HYLDA: End-to-end Hybrid Learning Domain Adaptation for LiDAR Semantic Segmentation

Eduardo R. Corral-Soto, Migank Rochan, Yannis Y. He, Shubhra Aich, Yang Liu, Liu Bingbing

Abstract—In this paper we address the problem of training a LiDAR semantic segmentation network using a fully-labeled source dataset and a target dataset that only has a small number of labels. To this end, we develop a novel image-to-image translation engine, and couple it with a LiDAR semantic segmentation network, resulting in an integrated domain adaptation architecture we call HYLDA. To train the system end-to-end, we adopt a diverse set of learning paradigms, including 1) self-supervision on a simple auxiliary reconstruction task, 2) semi-supervised training using a few available labeled target domain frames, and 3) unsupervised training on the fake translated images generated by the image-to-image translation stage, together with the labeled frames from the source domain. In the latter case, the semantic segmentation network participates in the updating of the image-to-image translation engine. We demonstrate experimentally that HYLDA effectively addresses the challenging problem of improving generalization on validation data from the target domain when only a few target labeled frames are available for training. We perform an extensive evaluation where we compare HYLDA against strong baseline methods using two publicly available LiDAR semantic segmentation datasets.

I. INTRODUCTION AND PRIOR WORK

Deep neural networks designed for LiDAR perception in autonomous driving and robotics have shown a great success in the last few years, which can be partly attributed to the availability of large labeled datasets for training. However, these networks usually fail to generalize with unseen data from a similar but different domain due to the domain shift. To mitigate this problem, a training dataset from the new domain can be captured and labeled to re-train the network. However, this process is time-consuming and expensive, especially for the point-level semantic segmentation task. In this work, we are given two datasets captured by different sensors at different geographic locations. The first one (source) is fully-labeled, while the second one (target) has only a small number of labeled frames. Our goal is to train a semantic segmentation network with improved generalization on the validation dataset from the target domain. There exist a number of techniques in the literature to address this problem, including transfer learning and fine-tuning, where the network is first trained on the source dataset, and then further trained (i.e. refined) using the available labeled target frames [2], [3], [4], [5], [6], [7], [8]. On the other hand, there is research in domain adaptation that focuses on reducing the domain shift between two or more domains. According to the taxonomy paper [9], domain adaptation methods can be classified into: 1) Discrepancy-based methods, which include the Maximum Mean Discrepancy (MMD) [10] and DeepCORAL [11], which minimize the global mean or covariance matrix discrepancy between the source and target domains, 2) Adversarial-based methods, which typically employ GANs and discriminators to reduce the domain shift via domain translation, and 3) Reconstruction-based methods, which use auxiliary reconstruction tasks to encourage feature invariance. A more recent LiDAR-focused domain adaptation survey [12] classifies methods into: 1) Domain-invariant data representation methods [13], [14], mainly based on hand-crafted data preprocessing to move different domains into a common representation (e.g. LiDAR data rotation and normalization), 2) Domain-invariant feature learning for finding a common representation space for the source and target domains [15], [16], 3) Normalization statistics that attempt to align the domain distributions by a normalization of the mean and variance of activations, and 4) Domain mapping, where source data is transformed, usually using GANs or adversarial training to appear like target data [17], [18], [19]. A few examples of domain mapping methods in the context of training data generation from simulated synthetic frames are [17], [18], [20], where a CycleGAN [21] is trained using unpaired simulated and real projected bird’s eye view (BEV) images to generate pseudo-labeled simulated data for off-line training of a BEV YOLOv3 [22] object detection network. SqueezeSegV2 [23] is another example that uses simulators to generate large quantities of labeled spherical projections of synthetic LiDAR data to train perception models. Other methods such as the LiDARNet [15] and the LCP [1] integrate CycleGAN into a task network for training on real LiDAR source and target datasets. LiDARNet works on LiDAR range images (spherical projection) concatenated with reflectivity, 2D coordinates, and normal maps to train a semantic segmentation network. The LCP integrates and adapts CycleGAN to operate with 64-channel feature pseudo-images for 3D object detection using PointPillars [24].

In contrast to the above methods, in this paper we present a novel image-to-image translation engine, and couple it with a LiDAR semantic segmentation network [25], resulting in an integrated domain adaptation architecture we call HYLDA. To train the system end-to-end, we adopt a diverse set of learning paradigms, including self-supervision on a simple auxiliary reconstruction task, semi-supervised training using the few available labeled target domain frames, and unsupervised training on the fake translated images generated by the image-to-image translation stage together with the labeled frames from the source domain.
### Summary of our contributions:

1) A novel image-to-image translation engine with generators split into separate encoder and decoder with configurable skip connections that allow us to make use of different learning paradigms to train the generators, including: 1) self-supervision, where we first cross-connect the skip connections between encoder and decoder from different generators, and 2) adversarial training, where we connect skip connections between encoder and decoder from the same generator.

2) A hybrid learning LiDAR perception task stage composed of encoder-decoder semantic segmentation networks, where a network is incrementally trained following different learning paradigms such as self-supervision on an auxiliary reconstruction task, semi-supervised training using a small number of labeled frames available, and finally, unsupervised training where the inputs are fake translated images from the image-to-image translation engine.

We present an extensive evaluation section where we compare HYLDA against strong baseline methods using two publicly available LiDAR semantic segmentation datasets. We also present experimental results from ablation studies.

### II. Problem being addressed

Due to the domain shift induced by the sensor and geographical differences in the given datasets, a model trained on the source dataset does not usually generalize well when evaluated directly on the target dataset, as shown by rows 2 and 13 from Table I. Therefore, we formulate our problem as follows: Given two datasets captured by different LiDAR sensors at different geographic locations, where the first one is fully-labeled, and the second one has only a small number of labeled frames, we seek to develop an architecture and training strategy that result in a trained model which improves generalization and performs well on validation data from the target domain.

### III. Our method

Our proposed HYLDA architecture, shown in Fig. 1, has three basic stages: 1) Input pre-processing, 2) our image-to-image translation engine, and 3) the task stage composed of LiDAR semantic segmentation networks.

#### A. Pre-processing

The inputs to HYLDA are two (source and target) LiDAR 3D point clouds and available labels. The point clouds are converted and normalized into spherical projection range view (RV) images of size $64 \times 2048 \times 5$ ($X$, $Y$, $Z$, range, remission) as done in [25], [23], [26], [27].

#### B. Image-to-image translation (i2i) engine

Our image-to-image translation engine is inspired by the CycleGAN and LCP methods [21], [1] where mapping functions $G: X \to Y$, and $F: Y \to X$ are learned using two discriminators $D_X$ and $D_Y$ and adversarial training. Here $Y$ is the source domain, and $X$ is the target domain as shown in Fig. 1.

Split generator encoder and decoder with configurable skip connections. We split each generator into separate encoder and decoder, and introduce configurable skip connections. This design allows us to make use of different learning paradigms to train our generators, including a) self-supervision where we cross-connect the skip connections between the encoder and decoder from different generators, and b) adversarial training, using intra-generator skip connections.

---

**Fig. 1:** HYLDA integrated end-to-end architecture, which has three basic stages: 1) Input pre-processing, 2) our image-to-image translation engine, and 3) the task stage composed of LiDAR semantic segmentation networks. See text for details.
Fig. 2: Dual-head PatchGAN. We modify the PatchGAN to introduce one additional classification head connected to an earlier (third) feature layer to “criticize” the fake outputs from the generator at two resolutions.

**Dual-head discriminator.** In the TSIT method [21], three PatchGAN [28] discriminators with identical architecture are instantiated to process different downscaled versions of a feature layer. In contrast, we instantiate a single PatchGAN discriminator, but modify its structure to introduce one additional classification head connected to an earlier feature layer from the same PatchGAN network as shown in Fig. 2. The goal of the dual-head PatchGAN discriminator is to “criticize” (at two resolutions) the fake outputs from the generator to provide more detailed feedback via backpropagation. For this, we upscale (to a common scale) and concatenate the head outputs to compute the LSGAN loss as described in section IV.

**Domain statistics loss.** In order to encourage the generators to output fake translated images that adhere to the underlying statistics from the real input dataset, we implement a statistics loss that combines DeepCORAL [11] and MMD [10], by taking into account both the covariance matrix and mean. We start by precomputing (jointly) the covariance matrix and mean image \((\Sigma_s, \mu_s)\) for the whole source dataset, and target dataset \((\Sigma_t, \mu_t)\). During training, the mean and covariance matrix \((\Sigma_t, \mu_t)\) are computed from each batch of fake translated images, and compared against their corresponding pre-computed references as shown in Fig. 1 by computing a statistics loss as described in section IV.

**C. Task stage (semantic segmentation)**

We focus on the LiDAR semantic segmentation task, and select the well-known SALSANext model [25] to develop our method. SALSANext receives a LiDAR 3D point cloud as an input, and converts it into a \(64 \times 2048 \times 5\) range view image as described in Section III-A. Its output is a \(64 \times 2048\) predicted pixel-level semantic class label map. We modified SALSANext to support the 11 semantic classes described in section IV-B.

**Reference source semantic segmentation network.** Inspired by the semantic consistency loss from CyCADA [29], our design consists of two instances of SALSANext. The first instance, which we call \(f_{RefSRC}\), is a semantic segmentation network that is decoupled from the rest of the system. We pre-train \(f_{RefSRC}\) using standard supervised learning using the fully-labeled source domain dataset, and keep its parameters fixed after training.

**Target semantic segmentation network.** We define \(f_{target}\) as the semantic segmentation network that needs to be trained to perform well on validation data from the target domain. We initialize \(f_{target}\) with the weights from the pre-trained \(f_{RefSRC}\), and split it into encoder \(f_{enc}\) and decoder so we can train \(f_{enc}\) that is coupled with an auxiliary decoder \(D_{encaux}\) (see Fig. 1) using self-supervision [27], [30]. \(D_{encaux}\) is a SALSANext decoder with a \(tanh\) activation at the output, which is adapted to work on a simple auxiliary identity reconstruction task. We then train \(f_{target}\) (encoder and decoder) using both, semi-supervised training using a few labeled frames, and unsupervised using fake translated images and labels from the source dataset.

**IV. HYLDA LOSSES AND TRAINING PROCEDURE**

Here we describe the sequential intermediate steps (6 in total) that execute for each batch (step) during the training of HYLDA, as well as the losses employed at each of the steps.

1) **Perform self-supervision of image-to-image translation generator encoders and decoders.** First, we cross-connect (concatenate) the skip connections (at two resolution feature layers) between the source encoder \(Enc_s\) and decoder \(Dec_s\), and between the target encoder \(Enc_t\) and decoder \(Dec_t\). Then, we input the source image \(y\) into \(Enc_s\) so that \(Dec_s\) generates a reconstructed identity image \(y^*\). Similarly, a target image \(x\) is fed into \(Enc_t\) to obtain a reconstructed image \(x^*\). We then use Eqn. (1) to compute the self-supervision auxiliary identity loss \(\mathcal{L}_{self}\) that we use to update the encoders and decoders from the generator.

\[
\mathcal{L}_{self} = \mathbb{E}_{y \sim p_{data}(y)} [||Dec_s(Enc_s(y)) - y||_1] + \mathbb{E}_{x \sim p_{data}(x)} [||Dec_t(Enc_t(x)) - x||_1].
\]

2) **Train the generators and discriminators with adversarial training.** We now connect (concatenate) the skip connections between the source encoder \(Enc_s\) and decoder \(Dec_t\), and between the target encoder \(Enc_t\) and decoder \(Dec_s\). Then feed the source image \(y\) through the source encoder \(Enc_s\). The output from \(Dec_t\) is a translated (fake) image \(x^*\), which is fed into the dual-head discriminator \(D_X\) to measure how realistic the generated fake image looks like compared to real images. A similar process is repeated by feeding the target image \(x\) through \(Enc_t\) to obtain a translated (fake) image \(y^*\) that is fed into the discriminator \(D_Y\). We then use Eqn. (2) to compute the LSGAN loss [31], [21], [1].

\[
\mathcal{L}_{LSGAN}(G, D_Y, X) = \mathbb{E}_{x \sim p_{data}(x)} [(D_Y(G(x)) - 1)^2],
\]
\[
\mathcal{L}_{LSGAN}(F, D_X, Y) = \mathbb{E}_{y \sim p_{data}(y)} [(D_X(F(y)) - 1)^2].
\]

where \(\mathcal{L}_{LSGAN}(G, D_Y, X)\) and \(\mathcal{L}_{LSGAN}(F, D_X, Y)\) are the least squares GAN losses used to train the generators.

3) **Compute the batch statistics loss for fake translated images.** We compute the mean and covariance matrix from the fake translated image batches, and measure the statistics similarity loss \(\mathcal{L}_{stats}\) using Eqn. (3).
\[ L_{stats} = \left( \mathbb{E}_{y \sim p_{data}(y)} \left[ \left\| \Sigma_y - \Sigma_s \right\|_1 \right] + \mathbb{E}_{x \sim p_{data}(x)} \left[ \left\| \Sigma_x - \Sigma_t \right\|_1 \right] \right) + \left( \mathbb{E}_{y \sim p_{data}(y)} \left[ \left\| \mu_y - \mu_s \right\|_1 \right] + \mathbb{E}_{x \sim p_{data}(x)} \left[ \left\| \mu_x - \mu_t \right\|_1 \right] \right) \] (3)

Finally, we compute the image-to-image loss \( L_{img2img} \) by adding the LSGAN and statistics loss using a scalar weight \( \beta \) as shown in Eqn. (4). We then update the whole image-to-image translation engine using this loss. We set \( \beta = 0.1 \) experimentally.

\[ L_{img2img} = L_{LSGAN}(G, D_Y, X) + \beta \times L_{stats}. \] (4)

4) Self-supervised training of target semantic segmentation encoder. First, we connect all skip connections from \( f_{enc} \) to the auxiliary identity reconstruction decoder \( Dec_{aux} \). Then, we feed a real target frame \( x \) into \( f_{enc} \), which propagates feature information into \( Dec_{aux} \) to output a reconstructed \( x \). Next, we measure the identity similarity \( L_1 \) loss using Eqn. (5), and update both \( f_{enc} \) and \( Dec_{aux} \).

\[ L_{ssnetself} = \mathbb{E}_{x \sim p_{data}(x)} \left[ \left\| Dec_{aux}(f_{enc}(x)) - x \right\|_1 \right]. \] (5)

5) Semi-supervised training of target semantic segmentation encoder and decoder. If a limited/small number (e.g. 100) of labeled target domain frames is available, we update \( f_{target} \) (encoder and decoder) via standard supervised learning using the available labeled frames and we compute the cross-entropy loss \( L_{ce}(y, \hat{y}) \) from Eqn. (6) as done in [25].

\[ L_{ce}(y, \hat{y}) = -\sum_i \alpha_i p(y_i) \log(p(\hat{y}_i)). \] (6)

6) Unsupervised training of target semantic segmentation encoder and decoder. Now we input the translated fake target image batch \( x^* \) into the network \( f_{target} \). It predicts a semantic segmentation map as an output. We use this map to compute two losses: 1) The segmentation task cross-entropy loss (Eqn. (7)) with respect to the source ground-truth label maps, and 2) The semantic task consistency loss \( L_{sem} \) which encourages \( f_{target} \) to output segmentation maps that are as similar as possible to those from \( f_{RefSrc} \). We aggregate these two losses using Eqn. (8), where \( \gamma \) is a scalar hyperparameter that we set to 1 experimentally. We use \( L_{unsupervised} \) to back-propagate through the image-to-image translation engine and update it so that it learns to generate better fake translated images in the next iteration.

\[ L_{uwce}(y, \hat{y}) = -\sum_i \alpha_i p(y_i) \log(p(\hat{y}_i)). \] (7)

\[ L_{sem} = \mathbb{E}_{y \sim p_{data}(y)} \left[ \left\| f_{RefSrc}(y) - f_{target}(x^*) \right\|_1 \right]. \] (8)
\( \mathcal{L}_{\text{unsupervised}} = \mathcal{L}_{\text{uwce}}(y, \hat{y}) + \gamma \times \mathcal{L}_{\text{sem}} \). 

We note that the spatial consistency between the fake image \( x^* \) and the source domain label map is encouraged in two ways: 1) By the skip connections between \( \text{Enc}_s \) and \( \text{Dec}_s \), and between \( \text{Enc}_t \) and \( \text{Dec}_t \), which propagate spatial information into the decoders, and 2) Through the semantic task consistency loss \( \mathcal{L}_{\text{sem}} \), which encourages \( f_{\text{target}} \) to output semantic maps that are spatially consistent with the semantic maps from \( f_{\text{RefSrc}} \). This enables us to leverage the source labels with the fake translated images \( x^* \).

A. Training details

To train HYLDA, we use the Adam optimizer with a constant learning rate of \( \eta_{bs} = 0.01 \) for \( f_{\text{target}} \), \( f_{\text{RefSrc}} \) and \( \text{Dec}_{\text{aux}} \) with data augmentation as done in \cite{25}. The image-to-image translation engine is trained using Stochastic Gradient Descend (SGD) with a constant learning rate of \( \eta_{kl} = 0.002 \). We train HYLDA for 75 epochs on Nvidia Tesla V100 GPUs, each with 32510 MB memory. Training on 8 GPUs with batch size 2 per GPU, takes approx. 2 hours per epoch for the domain adaptation in the SemanticKITTI \( \rightarrow \) nuScenes direction.

V. Experiments and evaluations

In inference mode, unseen frames \( x \) from the target domain validation dataset are directly fed into \( f_{\text{target}} \).

A. Datasets and domains

We study and develop our domain adaptation method using two popular LiDAR semantic segmentation datasets, namely SemanticKITTI \cite{32} and nuScenes \cite{33}. SemanticKITTI has 19130 labeled frames (point-level semantic class ID) for training, and 4071 labeled validation frames for evaluation. nuScenes has 28130 labeled frames for training, and 6019 labeled validation frames.

B. Segmentation class mapping between datasets

LiDAR semantic segmentation networks such as \cite{25}, \cite{23}, \cite{26} are usually trained and evaluated on datasets such as SemanticKITTI and nuScenes. Since these datasets are organized differently, a class mapping step is usually needed to compare domain adaptation performance on these datasets. We follow prior work \cite{27} for this and find 11 overlapping object classes in these datasets. The object classes are: \{Car, Bicycle, Motorcycle, Other vehicle, Pedestrian, Truck, Drivable surface, Sidewalk, Terrain, Vegetation, Man-made\}. All other objects fall under a background class.

C. Source and target dataset preparation

To train HYLDA, we utilize all of the available frames and labels from both \( Y \) (source) and \( X \) (target) datasets. Let \( N = |Y| \) and \( M = |X| \) be the number of available raw data frames for the source and target domains respectively. And let \( N_{gt} \) and \( M_{gt} \) be their corresponding number of labeled frames. For the source domain \( Y \) all frames are labeled \( (N = N_{gt}) \). On the other hand, in our problem, we are given only a few labeled target domain frames \( (M_{gt} \ll M) \). To simulate the availability of a small number of labeled frames using SemanticKITTI and nuScenes (where all training frames are labeled), we use random sampling of the target set \( X \) to select a subset of labeled frames. Specifically, we define the following target domain training subsets: \( X_{100}, X_{250}, \) and \( X_{500} \), where \( X_{100} \subset X_{250} \subset X_{500} \), and the subscripts denote the number of labeled frames.

D. Baseline preparation

We compared our HYLDA method against two strong baselines. The first baseline is Fine-tuning, where we initialize the network \( f_{\text{target}} \) with the pre-trained source \( f_{\text{RefSrc}} \) network weights, and train/refine \( f_{\text{target}} \) with each available subset of labeled target frames \( (X_{100}, X_{250}, \) and \( X_{500} ) \) for 100 epochs. The second baseline is the LCP method from \cite{1}, which is a state-of-the-art LiDAR domain adaptation method. The LCP uses CycleGAN as the image-to-image translation engine and the PointPillars 3D object detector \cite{24} as the task network, where both of them operate on 64-channel feature pseudo-images. We implemented the LCP within HYLDA by replacing our image-to-image translation with the CycleGAN image-to-image translation stage from the LCP, and modified it to operate with \( 64 \times 2048 \times 5 \) range view images. We then adapted it to work with the semantic segmentation network \( f_{\text{target}} \). Note that this LCP-based baseline does not use either of the auxiliary decoder \( \text{Dec}_{\text{aux}} \), reference network \( f_{\text{RefSrc}} \), nor semantic consistency loss. Moreover, the CycleGAN generators don’t have skip connections, and the discriminators are a single instance of the PatchGAN discriminator working at a single resolution.

E. Evaluation

We trained HYLDA in two domain adaptation directions: SemanticKITTI \( \rightarrow \) nuScenes \((K \rightarrow N)\), and nuScenes \( \rightarrow \) SemanticKITTI \((N \rightarrow K)\), under three scenarios of the labeled subsets of target frames \( X_{100}, X_{250}, \) and \( X_{500} \). We perform the evaluation on the validation dataset from the target domain using the Mean Intersection of the Union (mIoU) metric as done in \cite{25}, \cite{23}, \cite{26}. The results are summarized in Table 1. Rows 1 and 12 show the results for the oracles, where \( f_{\text{target}} \) is trained with all of the target \( M = M_{gt} \) labeled frames and evaluated on the target validation dataset. Rows 2 and 13 show the results of naïve training with the \( N = N_{gt} \) source labeled frames and evaluating on the target validation dataset. Note that training on the source domain and directly evaluating on the target domain results in poor generalization, which motivates the study of domain adaptation methods. Rows 3 through 11, and 14 through 22 show the results obtained with the Fine-tuning and LCP baselines, as well as with our proposed HYLDA method. We observe that none of the models is able to outperform the oracle, which is expected. In particular, on average, HYLDA is below the oracle by 23.1% in the \( K \rightarrow N \) direction and by 22.6% in the \( N \rightarrow K \) direction. We further observe that the performance of all models tends to
improve as the number of labeled target frames is increased. For the K → N direction, both, the LCP and HYLDA are consistently better than Fine-tuning. On average, HYLDA outperforms Fine-tuning by 6.4% in the K → N direction, and by 5.3% in the N → K direction. For the K → N direction HYLDA outperforms the LCP for X_{100} and X_{250} by 2.3% and 2.7% respectively. However, for X_{500}, LCP was better than HYLDA by 3.6%. On the other hand, for the nuScenes → SemanticKITTI direction we observe that the LCP results are inferior to the Fine-tuning baseline. In this direction HYLDA clearly outperforms the LCP by an average of 12.3%. By looking at Table I, we observe that the LCP had problems with four classes: bicycles, motorcycles, trucks, and other vehicles, whereas HYLDA achieves a much stronger performance on these classes demonstrating the importance of its modules. Fig. 3 shows qualitative results for two frames from the target validation set. In this figure, we observe that the segmentation maps produced by HYLDA are more accurate and visually closer to the oracle than the alternative methods, namely Naïve, Fine-tuning, and LCP.

F. Ablation studies

We conduct an ablation study on the N → K direction with the X_{250} labeled target domain semi-supervision dataset. Starting from the LCP configuration, we enable the following stages in HYLDA incrementally: 1) Image-to-image translation, including dual-head discriminator, skip connection mechanism, and statistics loss, 2) Auxiliary decoder (self-supervision of f_{enc}), and 3) Semantic consistency loss, where at this point the configuration corresponds to HYLDA. The bar plot in Fig. 4 summarizes the results of this study. We observe that the replacing the LCP CycleGAN-based image-to-image translation with our proposed HYLDA image-to-image translation engine results in a clear improvement of 2.8%. Enabling self-supervision (f_{end} and D_{encaux}) results in a more moderate contribution of 0.3%. However, when the semantic consistency loss is enabled together with all of the elements of HYLDA, the performance improved by 6.1%. We observed that the contributions from both, the dual-head discriminator and the statistics loss are meaningful when used in conjunction with other modules, instead of when examined individually.

VI. CONCLUSIONS AND FINAL REMARKS

We have presented HYLDA, a novel domain adaptation framework for LiDAR semantic segmentation that successfully addressed the challenging problem of improving generalization on validation data from a target domain with only a few available target labeled frames. We have demonstrated the effectiveness of our method on two publicly available LiDAR datasets, where HYLDA outperformed two strong domain adaptation baselines. We have also conducted ablation studies which showed that the main performance contributions in HYLDA can be attributed to our new image-to-image translation engine and to the use of the semantic consistency loss, combined with the use of multiple learning paradigms. We believe that our framework is generic enough to be used in other applications and tasks, such as 3D object detection.
REFERENCES

[1] E. R. Corral-Soto, A. Nabatchian, M. Gerdzhev, and L. Bingbing, “Lidar few-shot domain adaptation via integrated cyclegan and 3d object detector with joint learning delay,” in 2021 International Conference on Robotics and Automation (ICRA). IEEE, 2021.

[2] Y. Bengio, “Deep learning of representations for unsupervised and transfer learning,” in Proceedings of ICML workshop on unsupervised and transfer learning, 2012, pp. 17–36.

[3] I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning. MIT press, 2016.

[4] L. Torrey and J. Shavlik, “Transfer learning,” in Handbook of research on machine learning applications and trends: algorithms, methods, and techniques. IGI global, 2010, pp. 242–264.

[5] S. J. Pan and Q. Yang, “A survey on transfer learning,” IEEE Transactions on knowledge and data engineering, vol. 22, no. 10, pp. 1345–1359, 2009.

[6] N. Tajbakhsh, J. Y. Shin, S. R. Gurudu, R. T. Hurst, C. B. Kendall, M. B. Gotway, and J. Liang, “Convolutional neural networks for medical image analysis: Full training or fine tuning?” IEEE transactions on medical imaging, vol. 35, no. 5, pp. 1299–1312, 2016.

[7] R. Caruana, “Learning many related tasks at the same time with back-propagation,” in Advances in neural information processing systems, 1995, pp. 657–664.

[8] E. R. Corral-Soto and L. Bingbing, “Understanding strengths and weaknesses of complementary sensor modalities in early fusion for object detection,” in 2020 IEEE Intelligent Vehicles Symposium (IV). IEEE, pp. 1785–1792.

[9] M. Wang and W. Deng, “Deep visual domain adaptation: A survey,” Neurocomputing, vol. 312, pp. 135–153, 2018.

[10] A. Gretton, K. Borgwardt, M. Rasch, B. Schölkopf, and A. Smola, “A kernel method for the two-sample-problem,” Advances in neural information processing systems, vol. 19, pp. 513–520, 2006.

[11] B. Sun and K. Saenko, “Deep coral: Correlation alignment for deep domain adaptation,” in European conference on computer vision. Springer, 2016, pp. 443–450.

[12] L. T. Triess, M. Dreissig, C. B. Rist, and J. M. Zöllner, “A survey on deep domain adaptation for lidar perception,” arXiv preprint arXiv:2106.02377, 2021.

[13] C. B. Rist, M. Enzweiler, and D. M. Gavrila, “Cross-sensor deep domain adaptation for lidar detection and segmentation,” in 2019 IEEE Intelligent Vehicles Symposium (IV). IEEE, 2019, pp. 1535–1542.

[14] I. Alonso, L. R. Montesano, A. C. Murillo, et al., “Domain adaptation in lidar semantic segmentation,” arXiv preprint arXiv:2010.12239, 2020.

[15] P. Jiang and S. Saripalli, “Lidarinet: A boundary-aware domain adaptation model for lidar point cloud semantic segmentation,” arXiv preprint arXiv:2003.01174, 2020.

[16] M. Jaritz, T.-H. Vu, R. d. Charette, E. Wirbel, and P. Pérez, “xmuada: Cross-modal unsupervised domain adaptation for 3d semantic segmentation,” in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 12605–12614.

[17] K. Saleh, A. Abobakr, M. Attia, J. Iskander, D. Nahavandi, M. Hossny, and S. Nahavandi, “Domain adaptation for vehicle detection from bird’s eye view lidar point cloud data,” in Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops, 2019, pp. 0–0.

[18] A. E. Sallab, I. Sobh, M. Zahran, and N. Essam, “Lidar sensor modeling and data augmentation with gans for autonomous driving,” arXiv preprint arXiv:1905.07290, 2019.

[19] S. Zhao, Y. Wang, B. Li, B. Wu, Y. Gao, P. Xu, T. Darrell, and K. Keutzer, “epointda: An end-to-end simulation-to-real domain adaptation framework for lidar point cloud segmentation,” in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 35, no. 4, 2021, pp. 3500–3509.

[20] A. E. Sallab, I. Sobh, M. Zahran, and M. Shawky, “Unsupervised neural sensor models for synthetic lidar data augmentation,” arXiv preprint arXiv:1911.10575, 2019.

[21] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, “Unpaired image-to-image translation using cycle-consistent adversarial networks,” in Proceedings of the IEEE international conference on computer vision, 2017, pp. 2223–2232.

[22] J. Redmon and A. Farhadi, “Yolov3: An incremental improvement,” arXiv preprint arXiv:1804.02767, 2018.

[23] B. Wu, X. Zhou, S. Zhao, X. Yue, and K. Keutzer, “Squeezesegv2: Improved model structure and unsupervised domain adaptation for road-object segmentation from a lidar point cloud,” in 2019 International Conference on Robotics and Automation (ICRA). IEEE, 2019, pp. 4376–4382.

[24] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom, “Pointpilars: Fast encoders for object detection from point clouds,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 12697–12705.

[25] T. Cortinhal, G. Tzelepis, and E. E. Aksoy, “Salsanext: fast, uncertainty-aware semantic segmentation of lidar point clouds for autonomous driving,” arXiv preprint arXiv:2003.03653, 2020.

[26] M. Gerdzhev, R. Razani, E. Taghavi, and B. Liu, “Tornado-net: multiview total variation semantic segmentation with diamond inception module,” arXiv preprint arXiv:2008.10544, 2020.

[27] M. Rochan, S. Aich, E. R. Corral-Soto, A. Nabatchian, and B. Liu, “Unsupervised domain adaptation in lidar semantic segmentation with self-supervision and gated adapters,” arXiv preprint arXiv:2107.09783, 2021.

[28] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, “Image-to-image translation with conditional adversarial networks,” in Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 1125–1134.

[29] 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,” arXiv preprint arXiv:1711.03213, 2017.

[30] Y. Sun, X. Wang, Z. Liu, J. Miller, A. Efros, and M. Hardt, “Test-time training with self-supervision for generalization under distribution shifts,” in International Conference on Machine Learning. PMLR, 2020, pp. 9229–9248.

[31] X. Mao, Q. Li, H. Xie, R. Y. Lau, Z. Wang, and S. Paul Smolley, “Least squares generative adversarial networks,” in Proceedings of the IEEE international conference on computer vision, 2017, pp. 2794–2802.

[32] J. Behley, M. Garbade, A. Milioio, J. Quenzel, S. Behnke, C. Stachniss, and J. Gall, “Semantickitti: A dataset for semantic scene understanding of lidar sequences,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 9297–9307.

[33] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, “nussenec: A multimodal dataset for autonomous driving,” in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 11621–11631.