg [2], which focuses on the problem of point-cloud segmentation. SqueezeSeg projects a 3D LiDAR point cloud onto a spherical surface, and uses a 2D CNN to predict point-wise labels for the point cloud. SqueezeSeg is extremely efficient – the fastest version achieves an inference speed of over 100 frames per second. However, SqueezeSeg still has several limitations: first, its accuracy still needs to be improved to be practically useful. One important reason for accuracy degradation is *dropout noise* – missing points from the sensed point cloud caused by limited sensing range, mirror diffusion of the sensing laser, or jitter in incident angles. Such dropout noise can corrupt the output of SqueezeSeg's early layers, which reduces accuracy. Second, training deep learning models such as SqueezeSeg requires tens of thousands of labeled point clouds; however, collecting and annotating this data is even more time consuming and expensive than collecting comparable data from cameras. GTA-V is used to synthesize LiDAR point cloud as an extra source of training data [2]; however, this approach suffers from the domain shift problem [3] - models trained on synthetic data usually fail catastrophically on the real data, as shown in Fig. 1. Domain shift comes from different sources, but the absence of dropout noise and intensity signals in GTA-V are two important factors. Simulating realistic dropout noise and intensity is very difficult, as it requires sophisticated modeling of both the LiDAR device and the environment, both of which contain a lot of non-deterministic factors. As such, the LiDAR point clouds generated by GTA-V do not contain dropout noise and intensity signals. The comparison of simulated data and real data is shown in Fig. 1 (a), (b).

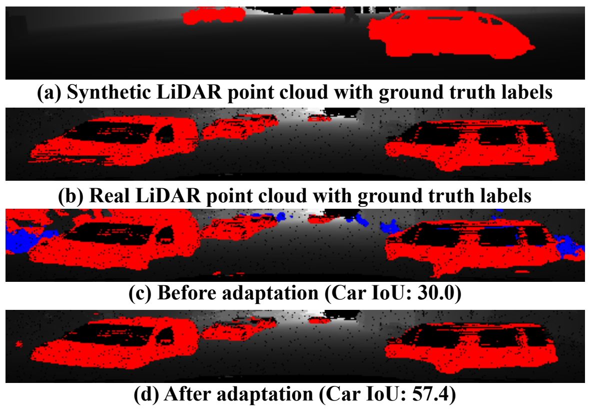

(a) Synthetic LiDAR point cloud with ground truth labels
(b) Real LiDAR point cloud with ground truth labels
(c) Before adaptation (Car IoU: 30.0)
(d) After adaptation (Car IoU: 57.4)

Fig. 1. An example of *domain shift*. The point clouds are projected onto a spherical surface for visualization (car in red, pedestrian in blue). Our domain adaptation pipeline improves the segmentation from (c) to (d) while trained on synthetic data.

In this paper, we focus on addressing the challenges above. First, to improve the accuracy, we mitigate the impact of dropout noise by proposing the Context Aggregation Module (CAM), a novel CNN module that aggregates contextual information from a larger receptive field and improves the robustness of the network to dropout noise. Adding CAM to the early layers of SqueezeSegV2 not only significantly improves

* Authors contributed equally.

its performance when trained on real data, but also effectively reduces the domain gap, boosting the network's real-world test accuracy when trained on synthetic data. In addition to CAM, we adopt several improvements to SqueezeSeg, including using focal loss [4], batch normalization [5], and LiDAR mask as an input channel. These improvements together boosted the accuracy of SqueezeSegV2 by 6.0% - 8.6% in all categories on the converted KITTI dataset [2].

Second, to better utilize synthetic data for training the model, we propose a domain adaptation training pipeline that contains the following steps: first, before training, we render intensity channels in synthetic data through *learned intensity rendering*. We train a neural network that takes the point coordinates as input, and predicts intensity values. This rendering network can be trained in a "self-supervised" fashion on unlabeled real data. After training the network, we feed the synthetic data into the network and render the intensity channel, which is absent from the original simulation. Second, we use the synthetic data augmented with rendered intensity to train the network. Meanwhile, we follow [6] and use *geodesic correlation alignment* to align the batch statistics between real data and synthetic data. 3) After training, we propose *progressive domain calibration* to further reduce the gap between the target domain and the trained network. Experiments show that the above domain-adaptation training pipeline significantly improves the accuracy of the model trained with synthetic data from 29.0% to 57.4% on the real world test data.

The contributions of this paper are threefold: 1) We improve the model structure of SqueezeSeg with CAM to increase its robustness to dropout noise, which leads to significant accuracy improvements of 6.0% to 8.6% for different categories. We name the new model SqueezeSegV2. 2) We propose a domain-adaptation training pipeline that significantly reduces the distribution gap between synthetic data and real data. Model trained on synthetic data achieves 28.4% accuracy improvement on the real test data. 3) We create a large-scale 3D LiDAR point cloud dataset, GTA-LiDAR, which consists of 100,000 samples of synthetic point cloud augmented with rendered intensity. The source code and dataset will be open-sourced.

II. RELATED WORK

3D LiDAR Point Cloud Segmentation aims to recognize objects from point clouds by predicting point-wise labels. Non-deep-learning methods [1], [7], [8] usually involve several stages such as ground removal, clustering, and classification. SqueezeSeg [2] is one early work that applies deep learning to this problem. Piewak et al. [9] adopted a similar problem formulation and pipeline to SqueezeSeg and proposed a new network architecture called LiLaNet. They created a dataset by utilizing image-based semantic segmentation to generate labels for the LiDAR point cloud. However, the dataset was not released, so we were not able to conduct a direct comparison to their work. Another category of methods is based on PointNet [10], [11], which treats a point cloud as an unordered set of 3D points. This is effective with 3D perception problems such as classification and segmentation. Limited by its computational complexity; however, PointNet is mainly used to process indoor scenes where the number of points is limited. Frustum-PointNet [12] is proposed for out-door object detection, but it relies on image object detection to first locate object clusters and feeds the cluster, instead of the whole point cloud, to the PointNet.

Unsupervised Domain Adaptation (UDA) aims to adapt the models from one labeled source domain to another unlabeled target domain. Recent UDA methods have focused on transferring deep neural network representations [13], [14]. Typically, deep UDA methods employ a conjoined architecture with two streams to represent the models for the source and target domains, respectively. In addition to the task related loss computed from the labeled source data, deep UDA models are usually trained jointly with another loss, such as a discrepancy loss [15], [16], [17], [18], [6], adversarial loss [19], [20], [21], [22], [23], [24], label distribution loss [18] or reconstruction loss [25], [26].

The most relevant work is the exploration of synthetic data [22], [18], [24]. By enforcing a self-regularization loss, Shrivastava et al. [22] proposed SimGAN to improve the realism of synthetic data using unlabeled real data. Another category of relevant work employs a discrepancy loss [15], [16], [17], [6], which explicitly measures the discrepancy between the source and target domains on corresponding activation layers of the two network streams. Instead of working on 2D images, we try to adapt synthetic 3D LiDAR point clouds by a novel adaptation pipeline.

Simulation has recently been used for creating large-scale ground truth data for training purposes. Richter et al. [27] provided a method to extract semantic segmentation for the synthesized in-game images. In [28], the same game engine is used to extract ground truth 2D bounding boxes for objects in the image. Yue et al. [29] proposed a framework to generate synthetic LiDAR point clouds. Richter et al. [30] and Krähenbühl [31] extracted more types of information from video games.

III. IMPROVING THE MODEL STRUCTURE

We propose SqueezeSegV2, by improving upon the base SqueezeSeg model, adding Context Aggregation Module (CAM), adding LiDAR mask as an input channel, using batch normalization [5], and employing the focal loss [4]. The network structure of SqueezeSegV2 is shown in Fig. 2.

A. Context Aggregation Module

LiDAR point cloud data contains many missing points, which we refer to as dropout noise, as shown in Fig. 1(b). Dropout noise is mainly caused by 1) limited sensor range, 2) mirror reflection (instead of diffusion reflection) of sensing lasers on smooth surfaces, and 3) jitter of the incident angle. Dropout noise has a significant impact on SqueezeSeg, especially in early layers of a network. At early layers where the receptive field of the convolution filter is very small, missing points in a small neighborhood can corrupt the output of the filter significantly. To illustrate this, we conduct

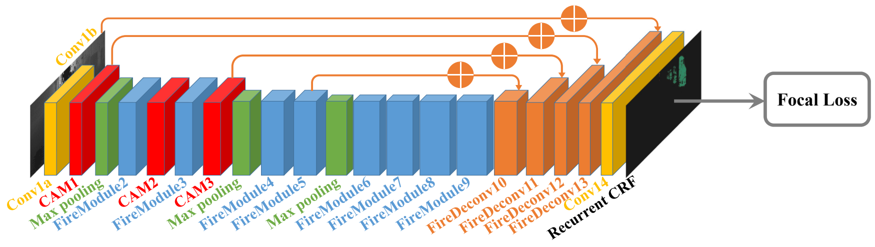


Fig. 2. Network structure of the proposed SqueezeSegV2 model for road-object segmentation from 3D LiDAR point clouds.

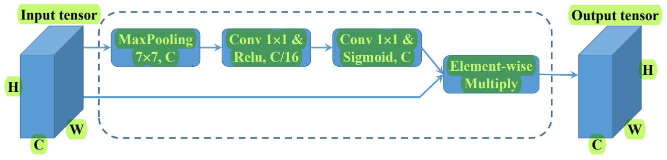


Fig. 3. Structure of Context Aggregation Module.

a simple numerical experiment, where we randomly sample an input tensor and feed it into a 3×3 convolution filter. We randomly drop out some pixels from the input tensor, and as shown in Fig. 4, as we increase the dropout probability, the difference between the errors of the corrupted output and the original output also increases.

This problem not only impacts SqueezeSeg when trained on real data, but also leads to a serious domain gap between synthetic data and real data, since simulating realistic dropout noise from the same distribution is very difficult.

To solve this problem, we propose a novel Context Aggregation Module (CAM) to reduce the sensitivity to dropout noise. As shown in Fig. 3, CAM starts with a max pooling with a relatively large kernel size. The max pooling aggregates contextual information around a pixel with a much larger receptive field, and it is less sensitive to missing data within its receptive field. Also, max pooling can be computed efficiently even with a large kernel size. The max pooling layer is then followed by two cascaded convolution layers with a ReLU activation in between. Following [32], we use the *sigmoid* function to normalize the output of the module and use an element-wise multiplication to combine the output with the input. As shown in Fig. 4, the proposed module is much less sensitive to dropout noise – with the same corrupted input data, the error is significantly reduced.

In SqueezeSegV2, we insert CAM after the output of the first three modules (1 convolution layer and 2 FireModules), where the receptive fields of the filters are small. As can be seen in later experiments, CAM 1) significantly improves the accuracy when trained on real data, and 2) significantly reduces the domain gap while trained on synthetic data and testing on real data.

B. Focal Loss

LiDAR point clouds have a very imbalanced distribution of point categories – there are many more background points than there are foreground objects such as cars, pedestrians, etc. This imbalanced distribution makes the model focus more on easy-to-classify background points which contribute

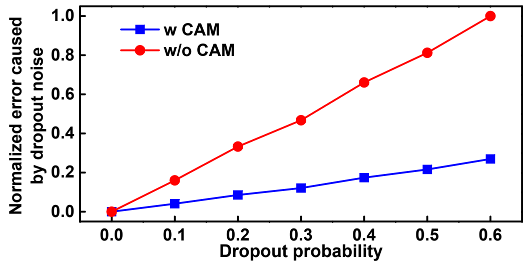


Fig. 4. We feed a random tensor to a convolutional filter, one with CAM before a 3×3 convolution filter and the other one without CAM. We randomly add dropout noise to the input, and measure the output errors. As we increase the dropout probability, the error also increases. For all dropout probabilities, adding CAM improve the robustness towards the dropout noise and therefore, the error is always smaller.

no useful learning signals, with the foreground objects not being adequately addressed during training.

To address this problem, we replace the original cross entropy loss from SqueezeSeg [2] with a focal loss [4]. The focal loss modulates the loss contribution from different pixels and focuses on hard examples. For a given pixel label t, and the predicted probability of p_t, focal loss [4] adds a modulating factor $(1 - p_t)^\gamma$ to the cross entropy loss. The focal loss for that pixel is thus

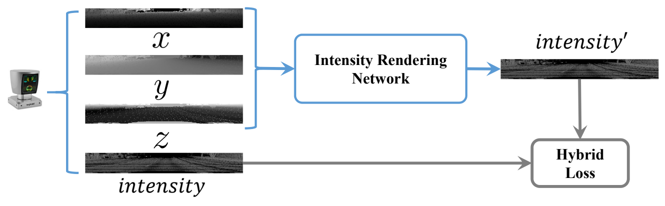
$$FL(p_t) = -(1 - p_t)^\gamma \log(p_t) \tag{1}$$

When a pixel is mis-classified and p_t is small, the modulating factor is near 1 and the loss is unaffected. As $p_t \to 1$, the factor goes to 0, and the loss for well-classified pixels is down-weighted. The focusing parameter γ smoothly adjusts the rate at which well-classified examples are down-weighted. When $\gamma = 0$, the Focal Loss is equivalent to the Cross Entropy Loss. As γ increases, the effect of the modulating factor is likewise increased. We choose γ to be 2 in our experiments.

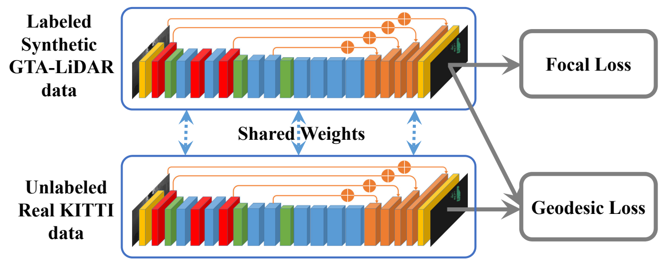
C. Other Improvements

LiDAR Mask: Besides the original (x, y, z, intensity, depth) channels, we add one more channel – a binary mask indicating if each pixel is missing or existing. As we can see from Table I, the addition of the mask channel significantly improves segmentation accuracy for cyclists.

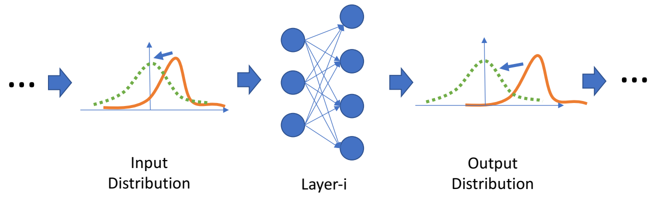
Batch Normalization: Unlike SqueezeSeg [2], we also add batch normalization (BN) [5] after every convolution layer. The BN layer is designed to alleviate the issue of internal covariate shift – a common problem for training



(a) Pre-training: Learned Intensity Rendering



(b) Training: Geodesic Correlation Alignment



(c) Post-training: Progressive Domain Calibration

Fig. 5. Framework of the proposed unsupervised domain adaptation method for road-object segmentation from the synthetic GTA-LiDAR dataset to the real-world KITTI dataset.

a deep neural network. We observe an improvement in car segmentation after using BN layers in Table I.

IV. Domain Adaptation Training

In this section, we introduce our unsupervised domain adaptation pipeline that trains SqueezeSegV2 on synthetic data and improves its performance on real data. We construct a large-scale 3D LiDAR point cloud dataset, GTA-LiDAR, with 100,000 LiDAR scans simulated on GTA-V. To deal with the *domain shift* problem, we employ three strategies: learned intensity rendering, geodesic correlation alignment, and progressive domain calibration, as shown in Fig. 5.

A. The GTA-LiDAR Dataset

We synthesize 100,000 LiDAR point clouds in GTA-V to train SqueezeSegV2. We use the framework in [31] to generate depth semantic segmentation maps, and use the method in [29] to do Image-LiDAR registration in GTA-V. Following [29], we collect 100,000 point cloud scans by deploying a virtual car to drive autonomously in the virtual world. GTA-V provides a wide variety of scenes, car types, traffic conditions, etc., which ensures the diversity of our synthetic data. Each point in the synthetic point cloud contains one label, one distance and x, y, z coordinates. However, it does not contain intensity, which represents the magnitude of the reflected laser signal. Also, the synthetic data does not contain dropout noise as in the real data. Because of such distribution discrepancies, the model trained on synthetic data fails to transfer to real data.

B. Learned Intensity Rendering

The synthetic data only contains $x, y, z, depth$ channels and does not have intensity. As shown in SqueezeSeg [2], intensity is an important signal. The absence of intensity can lead to serious accuracy loss. Rendering realistic intensity is a non-trivial task, since a multitude of factors that affect intensity, such as surface materials and LiDAR sensitivity, are generally unknown to us.

To solve this problem, we propose a method called *learned intensity rendering*. The idea is to use a network to take the $x, y, z, depth$ channels of the point cloud as input, and predict the intensity. Such rendering network can be trained with unlabeled LiDAR data, which can be easily collected as long as a LiDAR sensor is available. As shown in Fig. 5(a), we train the rendering network in a self-supervision fashion, splitting the x, y, z channels as input to the network and the intensity channel as the label. The structure of the rendering network is almost the same as SqueezeSeg, except that the CRF layer is removed.

The intensity rendering can be seen as a regression problem, where the ℓ_2 loss is a natural choice. However, ℓ_2 fails to capture the multi-modal distribution of the intensity – given the same input of x, y, z, the intensity can differ. To model this property, we designed a hybrid loss function that involves both classification and regression. We divide the intensity into $n = 10$ regions, with each region having a reference intensity value. The network first predicts which region the intensity belongs to. Once the region is selected, the network further predicts a deviation from the reference intensity. This way, the categorical prediction can capture the multi-modal distribution of the intensity, and the deviation prediction leads to more accurate estimations. We train the rendering network on the KITTI [33] dataset with the hybrid loss function and measure its accuracy with mean squared error (MSE). Compared to ℓ_2 loss, the converged MSE drops significantly by 3X from 0.033 to 0.011. A few rendered results using two different losses are shown in Fig. 6. After training the rendering network, we feed synthetic GTA-LiDAR data into the network to render point-wise intensities.

C. Geodesic Correlation Alignment

After rendering intensity, we train SqueezeSegV2 on the synthetic data with focal loss. However, due to distribution discrepancies between synthetic data and real data, the trained model usually fails to generalize to real data.

To reduce this domain discrepancy, we adopt geodesic correlation alignment during training. As shown in Fig. 5(b), at every step of training, we feed in one batch of synthetic data and one batch of real data to the network. We compute the focal loss on the synthetic batch, where labels are available. Meanwhile, we compute the geodesic distance [6] between the output distributions of two batches. The total loss now contains both the focal loss and the geodesic loss. Where the focal loss focuses on training the network to learn semantics from the point cloud, the geodesic loss penalizes discrepancies between batch statistics from two domains. Note that other distances such as the Euclidean distance

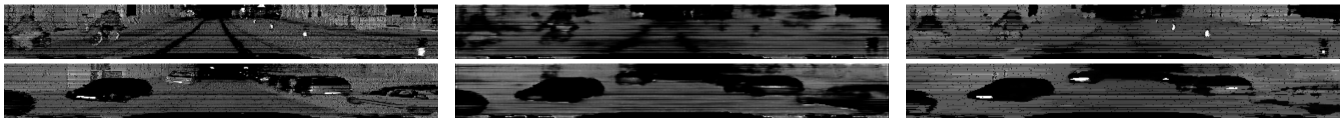

(a) Ground truth intensity **(b) Rendered intensity with ℓ_2 loss** **(c) Rendered intensity with hybrid loss**

Fig. 6. Rendered v.s. ground truth intensity in the KITTI dataset.

Algorithm 1: Progressive Domain Calibration

```
Input: Unlabeled real data $\mathcal{X}$, model $\mathcal{M}$
1 $\mathcal{X}^{(0)} \leftarrow \mathcal{X}$
2 for layer $l$ in the model $\mathcal{M}$ do
3     $\mathcal{X}^{(l)} \leftarrow \mathcal{M}^{(l)}(\mathcal{X}^{(l-1)})$
4     $\mu^{(l)} \leftarrow \mathbb{E}(\mathcal{X}^{(l)})$, $\sigma^{(l)} \leftarrow \sqrt{Var(\mathcal{X}^{(l)})}$
5     Update the BatchNorm parameters of $\mathcal{M}^{(l)}$
6     $\mathcal{X}^{(l)} \leftarrow (\mathcal{X}^{(l)} - \mu^{(l)})/\sigma^{(l)}$
7 end
Output: Calibrated model $\mathcal{M}$
```

can also be used to align the domain statistics. However, we choose the geodesic distance over the Euclidean distance since it takes into account the manifolds curvature. More details can be found in [6].

We denote the input synthetic data as X_{sim}, synthetic labels as Y_{sim} the input real data as X_{real}. Our loss function can be computed as

$$FL(X_{sim}, Y_{sim}) + \lambda \cdot GL(X_{sim}, X_{real}), \tag{2}$$

where FL denotes focal loss between the synthetic label and network prediction, GL denotes the geodesic loss between batch statistics of synthetic and real data. λ is a weight coefficient and we set it to 10 in our experiment. Note that in this step, we only require unlabeled real data, which is much easier to obtain than annotated data as long as a LiDAR sensor is available.

D. Progressive Domain Calibration

After training SqueezeSegV2 on synthetic data with geodesic correlation alignment, each layer of the network learns to recognize patterns from its input and extract higher level features. However, due to the non-linear nature of the network, each layer can only work well if its input is constrained within a certain range. Taking the ReLU function as an example, if somehow its input distribution shifts below 0, the output of the ReLU becomes all zero. Otherwise, if the input shifts towards larger than 0, the ReLU becomes a linear function. For deep learning models with multiple layers, distribution discrepancies from the input data can lead to distribution shift at the output of each layer, which is accumulated or even amplified across the network and eventually leads to a serious degradation of performance, as illustrated in Fig. 5(c).

To address this problem, we employ a post training procedure called progressive domain calibration (PDC). The idea is to break the propagation of the distribution shift through each layer with progressive layer-wise calibration. For a network trained on synthetic data, we feed the real data into the network. Staring from the first layer, we compute its output statistics (mean and variance) under the given input, and then re-normalize the output's mean to be 0 and its standard deviation to be 1, as shown in Fig. 5(c). Meanwhile, we update the batch normalization parameters (mean and variance) of the layer with the new statistics. We progressively repeat this process for all layers of the network until the last layer. Similar to geodesic correlation alignment, this process only requires unlabeled real data, which is presumably abundant. This algorithm is summarized in Algorithm 1. A similar idea was proposed in [34], but PDC is different since it performs calibration progressively, making sure that the calibrations of earlier layers do not impact those of later layers.

V. Experiments

In this section, we introduce the details of our experiments. We train and test SqueezeSegV2 on a converted KITTI [33] dataset as [2]. To verify the generalization ability, we further train SqueezeSegV2 on the synthetic GTA-LiDAR dataset and test it on the real world KITTI dataset.

A. Experimental Settings

We compare the proposed method with SqueezeSeg [2], one state-of-the-art model for semantic segmentation from 3D LiDAR point clouds. We use KITTI [33] as the real world dataset. KITTI provides images, LiDAR scans, and 3D bounding boxes organized in sequences. Following [2], we obtain the point-wise labels from 3D bounding boxes, all points of which are considered part of the target object. In total, 10,848 samples with point-wise labels are collected. For SqueezeSegV2, the dataset is split into a training set with 8,057 samples and a testing set with 2,791 samples. For domain adaptation, we train the model on GTA-LiDAR, and test it on KITTI for comparison.

Similar to [2], we evaluate our model's performance on class-level segmentation tasks by a point-wise comparison of the predicted results with ground-truth labels. We employ intersection-over-union (IoU) as our evaluation metric, which is defined as $IoU_c = \frac{|\mathcal{P}_c \cap \mathcal{G}_c|}{|\mathcal{P}_c \cup \mathcal{G}_c|}$, where $\mathcal{P}_c$ and $\mathcal{G}_c$ respectively denote the predicted and ground-truth point sets that belong to class-c. $|\cdot|$ denotes the cardinality of a set.

B. Improved Model Structure

The performance comparisons, measured in IoU, between the proposed SqueezeSegV2 model and baselines are shown in Table I. Some segmentation results are shown in Fig. 7.

From the results, we have the following observations. (1) both batch normalization and the mask channel can produce better segmentation results - batch normalization

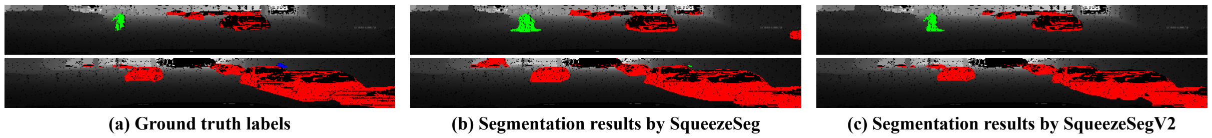


Fig. 7. Segmentation result comparison between SqueezeSeg [2] and our SqueezeSegV2 (red: car, green: cyclist). Note that in first row, SqueezeSegV2 produces much more accurate segmentation for the cyclist. In the second row, SqueezeSegV2 avoids a falsely detected car that is far away.

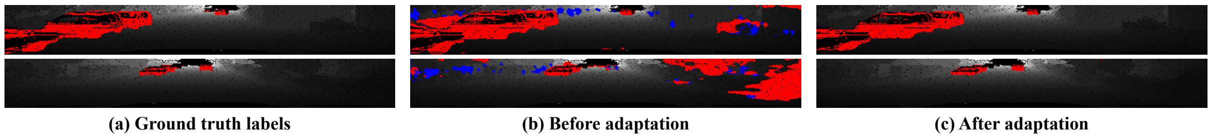


Fig. 8. Segmentation result comparison before and after domain adaptation (red: car, blue: pedestrian).

TABLE I

SEGMENTATION PERFORMANCE (IOU, %) COMPARISON BETWEEN THE PROPOSED SQUEEZESEGV2 (+BN+M+FL+CAM) MODEL AND STATE-OF-THE-ART BASELINES ON THE KITTI DATASET.

	Car	Pedestrian	Cyclist	Average
SqueezeSeg [2]	64.6	21.8	25.1	37.2
+BN	71.6	15.2	25.4	37.4
+BN+M	70.0	17.1	32.3	39.8
+BN+M+FL	71.2	22.8	27.5	40.5
+BN+M+FL+CAM	**73.2**	**27.8**	**33.6**	**44.9**
PointSeg [35]	67.4	19.2	32.7	39.8

+BN denotes using batch normalization. **+M** denotes adding LiDAR mask as input. **+FL** denotes using focal loss. **+CAM** denotes using the CAM module.

TABLE II

SEGMENTATION PERFORMANCE (IOU, %) OF THE PROPOSED DOMAIN ADAPTATION PIPELINE FROM GTA-LIDAR TO THE KITTI.

	Car	Pedestrian
SQSG trained on GTA [2]	29.0	-
SQSG trained on GTA-LiDAR	30.0	2.1
+LIR	42.0	16.7
+LIR+GCA	48.2	18.2
+LIR+GCA+PDC	50.3	18.6
+LIR+GCA+PDC+CAM	**57.4**	**23.5**
SQSG trained on KITTI w/o intensity [2]	57.1	-

SQSG denotes SqueezeSeg. **+LIR** denotes using learned intensity rendering. **+GCA** denotes using geodesic correlation alignment. **+PDC** denotes using progressive domain calibration. **+CAM** denotes using the CAM module.

boosts segmentation of cars, whereas the mask channel boosts segmentation of cyclists. (2) Focal loss improves segmentation of pedestrians and cyclists. The number of points corresponding to pedestrians and cyclists is low relative to the large number of background points. This class imbalance causes the network to focus less on the pedestrian and cyclist classes. Focal loss mitigates this problem by focusing the network on optimization of these two categories. (3) CAM significantly improves the performance of all the classes by reducing the network's sensitivity to dropout noise.

C. *Domain Adaptation Pipeline*

The performance comparisons, measured in IoU, between the proposed domain adaptation pipeline and baselines are shown in Table II. Some segmentation results are shown in Fig. 8. From the results, we have the following observations. (1) Models trained on the source domain without any adaptation does not perform well. Due to the influence of *domain discrepancy*, the joint probability distributions of observed LiDAR and road-objects greatly differ in the two domains. This results in the model's low transferability from the source domain to the target domain. (2) All adaptation methods are effective, with the combined pipeline performing the best, demonstrating its effectiveness. (3) Adding the CAM to the network also significantly boosts the performance on the real data, supporting our hypothesis that dropout noise is a significant source of domain discrepancy. Therefore, improving the network to make it more robust to dropout noise can help reduce the domain gap. (4) Compared with [2] where a SqueezeSeg model is trained on the real KITTI dataset but without intensity, our SqueezeSegV2 model trained purely on synthetic data and unlabeled real data achieves a better accuracy, showing the effectiveness of our domain adaptation training pipeline. (5) Compared with our latest SqueezeSegV2 model trained on the real KITTI dataset, there is still an obvious performance gap. Adapting the segmentation model from synthetic LiDAR point clouds is still a challenging problem.

VI. CONCLUSION

In this paper, we proposed SqueezeSegV2 with better segmentation performance than the original SqueezeSeg and a domain adaptation pipeline with stronger transferability. We designed a context aggregation module to mitigate the impact of dropout noise. Together with other improvements such as focal loss, batch normalization and a LiDAR mask channel, SqueezeSegV2 sees accuracy improvements of 6.0% to 8.6% in various pixel categories over the original SqueezeSeg. We also proposed a domain adaptation pipeline with three components: learned intensity rendering, geodesic correlation alignment, and progressive domain calibration. The proposed pipeline significantly improved the real world accuracy of the model trained on synthetic data by 28.4%, even outperforming a baseline model [2] trained on the real dataset.

ACKNOWLEDGEMENT

This work is partially supported by Berkeley Deep Drive (BDD), and partially sponsored by individual gifts from Intel and Samsung. We would like to thank Alvin Wan and Ravi Krishna for their constructive feedback.

References

[1] F. Moosmann, O. Pink, and C. Stiller, “Segmentation of 3d lidar data in non-flat urban environments using a local convexity criterion,” in *IV*, 2009, pp. 215–220.

[2] B. Wu, A. Wan, X. Yue, and K. Keutzer, “Squeezeseg: Convolutional neural nets with recurrent crf for real-time road-object segmentation from 3d lidar point cloud,” in *ICRA*, 2018.

[3] A. Torralba and A. A. Efros, “Unbiased look at dataset bias,” in *CVPR*, 2011, pp. 1521–1528.

[4] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, “Focal loss for dense object detection,” *IEEE TPAMI*, 2018.

[5] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” in *ICML*, 2015, pp. 448–456.

[6] P. Morerio, J. Cavazza, and V. Murino, “Minimal-entropy correlation alignment for unsupervised deep domain adaptation,” in *ICLR*, 2018.

[7] B. Douillard, J. Underwood, N. Kuntz, V. Vlaskine, A. Quadros, P. Morton, and A. Frenkel, “On the segmentation of 3d lidar point clouds,” in *ICRA*, 2011, pp. 2798–2805.

[8] D. Zermas, I. Izzat, and N. Papanikolopoulos, “Fast segmentation of 3d point clouds: A paradigm on lidar data for autonomous vehicle applications,” in *ICRA*, 2017, pp. 5067–5073.

[9] F. Piewak, P. Pinggera, M. Schäfer, D. Peter, B. Schwarz, N. Schneider, D. Pfeiffer, M. Enzweiler, and M. Zöllner, “Boosting lidar-based semantic labeling by cross-modal training data generation,” *arXiv preprint arXiv:1804.09915*, 2018.

[10] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, “Pointnet: Deep learning on point sets for 3d classification and segmentation,” in *CVPR*, 2017, pp. 77–85.

[11] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, “Pointnet++: Deep hierarchical feature learning on point sets in a metric space,” in *NIPS*, 2017, pp. 5099–5108.

[12] C. R. Qi, W. Liu, C. Wu, H. Su, and L. J. Guibas, “Frustum pointnets for 3d object detection from rgb-d data,” *arXiv preprint arXiv:1711.08488*, 2017.

[13] V. M. Patel, R. Gopalan, R. Li, and R. Chellappa, “Visual domain adaptation: A survey of recent advances,” *IEEE SPM*, vol. 32, no. 3, pp. 53–69, 2015.

[14] G. Csurka, “Domain adaptation for visual applications: A comprehensive survey,” *arXiv:1702.05374*, 2017.

[15] M. Long, Y. Cao, J. Wang, and M. Jordan, “Learning transferable features with deep adaptation networks,” in *ICML*, 2015, pp. 97–105.

[16] B. Sun, J. Feng, and K. Saenko, “Correlation alignment for unsupervised domain adaptation,” in *Domain Adaptation in Computer Vision Applications*, 2017, pp. 153–171.

[17] J. Zhuo, S. Wang, W. Zhang, and Q. Huang, “Deep unsupervised convolutional domain adaptation,” in *ACM MM*, 2017, pp. 261–269.

[18] Y. Zhang, P. David, and B. Gong, “Curriculum domain adaptation for semantic segmentation of urban scenes,” in *ICCV*, 2017, pp. 2039–2049.

[19] M.-Y. Liu and O. Tuzel, “Coupled generative adversarial networks,” in *NIPS*, 2016, pp. 469–477.

[20] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky, “Domain-adversarial training of neural networks,” *JMLR*, vol. 17, no. 1, pp. 2096–2030, 2016.

[21] E. Tzeng, J. Hoffman, K. Saenko, and T. Darrell, “Adversarial discriminative domain adaptation,” in *CVPR*, 2017, pp. 2962–2971.

[22] A. Shrivastava, T. Pfister, O. Tuzel, J. Susskind, W. Wang, and R. Webb, “Learning from simulated and unsupervised images through adversarial training,” in *CVPR*, 2017, pp. 2242–2251.

[23] K. Bousmalis, N. Silberman, D. Dohan, D. Erhan, and D. Krishnan, “Unsupervised pixel-level domain adaptation with generative adversarial networks,” in *CVPR*, 2017, pp. 3722–3731.

[24] J. Hoffman, E. Tzeng, T. Park, J.-Y. Zhu, P. Isola, K. Saenko, A. A. Efros, and T. Darrell, “Cycada: Cycle-consistent adversarial domain adaptation,” in *ICML*, 2018.

[25] M. Ghifary, W. Bastiaan Kleijn, M. Zhang, and D. Balduzzi, “Domain generalization for object recognition with multi-task autoencoders,” in *ICCV*, 2015, pp. 2551–2559.

[26] M. Ghifary, W. B. Kleijn, M. Zhang, D. Balduzzi, and W. Li, “Deep reconstruction-classification networks for unsupervised domain adaptation,” in *ECCV*, 2016, pp. 597–613.

[27] S. R. Richter, V. Vineet, S. Roth, and V. Koltun, “Playing for data: Ground truth from computer games,” in *ECCV*, 2016, pp. 102–118.

[28] M. Johnson-Roberson, C. Barto, R. Mehta, S. N. Sridhar, K. Rosaen, and R. Vasudevan, “Driving in the matrix: Can virtual worlds replace human-generated annotations for real world tasks?” in *ICRA*, 2017, pp. 746–753.

[29] X. Yue, B. Wu, S. A. Seshia, K. Keutzer, and A. L. Sangiovanni-Vincentelli, “A lidar point cloud generator: from a virtual world to autonomous driving,” in *ICMR*, 2018, pp. 458–464.

[30] S. R. Richter, Z. Hayder, and V. Koltun, “Playing for benchmarks,” in *ICCV*, 2017, pp. 2232–2241.

[31] P. Krähenbühl, “Free supervision from video games,” in *CVPR*, 2018, pp. 2955–2964.

[32] J. Hu, L. Shen, and G. Sun, “Squeeze-and-excitation networks,” in *CVPR*, 2018, pp. 7132–7141.

[33] A. Geiger, P. Lenz, and R. Urtasun, “Are we ready for autonomous driving? the kitti vision benchmark suite,” in *CVPR*, 2012, pp. 3354–3361.

[34] Y. Li, N. Wang, J. Shi, X. Hou, and J. Liu, “Adaptive batch normalization for practical domain adaptation,” *PR*, vol. 80, pp. 109–117, 2018.

[35] Y. Wang, T. Shi, P. Yun, L. Tai, and M. Liu, “Pointseg: Real-time semantic segmentation based on 3d lidar point cloud,” *arXiv preprint arXiv:1807.06288*, 2018.