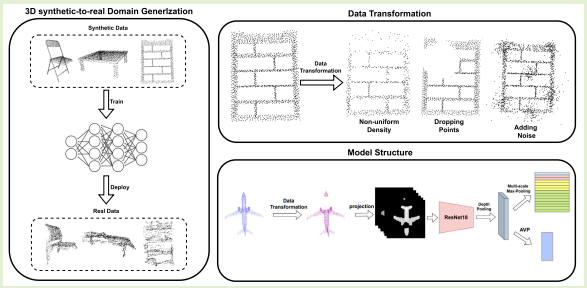
DG-MVP: 3-D Domain Generalization via Multiple Views of Point Clouds for Synthetic-to-RGBD Sensor Data Classification

Huantao Ren, Minmin Yang, and Senem Velipasalar, *Senior Member, IEEE*

***Abstract*—RGB-D camera sensors are used for capturing 3-D point cloud data for various applications, including autonomous vehicles, and object recognition and tracking. However, collecting large-scale datasets with RGB-D sensors and annotating the data is costly and labor-intensive. In contrast, sampling point clouds from computer-aided design (CAD) models provides a more efficient and cost-effective alternative. Yet, models trained on CAD-generated data often struggle to generalize to real-world RGB-D sensor data due to significant domain shifts. This challenge arises because point clouds sampled from CAD models are structured and accurately represent object shapes, whereas RGB-D sensor data frequently suffer from occlusions, missing points, nonuniform density, and noise, making it difficult to capture complete object geometry. Therefore, it is critical to develop methods that can generalize and perform well on real-world RGB-D sensor data, even when they are trained on synthetic data. Existing approaches for 3-D domain generalization (DG) employ point-based backbones to extract point cloud features. Yet, we have found that a large number of point features are discarded by point-based methods through the max-pooling operation. This is a significant waste, especially considering the fact that generalizing from synthetic domain to RGB-D sensor data domain is more challenging than supervised learning, and RGB-D sensor data are already affected by missing points and occlusion to begin with. To address these issues, we propose a novel method that can generalize to unseen real-world RGB-D sensor data. Our proposed 3-D point cloud DG method employs 2-D projections of a 3-D point cloud to alleviate the issue of missing points and involves a convolution-based model to extract features. To simulate the noisy points commonly found in real-world RGB-D sensor data, we propose a data transformation method that introduces both global and local noisy points. The experiments, performed on the PointDA-10 and Sim-to-Real benchmarks, demonstrate the effectiveness of our proposed method, which outperforms different baselines and can transfer well from the synthetic domain to the domain of real-world RGB-D sensor data.**



***Index Terms*—Depth images, domain generalization (DG), point cloud classification, RGB-D sensor, synthetic-to-real world.**

I. Introduction

With the rapid advancement and growing accessibility of 3-D sensors, such as RGB-D camera and LiDAR sensors, 3-D point cloud analysis has garnered significant interest from the research community. The RGB-D sensors, such as Microsoft Kinect[1] and Intel[1] RealSense[2], capture both color (RGB) and depth (D) information, providing richer spatial data compared to traditional RGB cameras, which only capture 2-D color images. These sensors rely on different depth-sensing principles. Early models of Kinect[1] and RealSense[2] employed stereo vision or structured light techniques, which estimate depth via parallax and may suffer from reduced accuracy in background regions or outdoor environments due to lighting interference. For instance, the original Kinect[1] becomes less reliable under strong sunlight, limiting its outdoor applicability. In contrast, some RealSense[2] models and later depth sensors have adopted time-of-flight (ToF) technology, which improves depth accuracy and robustness under varied lighting conditions [1]. LiDAR sensors, on the other hand, directly measure depth using laser pulses and offer high precision in outdoor settings. However, RGB information in LiDAR systems is often acquired separately, leading to potential misalignment between the color and depth channels, especially when the sensors are not intrinsically calibrated. This difference in sensing mechanisms and data quality

Received 11 July 2025; accepted 10 August 2025. Date of publication 9 September 2025; date of current version 16 October 2025. This work was supported in part by the U.S. National Science Foundation (NSF) under Award 2221875. The associate editor coordinating the review of this article and approving it for publication was Prof. Shih-Chia Huang. *(Corresponding author: Huantao Ren.)*

The authors are with the Department of Electrical Engineering and Computer Science, Syracuse University, Syracuse, NY 13244 USA (e-mail: hren11@syr.edu; myang47@syr.edu; svelipas@syr.edu).

Digital Object Identifier 10.1109/JSEN.2025.3605134

[1]Registered trademark.

[2]Trademark.

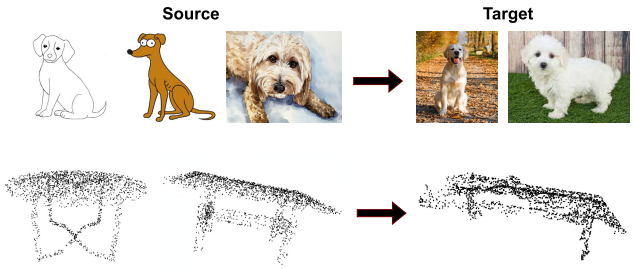


Fig. 1. Different domain shifts observed with 2-D images and 3-D point clouds.

across sensor types highlights the importance of selecting the appropriate sensor based on the application and environment. Today, RGB-D and LiDAR sensors play a crucial role in robotics, medical imaging, industrial automation, and AI-driven 3-D vision systems, shaping the future of intelligent perception.

Among different applications, our focus in this work is robust indoor 3-D object classification using data captured by RGB-D sensors while addressing the domain shift issue between synthetically generated 3-D point clouds and real-world data. In recent years, numerous deep neural network models [2], [3], [4], [5], [6] have been proposed for 3-D point cloud classification. Although significant advances have been achieved in point cloud classification, existing methods either rely on large-scale and well-annotated datasets and/or are trained and tested on the splits of the same dataset. However, for real-world applications, scanning objects with RGB-D or LiDAR sensors and then performing annotation to build a sufficiently large and representative dataset is labor-intensive and costly. Sampling point clouds from well-annotated computer-aided design (CAD) models, on the other hand, is relatively faster and easier. Thus, in practice, using a synthetic CAD-based dataset (e.g., ModelNet [7] and ShapeNet [8]) as the training set and evaluating the model on a real-world dataset collected with RGB-D Sensors (e.g., ScanNet [9]) is a more labor- and cost-effective way. Yet, this approach introduces a significant domain shift challenge since point clouds captured by real RGB-D camera sensors often suffer from occlusion and missing and/or noisy points, whereas point clouds sampled from CAD models are very regular. The significant divergence between the distributions of the synthetic and real point clouds causes the performance of many existing point cloud classification models to degrade significantly. Hence, to achieve better classification performance on RGB-D sensor data more efficiently, it is essential to develop an effective model to mitigate the domain shift issue between the synthetically generated and real-world sensor data.

In this article, we focus on addressing the challenges related to generalizing to real-world sensor data by developing a domain generalization (DG) approach. We train our proposed model by only using synthetic point cloud data (referred to as the source data), which can then achieve high classification performance on real-world RGB-D sensor data (referred to as the target data) *that is not accessible during training*. With 2-D images, 2-D DG methods have been extensively investigated [10], [11], [12], [13], [14]. Yet, the variations among 2-D samples often stem from different styles, such as photo style, cartoon style, and sketch. In contrast, the domain gap between 3-D point cloud samples primarily arises from geometric differences, such as missing points and background noise, as shown in Fig. 1. Thus, 2-D DG methods cannot be readily applied to 3-D sensor data.

Existing approaches for 3-D DG [15], [16], [17] use well-known point-based methods, such as DGCNN [6] or PointNet [2], as their backbones due to their simplicity and effectiveness in representing 3-D object shapes. PointNet [2] employs multilayer perceptron (MLP) and uses max-pooling operation at the end to extract permutation-invariant point features. DGCNN [6] aggregates neighbors' features of each point in each edge convolution layer and also performs max pooling at the end. In this work, we first show that point-based methods are not the most suitable models for 3-D DG for the following reasons.

1) Effectiveness of a point-based method in representing data is related to the number of points kept after max pooling. Our experiments show that models with better DG ability tend to use more points for the final prediction. Considering that the DG problem is already more challenging than supervised classification, it is even more important to make effective use of the available points. Many point-based methods employ max pooling, which often leads to the loss of potentially useful points and reduces the shape representation ability.
2) Point-based methods are highly sensitive to occlusions and missing points, which are common in RGB-D sensor data. When trained on synthetic data and tested on real-world RGB-D sensor data, the performance of these methods significantly degrades due to the simulation-to-reality gap.

To bridge the domain gap between synthetic CAD-generated data and RGB-D sensor data, we propose an effective 3-D DG network, referred to as the DG-MVP, using multiple views of point clouds for the classification of RGB-D sensor data. Instead of a point-based approach, our proposed DG-MVP employs multiple 2-D projections of a point cloud as input. We first project a 3-D point cloud onto six orthogonal planes, by using SimpleView [18], to obtain six depth images. Several example depth images obtained from different angles can be seen in Fig. 2. We employ Resnet18 [19] as the backbone to transform each input depth image into a 3-D feature map with the height, width, and channel dimensions. Then, depth pooling is used across all six depth images, outputting the most prominent features from each depth image. Next, to learn more local features, we propose a multiscale max-pooling (MMP) module, which horizontally divides the feature map into several parts and applies max pooling on each part to obtain the final representation. The effectiveness of our proposed method is analyzed through experiments, showing that it outperforms many point-based and multiview-based methods on DG of point cloud classification. To simulate missing points and changes in scanning density, which are often faced in real-world RGB-D point cloud data and prevent the model from overfitting to the synthetic source data, we also employ two point transformation approaches [15]

during training. Moreover, RGB-D sensors usually capture noise from surrounding objects in the real world, which leads to object deformation. To address this, we also propose adding both global and local noisy points to the original point cloud to improve the model's robustness against geometric variations. The experiments performed on two synthetic-to-real RGB-D sensor data benchmarks (PointDA-10 [20] and Sim-to-Real [15]) demonstrate that our DG-MVP consistently outperforms the state-of-the-art (SOTA) methods.

The main contributions of this work include the following.

1) We first analyze point utilization of the most commonly used point-based backbones and argue that they are not suitable for addressing the domain gap between the real-world RGB-D sensor data and synthetic data.
2) We visualize projected depth images of point clouds and observe that certain projections remain visually similar between CAD data and RGB-D sensor data, despite missing points and deformations. Motivated by these observations, we propose DG-MVP as an effective 3-D DG network using multiple 2-D projection views of point clouds for RGB-D sensor data classification.
3) We propose a multiscale max pooling module (MMP) to generate more local and descriptive features.
4) We propose a data transformation method to simulate noisy points and object deformations commonly encountered in real-world data collected by RGB-D sensors.
5) Our proposed DG-MVP outperforms different baselines on two synthetic-to-real RGB-D sensor data benchmarks (PointDA-10 and Sim-to-Real), even surpassing some 3-D domain adaptation methods *that utilize target RGB-D data during training*.
6) We perform ablation studies to show the effectiveness of the components of the DG-MVP.

To the best of our knowledge, this is the first work exploring the use of projection images for 3-D DG and demonstrating the limitations of many point-based backbones in this task. We hope that our findings encourage further research on leveraging depth images for addressing the domain shift between synthetic and real-world RGB-D sensor data.

II. Related Work

A. Point Cloud Classification

Based on input format, point cloud classification methods can be classified into three categories, namely, volumetric-, multiview-, and point-based methods. *Volumetric-based* methods, such as VoxNet [21], map unstructured 3-D point clouds into a set of voxels and then use 3-D convolutional neural networks (CNNs) to perform prediction. SEGCloud [22] first converts a point cloud into coarse voxels and then uses a 3-D fully convolutional network to make prediction.

Multiview-based methods usually render a group of 2-D images, by projecting 3-D points from different angles and then applying 2-D image processing models to make prediction. MVCNN [23] obtains 2-D projection images from 12 views and uses a CNN model to extract view-based features. GVCNN [3] leverages the relationship between multiple views (8 or 12) by grouping them based on their discrimination scores. Later on, SimpleView [18] showed that projecting a point cloud onto just six orthogonal planes and processing these projection images through ResNet [19] can be highly effective. More recently, Chen et al. [24] introduced ViewNet, which is a two-branch multiview-based backbone for few-shot 3-D classification.

Point-based methods directly take raw point clouds as input. PointNet [2], which is a pioneering work in this category, uses shared MLPs to extract point features and then applies max pooling to obtain permutation-invariant features. However, PointNet is unable to capture local features. To address this issue, PointNet++ [4] integrates hierarchical sampling, local feature learning from the neighborhood of each point, and multiscale aggregation. DGCNN [6] enhances point cloud processing by dynamically constructing graphs for each point cloud and using EdgeConv layers to capture local geometric features. PointCNN [5] performs traditional convolution on point clouds by transforming neighboring points into a canonical order. GDANet [25] introduces the geometry-disentangle module to dynamically separate point clouds into sharp and gentle regions of 3-D objects. These regions interact within each layer, allowing their features to enhance the overall point features. There are also some multimodal approaches proposed to fully describe a 3-D shape. FusionNet [26] integrates voxelized data and multiview images for 3-D object classification. CMFF [27] employs DGCNN and ResNet to fuse 3-D point cloud data and 2-D projection images for few-shot 3-D point cloud classification. POV [28] is another recent work utilizing both 2-D projection images and 3-D point clouds.

While these methods excel within a single domain, whether it be synthetic CAD-generated or real-world data captured by RGB-D sensors, their performance drops significantly in cross-domain scenarios.

B. Point Cloud Domain Adaptation and Generalization

Lately, unsupervised domain adaptation (UDA) for point clouds has attracted attention. This task typically assumes access to target domain data during training, allowing the model to adapt and improve generalization across domains. PointDAN [20] is a pioneering work in 3-D UDA, aligning local features between source and target data. DefRec + PCM [29] adopts self-supervised learning. It reconstructs partially distorted point clouds and employs mixup to align features across domains. GAST [30] enhances feature alignment by predicting the rotation angle of a mixed point cloud object and identifying the distorted part of a point cloud. GLRV [31] proposed two self-supervised methods, scale prediction and 3-D-2-D-3-D projection reconstruction, to align the distribution of source and target features.

Compared to 3-D UDA, 3-D DG is a more realistic, yet more challenging task, since target data are inaccessible during training. Metasets [15] designs three point augmentation tasks to simulate RGB-D camera-collected data and proposes to meta-learn representations from a group of augmented point clouds. PDG [17] learns generalizable part-level features with the augmented point clouds as in [15]. SUG [16] first splits

one source domain into different subdomains and proposes a multigrained subdomain alignment method to learn the domain-agnostic features. Push-and-pull [32] improves the generalization performance by designing a learnable 3-D data augmentor to push the augmented samples away from the originals, and a representation regularization objective to pull the predictions back to their original space. The aforementioned works mainly use DGCNN [6] or PointNet [2] to extract point features, which are both point-based and not the most suitable backbones for the synthetic-to-real RGB-D data DG task. We further explain these limitations in Section III.

To the best of our knowledge, our proposed work is the first to explore multiview images for 3-D DG. Unlike existing multiview methods developed for domain-specific settings, such as ViewNet [24] and CMFF [27], our method is designed with the aim of DG robustness. ViewNet [24] employs a two-branch architecture: one branch extracts view-specific features, while the other captures holistic information using a view pooling module that groups and aggregates features across predefined view combinations. While this structure can model complex dependencies, it may lead to overfitting to the source domain, especially in the presence of large domain gaps. In contrast, our method uses a single-branch architecture to reduce the overfitting risk. CMFF [27], a multimodal approach, fuses 3-D and 2-D features using separate point- and image-based backbones. Its 2-D branch modifies ResNet to accept six projected depth maps as input channels. This design is sensitive to view ordering and alignment, which are typically consistent in synthetic data but vary significantly in real-world RGB-D scans. In contrast, we process each projected image independently using a single-channel ResNet18 and perform view aggregation via max pooling, making our approach more robust to inconsistent view orders in the target domain.

III. Motivation

A. Point Utilization Analysis

Many point-based methods employ max pooling to obtain permutation-invariant features, which causes only a portion of points' features to be used while discarding the rest. It was shown in [33] that these discarded points are actually useful for supervised learning. In this section, we investigate the number of points utilized after max pooling, for the DG task on PointDA-10 [20], for three point-based methods, namely, PointNet++ [4], DGCNN [6], and GDANet [25]. PointDA-10 contains three domains: ModelNet [7], ShapeNet [8], and ScanNet [9]. Each subset contains the same ten classes. ModelNet and ShapeNet are synthetic datasets with points sampled from CAD models, while ScanNet is collected by an RGB-D camera sensor. We conduct experiments in two synthetic-to-real scenarios: ModelNet to ScanNet and ShapeNet to ScanNet. Each input point cloud contains 1024 points. The classification accuracy, shown in Fig. 3, represents the average values across both settings. It can be seen that the prediction accuracy is positively correlated with the number of points kept after max pooling, i.e., point utilization. This indicates that models with stronger generalization ability tend to utilize a greater number of points. For example, GDANet achieves the highest accuracy among the three methods and also has the highest point utilization, with around 221 points, compared to 95 for PointNet++ and 151 for DGCNN. However, even for the best-performing method, GDANet, fewer than 25% of the points are used in the final prediction, meaning many potentially informative points are discarded. Moreover, neighbor-aggregation modules, such as EdgeConv in DGCNN and point grouping in PointNet++, may further hinder generalization. These modules are designed to aggregate features from local neighborhoods and have shown effectiveness in domain-specific settings. However, in cross-domain scenarios, the neighboring points in the source and target domains often exhibit significantly different local structures, especially since point clouds in real-world domains commonly suffer from missing points and shape distortions.

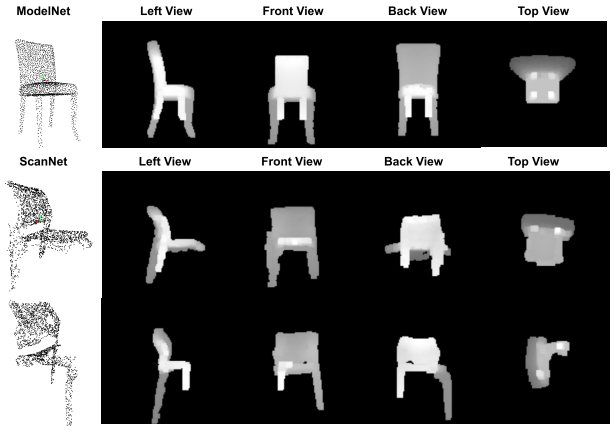


Fig. 2. Example point clouds for chairs and their corresponding depth images for ModelNet and ScanNet datasets.

To address this problem, the discarded points can be reused to enhance point utilization, and more robust local feature-extraction modules can be designed to better capture object geometry. Alternatively, the point-based backbone, such as DGCNN, can be replaced to generate more representative features. Thus, motivated by this and different from previous works, such as SUG [16] and PDG [17], we replace the point-based backbone with an alternative backbone.

B. Multiview Depth Image Analysis

As discussed above, when point clouds sampled from CAD models are used as the source domain, and those captured by real-world RGB-D sensors are used as the target domain, a significant domain shift occurs. Since point-based methods take raw 3-D points as input directly, this negatively affects models' generalization ability when dealing with missing points and shape distortions, which are frequently encountered in the RGB-D sensor-collected target domain. On the other hand, when point clouds are projected into depth images from different angles, certain depth images tend to be more robust against missing points, as shown in Fig. 2, where the first row and last two rows show example point clouds and their corresponding depth images from ModelNet (source) and ScanNet (target) datasets, respectively. In ScanNet, the first chair is missing its front legs, while the second chair is missing the left part of the seat and the right back leg. Despite

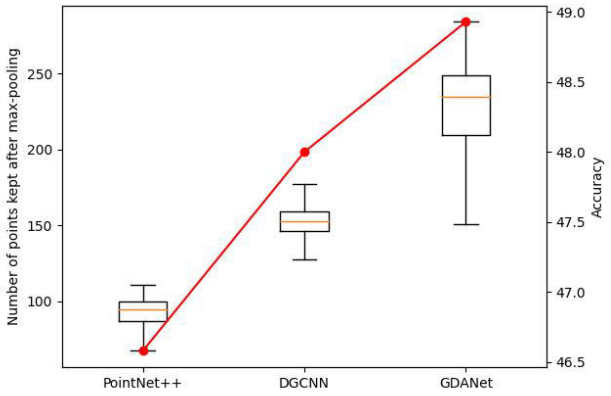


Fig. 3. Classification accuracy versus the number of points retained after max pooling for PointNet++, DGCNN, and GDANet on PointDA-10.

TABLE I
COMPARISON OF DIFFERENT 3-D BACKBONE ARCHITECTURES

Method	Input	$M \rightarrow S^*$	$S \rightarrow S^*$	Avg
PointNet++ [4]	Point	49.92	43.24	46.58
DGCNN [6]	Point	50.08	45.92	48.00
PointCNN [5]	Point	50.54	45.51	48.03
GDANet [25]	Point	49.8	48.05	48.93
ViewNet [24]	Multi-View	42.85	40.42	41.06
SimpleView [18]	Multi-View	49.29	45.22	47.26
CMFF [27]	Multi-View + Point	**50.76**	**48.33**	**49.58**

varying locations of the missing points, some projections, e.g., the front and top views of the first ScanNet chair, and the left, front, and back views of the second ScanNet chair, still resemble those of the CAD-based point cloud, providing robustness to the issue. Thus, employing multiple depth images holds more promise for synthetic-to-real RGB-D data DG task compared to using only raw point clouds.

We also compared the performance of some point- and multimodality-based methods (not designed for DG) on the PointDA-10 dataset and present the results in Tab. I, where $M \rightarrow S^*$ and $S \rightarrow S^*$ indicate that the model is trained on synthetic datasets, ModelNet (M) and ShapeNet (S), respectively, and tested on ScanNet (S^*), collected using a real RGB-D sensor. As can be seen, CMFF [27], which employs DGCNN and ResNet to extract features from both 3-D points and multiple depth images, respectively, has better generalization ability compared to point-based methods. This indicates that multiple depth images provide valuable information and robustness for DG. However, having two separate branches for two different modalities is computationally expensive, and current multiview-based methods generally do not perform as well as the point-based methods on the DG task. Hence, these observations provide the motivation to develop a method that does not rely on point-based backbones (which use max pooling) and instead exploits the benefits of using multiple depth images for 3-D DG.

IV. PROPOSED METHOD

A. Problem Definition

In DG on point clouds, given a source domain $S = \{x_i, y_i\}_{i=1}^{n_s}$, with n_s-many labeled synthetic point clouds, our goal is to train a model f on the synthetic source domain, which can generalize to an unseen real-world target domain $T = \{x_i, y_i\}_{i=1}^{n_t}$, where x_i and y_i denote the ith point cloud sample and its label in the respective dataset, respectively. It should be noted that only the data and labels from the source domain are accessible during training, while data and labels from the target domain are only used for evaluation.

B. Data Transformation

Considering the irregular distribution of points in RGB-D sensor-collected data (due to, e.g., missing points and noise), we apply three different sets of transformations to the synthetic point clouds to enhance the model's generalization ability and better simulate real-world data variations. In addition to applying the transformation techniques proposed in [15] to simulate missing points and different densities, we propose *a new data transformation method* to simulate noisy points at *both global and local levels*, as described below.

Due to sensor limitations, occlusions, and surface properties that hinder accurate depth measurement, RGB-D sensor-collected data often contain missing points. Factors, such as reflective or transparent surfaces, depth range constraints, and viewpoint dependency, further contribute to incomplete point cloud. Therefore, the first set of transformations aims to simulate missing points and occlusion [15]. Let $P \in \mathbb{R}^{M\times 3}$ denote an original normalized point set containing 3-D coordinates of M points. We randomly select a point p_i from P and drop its nearest neighbors using various drop rates (0.24, 0.36, and 0.45).

Moreover, the density of RGB-D sensor-collected point clouds is nonuniform due to the perspective projection of depth sensors. Points closer to the camera appear denser, while those farther away become more sparse due to the depth resolution being inversely related to distance. To simulate nonuniform densities, Huang et al. [15] proposed a method that removes points based on their distance from a randomly selected point on a unit sphere, using a probabilistic approach controlled by a parameter g. We employ this approach as the second set of transformations and set g to 1.3, 1.4, and 1.6. Examples showing the original and transformed point clouds (with missing points and nonuniform density) are presented in Fig. 4.

In addition to missing points and nonuniform density, point clouds captured by RGB-D sensors often suffer from noise. For instance, when an RGB-D camera scans objects, it inevitably captures noise from the background or surrounding objects. Moreover, depth measurement errors and motion artifacts can further introduce noisy points, affecting the quality of the captured point clouds. To address this, we propose a new, third set of data transformation method to simulate noisy points at both *global* and *local* levels, as described next.

Global Noise: To introduce global noise, we first normalize a point cloud to the $[-1, 1]$ range. Then, we uniformly sample random points within a unit cube and append them to the normalized point cloud, resulting in the noise-corrupted data. Each (x-, y-, z)-coordinate is drawn from a uniform distribution in the $[-1,1]$ range. This results in a global effect on the overall data as seen in the first column of Fig. 5.

Local Noise: To simulate shape deformations and noise (e.g., the table surface in the target domain in Fig. 1) caused by RGB-D sensors, we introduce random clusters of noisy data around local points. The total number of noisy points to be added is predefined. We randomly select N points from the point cloud as the cluster centers. For each cluster, Gaussian noise, with a random standard deviation between 0.075 and 0.125, is added to simulate typical sensor noise. This range is chosen to ensure that the added noise remains localized and realistic—values lower than 0.075 result in overly dense clusters, while values above 0.125 produce overly sparse points, with noise resembling global perturbations when the standard deviation exceeds 0.2. Any points falling outside the unit cube are rescaled back within the boundaries. The noisy points are then appended to the point cloud, resulting in a localized effect as seen in the second column of Fig. 5.

Both global and local noises are applied together to simulate RGB-D sensor data, with the number of added noisy points determined by a predefined severity level l. Higher severity levels introduce more noise points, resulting in a less distinct object outline. In our work, we set l to 1, 4, or 8, corresponding to the addition of 100, 400, and 800 noisy points, respectively. Examples of transformed point clouds with varying severity levels are shown in the last column of Fig. 5.

Each point cloud is assigned one of the ten possible states: keeping the original cloud or applying one of the nine possible random transformation choices described above (i.e., dropping points (DPs) with three different rates, density adjustment with three different rates, and adding noise with three different severity levels). Among the samples, 25% of them are kept in their original form, 25% undergo point dropping, another 25% have their density adjusted, and the remaining 25% are subjected to noise addition. After obtaining the transformed point cloud $P' \in \mathbb{R}^{M' \times 3}$, following [18], we adopt a projection-based rendering approach using six predefined canonical views: front, back, left, right, top, and bottom. Each view is defined by a unique combination of Euler angle rotations and a fixed translation to simulate a virtual camera's position and orientation. The point cloud is transformed accordingly and then projected onto a 2-D depth map using a perspective projection model. During projection, each point's depth is scattered to neighboring pixels with inverse depth weighting to reduce sparsity. A morphological dilation is further applied to smooth the resulting depth maps. The transformed point cloud P' is ultimately rendered into multiview depth images $F_1 \in \mathbb{R}^{6\times 1\times H\times W}$, where 6, 1, H, and W represent the number of views, number of channels, and the height and width of each depth image, respectively, and both H and W are 128. It is worth noting that this projection process does not require any additional learnable parameters or external tools, making it lightweight and easy to integrate. Examples of projection images are shown in Fig. 2.

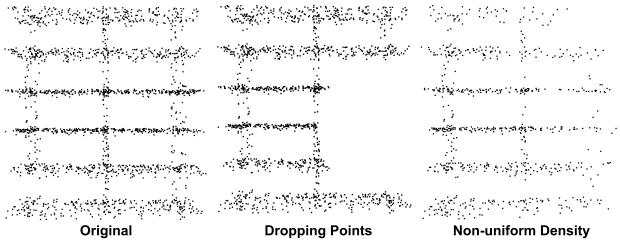


Fig. 4. Examples of transformed point clouds via DPs and nonuniform point density.

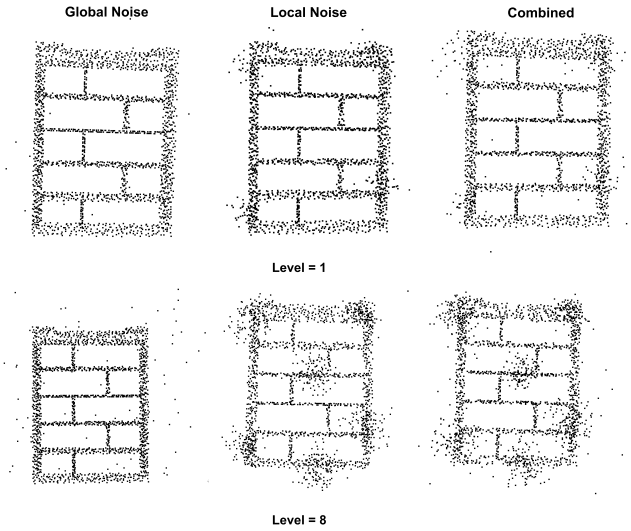


Fig. 5. Examples of transformed point clouds with global noise, local noise, and a combination of both, applied at different levels of severity.

C. Network Architecture

Based on the insights gained from the analysis in Section III, and to address the issues encountered in real-world data captured by RGB-D sensors, we propose an effective DG network for RGB-D point cloud classification, referred to as the DG-MVP, which uses multiple views of point clouds. We employ multiple projected 2-D depth images of a point cloud as input, and adopt ResNet18 as the backbone to extract 2-D features. To preserve more global and local features, a multiscale MMP is proposed and integrated at the end of ResNet18, enhancing the network's ability to capture more representative information. The overall architecture of our proposed DG-MVP is presented in Fig. 6.

After data transformation and projecting the point cloud into 2-D depth images, we use ResNet18 to obtain a feature map for each depth image. The output feature map is denoted as $F_2 \in \mathbb{R}^{6\times C\times H\times W}$, where C represents the number of channels, which is 256 in our work. To obtain a holistic understanding of the entire object and by drawing inspiration from GaitBase [34], which is a gait recognition network, we use element-wise maximum operation across six views, referred to as the Depth Pooling.

Alternatives to depth pooling are average pooling and view pooling [24]. Average pooling computes the mean value across all views. View pooling groups the views into pairs based on visual similarity (left–right, top–bottom, and front–back) and triplets (top–front–left and bottom–back–right) to provide a comprehensive representation of an object's overall shape. Max pooling is then applied within each group along the dimension of the number of projections. These pooled features are then concatenated and passed through a convolutional

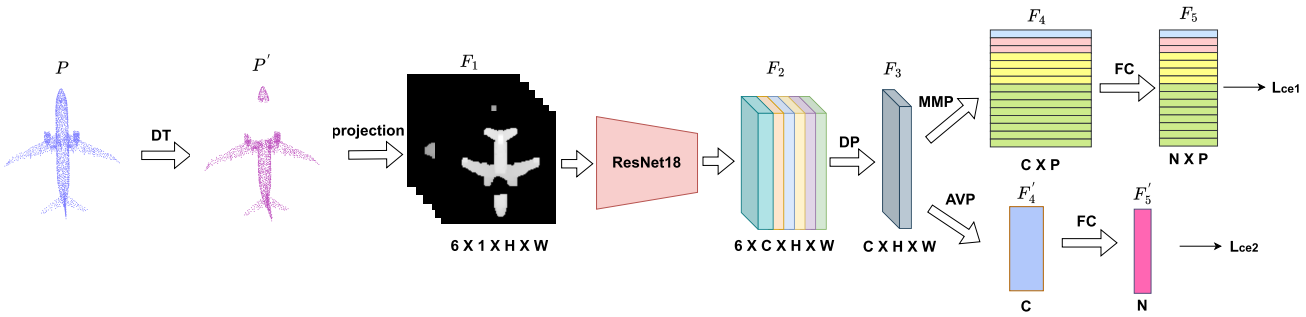


Fig. 6. Pipeline of DG-MVP. DT, DP, MMP, AVP, and FC represent data transformation, depth pooling, multiscale max pooling, average pooling, and FC layer, respectively. *H*, *W*, *C*, *P*, and *N* denote height, weight, the number of channels, the total number of strips, and the number of classes, respectively.

layer. This design was proven effective for few-shot learning in a domain-specific setting. However, these two pooling methods do not perform as well as depth pooling for DG. Average pooling can result in the loss of detailed information, leading to an inaccurate representation. Moreover, since synthetic CAD models are manually designed and stored in a canonical coordinate system, ModelNet and ShapeNet datasets are axis-aligned. In contrast, ScanNet is not axis-aligned since RGB-D sensors scan objects and environments without a predefined alignment, resulting in arbitrary orientations. For instance, in ModelNet, a chair might always be oriented so that its legs are parallel to the z-axis, and its seat is parallel to the xy plane. In ScanNet, a similar chair might appear tilted, rotated, or at varying angles. As a result, different views of the same object in the target domain are inconsistent. As a result, the same object may be viewed from very different angles, and the visual similarity between canonical views (e.g., front and back) may no longer hold. For example, in the target domain, the front and back views of a chair may present completely different appearances, undermining the assumptions behind the fixed grouping strategy used in view pooling. Moreover, the convolutional layer used to compress the grouped features in view pooling can lead to overfitting to the source domain. While it is effective at capturing complex source-domain patterns, it may not generalize well when the view distribution and object appearances shift significantly in the target domain. Although randomly rotating the point cloud may help, we adopt a simple, yet effective, depth pooling strategy in cross-domain settings, formulated as $F_3 = \text{maxP}(F_2, \text{dim} = 0)$, instead of average pooling and view pooling. The supporting experimental results, in Section VI-B under ablation studies, show that depth pooling yields superior results.

After obtaining $F_3 \in \mathbb{R}^{C\times H\times W}$, the model splits into two branches. The bottom branch in Fig. 6 follows the original ResNet architecture, consisting of an average pooling layer and a fully connected (FC) layer. The top branch, on the other hand, includes our proposed MMP module to capture additional global and local features, followed by an FC layer for prediction. Finally, cross-entropy loss is used in each branch to obtain the losses L_{ce1} and L_{ce2}. The overall loss function is

$$L = \lambda_1 L_{\text{ce1}} + \lambda_2 L_{\text{ce2}} \tag{1}$$

where λ_1 and λ_2 are hyperparameters to balance the two loss components. In this work, both λ_1 and λ_2 are set to 0.5.

D. Multiscale Max Pooling

To capture both broad and fine-grained features, we design a modified multiscale max pooling (MMP) approach, which was originally presented in [35] for person re-identification. The structure of MMP is shown in Fig. 7. The input is F_3, which is the output of the depth pooling layer. MMP employs different scales: $n_1 = 1$, $n_2 = 2$, $n_3 = 4$, and $n_4 = 8$. The input feature map is split into n_i strips along the height dimension, where each strip contains $((h/n_i) \times w)$-many pixels. Then, global max pooling is performed on each strip to get 1-D features. Next, an FC layer and batch normalization are applied to each strip to map the features into a discriminative space. It is important to note that each strip has its own FC layer, as strips from different scales represent features with varying receptive fields. Finally, all strips are concatenated to obtain $n = \sum_{i=1}^{4} n_i$ strips in total.

Different from [35], we apply separate FC layers to each individual strip, instead of using a single FC layer after concatenating all distinct strips. This design allows each layer to focus on a specific region, enabling the model to learn localized features more effectively from different receptive fields. Moreover, we apply only max pooling to each strip, rather than combining max pooling and average pooling, since empirical results consistently show that average pooling has little impact.

V. Experiments

A. Dataset and Implementation Details

We evaluate the generalization performance of our proposed DG-MVP on the PointDA-10 [20] and Sim-to-Real [15] benchmarks. In both datasets, the source domains originate from ModelNet and ShapeNet, where point clouds are uniformly sampled from synthetically generated 3-D CAD models. The target domains for PointDA-10 and Simt-to-Real benchmarks are ScanNet [9] and ScanObjectNN [36] datasets, respectively. Both datasets consist of indoor scene data collected using RGB-D sensors. ScanNet is composed of real-world data scanned by a structure sensor, a commodity RGB-D sensor with design similar to the Microsoft Kinect[1] v1. ScanobjectNN is a real-world indoor point cloud dataset

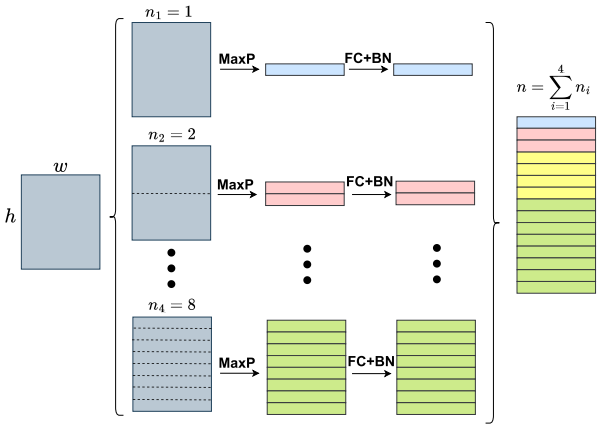


Fig. 7. Structure of multiscale max pooling. h, w, and n_i represent the height and weight of the feature map, and the number of strips, respectively. Max$P$, FC, and BN denote max pooling, FC layer, and batch normalization, respectively.

constructed from two scene meshes datasets, ScanNet and SceneNN [37]. SceneNN is an indoor scene dataset collected using two RGB-D sensors, namely, the Asus[1] Xtion PRO and Microsoft Kinect[1] v2. Unlike synthetic datasets, point clouds collected by these RGB-D sensors usually contain missing points, are not axis-aligned, and do not often represent objects fully.

PointDA-10 [20] is composed of three datasets with the same ten categories: 1) ModelNet (M) contains 4183 training and 856 test samples; 2) ShapeNet (S) contains 17 378 training and 2492 test samples; and 3) ScanNet ($S*$) contains 6110 training and 2048 test samples. Different from previous works [16], [32], we conduct experiments solely on the most challenging and realistic scenario, by using synthetic data (ModelNet and ShapeNet) as the source domain and the real-world data (ScanNet) as the target domain. For a commensurate comparison with previous works, we follow the same data preparation and experiment settings as in [17] and [30]. Specifically, each point cloud is down-sampled to 1024 points and randomly rotated along the z-axis, as described in [29], for training.

Sim-to-Real [15] is a 3-D DG dataset consisting of three datasets: ModelNet (M), ShapeNet (S), and ScanObjectNN (SO) [36]. ScanobjectNN contains different deformation variants, such as background occurrence (SO_B). Following [15], we consider four synthetic-to-real settings: $M \rightarrow \text{SO}$ and $M \rightarrow \text{SO}_B$ refer to training on ModelNet and evaluating on ScanobjectNN and ScanObjectNN with background, respectively, with 11 shared classes; $S \rightarrow \text{SO}$ and $S \rightarrow \text{SO}_B$ represent ShapeNet as the source domain, with ScanObjectNN and ScanObjectNN with background as the target domains, sharing 9 classes. We train the models for 200 epochs on PointDA-10 and 160 epochs on Sim-to-Real, with a batch size of 32, using one Quadro RTX 6000 GPU. The Adam optimizer is used with a learning rate of 0.001 and a weight decay of 0.00005. To reduce the randomness in selecting data transformation methods and ensure every transformation method is applied a similar number of times, we maintain a fixed application frequency for each transformation within a batch. Specifically, DPss, nonuniform density transformation, and noisy point addition are each applied to 25% of the samples, and the remaining 25% of the samples are kept as original. All experiments are repeated three times, and the average accuracy is reported.

B. Results on PointDA-10

We compare our proposed DG-MVP with the SOTA 3-D DG methods, including the meta-learning approach MetaSets [15], the part-level feature alignment method PDG [17], the adversarial training method Push-and-Pull [32], and the domain alignment method SUG [16]. We also compare with the SOTA 3-D UDA methods, which *utilize target domain data during training*, including PointDAN [20], RS [38], DefRec + PCM [29], GAST [30], GLRV [31], and PC-Adapter [39].

The results are summarized in Table II, where "DG-MVP w/o DT" represents our results obtained without applying any data transformation, "DG-MVP (DP+NUD)" refers to only applying transformations of DPs and nonuniform density (NUD) to the original point cloud, while "DG-MVP (DP+NUD+N)" utilizes all data transformation methods, namely, DPs, simulating nonuniform density, and adding noise (N). To provide a holistic view of generalization performance, we also report the average accuracy across the $M \rightarrow S^*$ and $S \rightarrow S^*$ settings. While each setting captures different domain shifts, the average helps summarize a model's overall robustness when transferring from synthetic to real-world domains. We can see that our DG-MVP achieves an average accuracy of 56.19% in the synthetic-to-real RGB-D scenario, surpassing all other 3-D DG methods. It outperforms the second-best model, SUG, by 1.49% in average accuracy and by 4.44% in the $S \rightarrow S^*$ scenario. It is also notable that our method even outperforms more than half of the 3-D UDA methods, namely, PointDAN, RS, DefRec+PCM, and PC-Adapter, without requiring access to target data during training. The results also show that adding noisy points further improves the results (compared to "DG-MVP (DP+NUD)") by 0.87% in $M \rightarrow S^*$ and 2.43% in $S \rightarrow S^*$, demonstrating that introducing noisy points into synthetic point clouds effectively contributes to the simulation of real-world RGB-D sensor-captured point cloud characteristics for DG.

All the baselines in Table II are point-based and all but one employ DGCNN as their backbone. Thus, we also utilize t-SNE and visualize the feature distribution on the target RGB-D data domain for DGCNN, and our proposed DG-MVP without data transformation (DG-MVP w/o DT) in Fig. 8. It can be observed that the feature distribution of our method is better separated and more distinct compared to DGCNN for both $M \rightarrow S^*$ and $S \rightarrow S^*$ scenarios. This further demonstrates that our DG-MVP can extract better domain-invariant and distinguishing features.

C. Results on Sim-to-Real

We conduct experiments on four synthetic-to-real RGB-D scenarios on this dataset, namely, $M \rightarrow \text{SO}$, $M \rightarrow \text{SO}_B$, $S \rightarrow \text{SO}$, and $S \rightarrow \text{SO}_B$. Since the dataset released in [15] did

TABLE II
CLASSIFICATION ACCURACY (IN %) OF VARIOUS 3-D UDA AND 3-D DG METHODS ON THE POINTDA-10 DATASET

Method	UDA/DG	$M \rightarrow S^*$	$S \rightarrow S^*$	Avg
RS [38]	UDA	44.8	45.7	45.25
PointDAN [20]		45.3	46.9	46.10
DefRec+PCM [29]		51.8	54.5	53.15
PC-Adapter [39]		58.2	53.7	55.95
GAST [30]		59.8	56.7	58.25
GLRV [31]		60.4	57.7	59.05
MetaSets [15]	DG	52.3	42.1	47.20
Push-and-Pull [32]		55.3	47.0	51.15
PDG [17]		**57.9**	50.0	53.95
SUG [16]		57.2	52.2	54.70
DG-MVP w/o DT (ours)		50.31	51.03	52.38
DG-MVP (DP+NUD) (ours)		54.87	54.21	54.54
DG-MVP (DP+NUD+N) (ours)		55.74	**56.64**	**56.19**

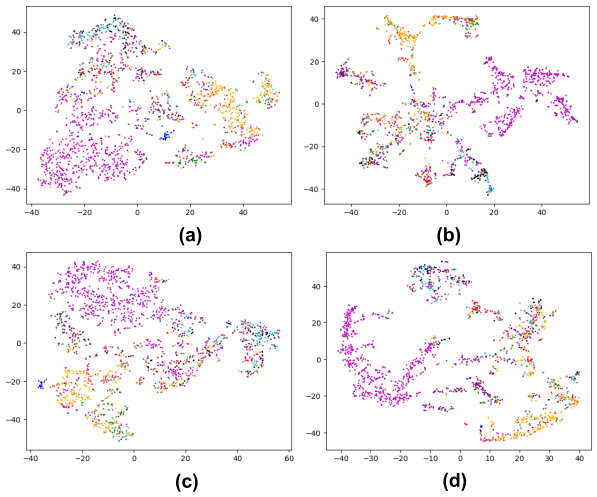


Fig. 8. t-SNE visualization of feature distribution of the target domain samples on PointDA-10. (a) DGCNN: M→S^*. (b) DG-MVP w/o DT: M→S^*. (c) DGCNN: S→S^*. (d) DG-MVP w/o DT: S→S^*.

not include ScanObjectNN with backgrounds, we collected these samples with backgrounds directly from the original ScanObjectNN website. All reported results were obtained by running the published code[3,4] using the same data and preprocessing pipeline. We compare with SOTA DG methods, namely, Metasets and PDG, which report their results on this dataset. We also compare with point-based backbones used by the baselines in Table II, namely, PointNet and DGCNN, for reference. The results for overall accuracy are summarized in Tables III and IV when M and S are the source domains, respectively. In Table III, we can see that our method significantly outperforms Metasets and PDG for both $M\rightarrow$ SO and $M\rightarrow$ SO$_B$. On average, it surpasses Metasets and PDG by 7.51% and 4.52%, respectively. Since Metasets and PDG do not employ noise addition, even without incorporating this step, DG-MVP (PD+NUD) still outperforms Metasets and PDG by 7.12% and 4.13%, respectively. In addition, to compare with PDG, we show the confusion matrices of class-wise classification accuracy in Fig. 9. DG-MVP outperforms PDG by 6.83% and 2.78% in terms of average class accuracy (ACA) in $M\rightarrow$SO and $M\rightarrow$SO$_B$, respectively.

[3]https://github.com/thuml/Metasets/tree/main
[4]https://github.com/weixmath/PDG?tab=readme-ov-file

TABLE III
ACCURACY ON MODELNET (M) TO SCANOBJECTNN W/O BACKGROUND (SO) AND SCANOBJECTNN W/ BACKGROUND (SO$_B$) BENCHMARK

Method	$M \rightarrow SO$	$M \rightarrow SO_B$	Avg
PointNet [2]	58.68	52.68	55.68
DGCNN [6]	61.37	53.26	57.32
Metasets [15]	60.30	62.53	61.41
PDG [17]	64.81	63.99	64.40
DG-MVP w/o DT (ours)	64.63	60.79	62.71
DG-MVP (DP+NUD) (ours)	70.68	66.37	68.53
DG-MVP (DP+NUD+N) (ours)	**71.42**	**66.42**	**68.92**

TABLE IV
ACCURACY ON SHAPENET (S) TO SCANOBJECTNN W/O BACKGROUND (SO) AND SCANOBJECTNN W/ BACKGROUND (SO$_B$) BENCHMARK

Method	$S \rightarrow SO$	$S \rightarrow SO_B$	Avg
PointNet [2]	54.63	49.25	51.94
DGCNN [6]	54.06	51.56	52.81
Metasets [15]	52.75	54.5	53.63
PDG [17]	58.93	**56.91**	57.92
DG-MVP w/o DT (ours)	56.06	52.75	54.41
DG-MVP (PD+NUD) (ours)	59.88	56.31	58.10
DG-MVP (PD+NUD+N) (ours)	**60.81**	56.75	**58.78**

TABLE V
ACA AND OVERALL ACCURACY, WITH AND WITHOUT THE MMP FOR $M\rightarrow S^*$ AND $S\rightarrow S^*$

	$M \rightarrow S^*$ OA (ACA)	$S \rightarrow S^*$ OA (ACA)
DG-MVP w/o MMP	49.07 (45.00)	47.88 (34.04)
DG-MVP w MMP	**50.31 (45.17)**	**51.61 (37.20)**

In Table IV, our method outperforms Metasets, on average accuracy, with and without data transformation. DG-MVP outperforms PDG by 1.88% and 0.86% in the $S\rightarrow$ SO setting and average accuracy, respectively, and achieves comparable results in the $S\rightarrow$ SO$_B$ setting. Moreover, our DG-MVP surpasses, even without any data transformation, the most commonly used point-based backbone, namely, DGCNN, by 5.39% and 1.6% in ModelNet to ScanObjectNN and ShapeNet to ScanObjectNN, respectively. These results further validate the effectiveness of our proposed method.

VI. ABLATION STUDIES

We have conducted ablation studies on the PointDA-10 dataset to study the impact of different components of our proposed method. In the following, except Section VI-D, each component is evaluated *without applying data transformation*.

A. Impact of MMP

As explained above, we designed an MMP module to aggregate global and local features. Table V shows the accuracy of DG-MVP with and without the MMP module, where the numbers inside and outside brackets represent the ACA and overall accuracy (OA), respectively. As can be seen, with MMP, all accuracy values are increased for both $M\rightarrow S^*$ and $S\rightarrow S^*$ settings. For $S\rightarrow S^*$ setting, the OA and ACA increase as much as 3.73% and 3.16%, respectively.

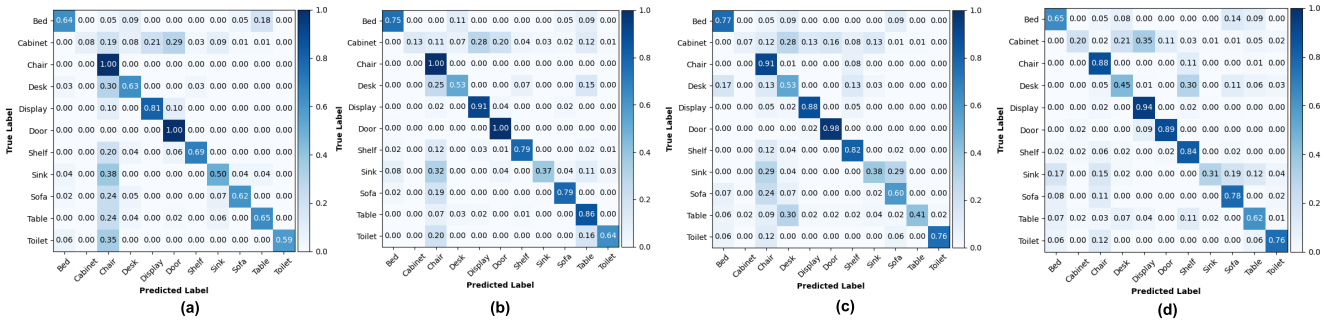


Fig. 9. Confusion matrices for class-wise classification accuracy for PDG and our DG-MVP on ModelNet (M) → ScanObjectNN w/o background (SO) and ModelNet (M) → ScanObjectNN with background (SO_B) settings. (a) PDG: M→SO. (b) DG-MVP: M→SO. (c) PDG: M→SO_B. (d) DG-MVP: M→SO_B.

TABLE VI
COMPARISON OF DEPTH POOLING, AVERAGE POOLING, AND VIEW POOLING

	$M \rightarrow S^*$ OA (ACA)	$S \rightarrow S^*$ OA (ACA)
Average pooling	43.75 (43.71)	48.84 (36.2)
View Pooling	45.56 (42.24)	47.48 (33.57)
Depth Pooling	**50.31 (45.17)**	**51.61 (37.2)**

TABLE VII
EFFECT OF BACKBONE DEPTH IN DG

Backbone	$M \rightarrow S^*$ OA (ACA)	$S \rightarrow S^*$ OA (ACA)
ResNet9	47.97 (43.64)	50.05 (37.1)
ResNet18	**50.31 (45.17)**	**51.61 (37.2)**
ResNet50	47.31 (44.94)	51.29 (36.87)

B. Impact of Depth Pooling

We analyze the role of depth pooling by replacing it with average pooling and view pooling and comparing the performances. The results summarized in Table VI show that depth pooling significantly outperforms the other two pooling methods in terms of both ACA and overall accuracy. While view pooling captures the relationships between different view angles, it is not effective for DG since real-world data collected by RGB-D sensors are not axis-aligned.

C. Impact of Backbone

To explore the impact of backbone depth and complexity on the generalization ability, we replace ResNet18 with ResNet9 and ResNet50. The results in Table VII show that ResNet50 does not improve the generalization ability. On the other hand, if the backbone is too shallow, it fails to learn representative features. ResNet18 performs best in both the $M \rightarrow S^*$ and $S \rightarrow S^*$ scenarios, leading us to employ it as our backbone.

D. Impact of Data Transformation

We evaluate the impact of different data transformation methods on performance by testing each method—DP, nonuniform density (NUD), and noisy point addition (N)—individually as well as a combination of all three. Each transformation is applied alongside the "keeping original" approach, meaning some samples undergo transformations, while others are kept in their original form. This ensures that the model still has sufficient exposure to complete object shapes. For example, when applying a single type of transformation, half of the samples are transformed, and the other half remain unchanged. When combining all three transformation methods, 75% of the samples are transformed, while the remaining 25% is kept as original. The results in Table VIII show that each transformation method contributes differently to DG performance. For instance, using only NUD achieves better overall accuracy compared to using only DP or N. However, using only NUD results in the lowest ACA, indicating that the performance for certain classes is weaker compared to other transformations. When all three methods are combined, the model achieves the best performance in both overall accuracy and ACA, demonstrating that each transformation method uniquely enhances DG.

TABLE VIII
EFFECT OF DIFFERENT DATA TRANSFORMATION METHODS IN DG

Dropping Points (DP)	Non-uniform Density (NUD)	Adding Noise (N)	$M \rightarrow S^*$ OA (ACA)	$S \rightarrow S^*$ OA (ACA)	Avg OA (ACA)
✓			50.82 (49.26)	53.25 (38.40)	52.04 (43.83)
	✓		53.25 (48.08)	53.99 (34.44)	53.62 (41.26)
		✓	51.38 (46.91)	52.8 (36.54)	52.09 (41.73)
✓	✓	✓	**55.74 (50.07)**	**56.64 (39.69)**	**56.19 (44.88)**

E. Impact of the Number of Views

The results of using different numbers of views to obtain depth images to train our model are summarized in Table IX. In the case of six views, we use the six orthogonal views. For eight views, we select the projections from the vertices of the cube centered on the object. For 14 clock views, we select the first 12 views around the object every 30°. The remaining two views are the top and bottom views. For 14 cube views, we combine the six orthogonal views and eight views from the vertices. The results illustrate that using six orthogonal views provides the best average accuracy. While 14 cube views in the $M \rightarrow S^*$ setting and 14 clock views in the $S \rightarrow S^*$ setting perform slightly better for the overall accuracy (by 0.68% and 0.17%, respectively), they do not improve class average accuracy and require *significantly more* computing power. Using six views strikes a balance by conserving computational resources while effectively capturing the object's features.

TABLE IX
COMPARISON OF USING A DIFFERENT NUMBER OF VIEWS

No. of views	$M \rightarrow S^*$ OA (ACA)	$S \rightarrow S^*$ OA (ACA)	Avg OA (ACA)
8 views	44.66 (40.8)	45.28 (34.45)	44.97 (37.63)
14 clock views	49.80 (41.84)	**51.78 (37.06)**	50.79 (39.45)
14 cube views	**50.99 (43.34)**	49.41 (35.48)	50.2 (39.41)
6 views (ours)	50.31 (45.17)	51.61 (37.2)	**50.96** (41.19)

TABLE X
NETWORK COMPLEXITY OF PDG, DG-MVP, AND DG-MVP WITHOUT THE PROJECTION PROCESS (W/O PROJ)

	OA (%)	Para. (M)	Time (ms)
PDG [17]	53.95	15.46	29.51
DG-MVP (w/o Proj.)	**56.19**	**3.84**	**18.95**
DG-MVP	**56.19**	**3.84**	19.52

F. Analysis of Computational Complexity

In this section, we compare the complexity of our method with that of the second-best performing method on the Sim2Real dataset, namely, PDG, and analyze the impact of the projection process. The results are summarized in Table X, where "Para." denotes the total number of trainable parameters and "Time" refers to the average inference time per sample. The inference time is measured by averaging the total inference time across all samples in the test set on an NVIDIA 1080Ti GPU. It can be seen that our method not only achieves better performance but also has significantly fewer parameters and less inference time compared to PDG. To evaluate the overhead introduced by the multiview projection, we also test a variant where the projected depth images are precomputed and loaded during inference, denoted as "DG-MVP (w/o Proj.)." The projection process itself does not introduce any additional trainable parameters and adds only 0.57-ms inference time per sample. Even with the projection step included, the total inference time is only 19.52 ms per sample, demonstrating strong potential for real-time applications. Additionally, the model contains just 3.84 million parameters, making it lightweight and well-suited for deployment on resource-constrained devices.

VII. CONCLUSION

In this article, we have proposed a novel and effective DG method, DG-MVP, for the classification of 3-D point clouds captured by RGB-D sensors. In order to mitigate the domain shift between synthetic and real-world, RGB-D sensor-captured point clouds, caused mainly by missing points and occlusion frequently occurring in RGB-D sensor data, we employ six different depth images as input. These depth images are obtained by projecting a point cloud onto six orthogonal planes. To better simulate real-world data, in addition to the traditional methods of DPs and modifying point density, we have presented a new way of data transformation by adding noisy points to synthetic data locally as well as globally. Furthermore, we have employed Depth Pooling to extract the most prominent features across all projection images. To obtain more global and local features, we have proposed a modified MMP approach, which splits the feature maps into many strips and performs max pooling at each strip. We have performed extensive experiments on the PointDA-10 and Sim-to-Real benchmarks for all synthetic-to-real settings. Results have shown that the proposed DG-MVP outperforms existing 3-D DG methods in most scenarios and even surpasses several 3-D domain adaptation methods, which require access to target domain data during training. Our findings show that projected depth images serve as an effective bridge between synthetic and real-world domains. This opens the door to leveraging well-trained large-scale vision or vision-language models to further improve generalization in complex or open-world 3-D scenarios as a future direction.

REFERENCES

[1] D. Marr, T. Poggio, E. C. Hildreth, and W. E. L. Grimson, *A Computational Theory of Human Stereo Vision*. Cham, Switzerland: Springer, 1991.

[2] R. Q. Charles, H. Su, M. Kaichun, and L. J. Guibas, "PointNet: Deep learning on point sets for 3D classification and segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 77–85.

[3] Y. Feng, Z. Zhang, X. Zhao, R. Ji, and Y. Gao, "GVCNN: Group-view convolutional neural networks for 3D shape recognition," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 264–272.

[4] C. R. Qi, Y. Li, H. Su, and L. Guibas, "PointNet++: Deep hierarchical feature learning on point sets in a metric space," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 5099–5108.

[5] Y. Li, R. Bu, M. Sun, W. Wu, X. Di, and B. Chen, "PointCNN: Convolution on X-transformed points," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 31, 2018, pp. 1–7.

[6] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, "Dynamic graph CNN for learning on point clouds," *ACM Trans. Graph.*, vol. 38, no. 5, pp. 1–12, Oct. 2019.

[7] Z. Wu et al., "3D ShapeNets: A deep representation for volumetric shapes," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 1912–1920.

[8] A. X. Chang et al., "ShapeNet: An information-rich 3D model repository," 2015, *arXiv:1512.03012*.

[9] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner, "ScanNet: Richly-annotated 3D reconstructions of indoor scenes," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 5828–5839.

[10] R. Volpi and V. Murino, "Addressing model vulnerability to distributional shifts over image transformation sets," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 7979–7988.

[11] F. Qiao, L. Zhao, and X. Peng, "Learning to learn single domain generalization," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 12556–12565.

[12] S. Motiian, M. Piccirilli, D. A. Adjeroh, and G. Doretto, "Unified deep supervised domain adaptation and generalization," in *Proc. IEEE Int. Conf. Comput. Vis.*, May 2017, pp. 5715–5725.

[13] K. Muandet, D. Balduzzi, and B. Schölkopf, "Domain generalization via invariant feature representation," in *Proc. Int. Conf. Mach. Learn.*, 2013, pp. 10–18.

[14] D. Li, Y. Yang, Y.-Z. Song, and T. M. Hospedales, "Learning to generalize: Meta-learning for domain generalization," in *Proc. AAAI Conf. Artif. Intell.*, 2018, vol. 32, no. 1, pp. 3490–3497.

[15] C. Huang, Z. Cao, Y. Wang, J. Wang, and M. Long, "MetaSets: Meta-learning on point sets for generalizable representations," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 8859–8868.

[16] S. Huang, B. Zhang, B. Shi, H. Li, Y. Li, and P. Gao, "SUG: Single-dataset unified generalization for 3D point cloud classification," in *Proc. 31st ACM Int. Conf. Multimedia*, Oct. 2023, pp. 8644–8652.

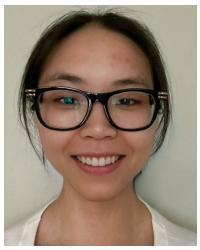
[17] X. Wei, X. Gu, and J. Sun, "Learning generalizable part-based feature representation for 3D point clouds," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 35, 2022, pp. 29305–29318.

[18] A. Goyal, H. Law, B. Liu, A. Newell, and J. Deng, "Revisiting point cloud shape classification with a simple and effective baseline," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2021, pp. 3809–3820.

[19] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.
[20] C. Qin, H. You, L. Wang, C.-C. J. Kuo, and Y. Fu, "PointDAN: A multi-scale 3D domain adaption network for point cloud representation," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 7192–7203.
[21] D. Maturana and S. Scherer, "VoxNet: A 3D convolutional neural network for real-time object recognition," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep. 2015, pp. 922–928.
[22] L. Tchapmi, C. Choy, I. Armeni, J. Gwak, and S. Savarese, "SEGCloud: Semantic segmentation of 3D point clouds," in *Proc. Int. Conf. 3D Vis. (3DV)*, Oct. 2017, pp. 537–547.
[23] H. Su, S. Maji, E. Kalogerakis, and E. Learned-Miller, "Multi-view convolutional neural networks for 3D shape recognition," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 945–953.
[24] J. Chen, M. Yang, and S. Velipasalar, "ViewNet: A novel projection-based backbone with view pooling for few-shot point cloud classification," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2023, pp. 17652–17660.
[25] M. Xu, J. Zhang, Z. Zhou, M. Xu, X. Qi, and Y. Qiao, "Learning geometry-disentangled representation for complementary understanding of 3D object point cloud," in *Proc. AAAI Conf. Artif. Intell.*, 2021, vol. 35, no. 4, pp. 3056–3064.
[26] V. Hegde and R. Zadeh, "FusionNet: 3D object classification using multiple data representations," 2016, *arXiv:1607.05695*.
[27] M. Yang, J. Chen, and S. Velipasalar, "Cross-modality feature fusion network for few-shot 3D point cloud classification," in *Proc. IEEE/CVF Winter Conf. Appl. Comput. Vis. (WACV)*. Waikoloa, HI, USA: IEEE, Jan. 2023, pp. 653–662.
[28] H. Ren, J. Wang, M. Yang, and S. Velipasalar, "PointOfView: A multi-modal network for few-shot 3D point cloud classification fusing point and multi-view image features," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2024, pp. 784–793.
[29] I. Achituve, H. Maron, and G. Chechik, "Self-supervised learning for domain adaptation on point clouds," in *Proc. IEEE Winter Conf. Appl. Comput. Vis. (WACV)*, Jan. 2021, pp. 123–133.
[30] L. Zou, H. Tang, K. Chen, and K. Jia, "Geometry-aware self-training for unsupervised domain adaptation on object point clouds," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 6403–6412.
[31] H. Fan, X. Chang, W. Zhang, Y. Cheng, Y. Sun, and M. Kankanhalli, "Self-supervised global-local structure modeling for point cloud domain adaptation with reliable voted pseudo labels," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022, pp. 6377–6386.
[32] J. Xu, X. Ma, L. Zhang, B. Zhang, and T. Chen, "Push-and-pull: A general training framework with differential augmentor for domain generalized point cloud classification," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 34, no. 8, pp. 7165–7175, Aug. 2024.
[33] J. Chen, B. Kakillioglu, H. Ren, and S. Velipasalar, "Why discard if you can recycle: A recycling max pooling module for 3D point cloud analysis," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022, pp. 549–557.
[34] C. Fan, J. Liang, C. Shen, S. Hou, Y. Huang, and S. Yu, "OpenGait: Revisiting gait recognition toward better practicality," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2023, pp. 9707–9716.
[35] Y. Fu et al., "Horizontal pyramid matching for person re-identification," in *Proc. AAAI Conf. Artif. Intell.*, vol. 33, 2019, pp. 8295–8302.
[36] M. A. Uy, Q.-H. Pham, B.-S. Hua, T. Nguyen, and S.-K. Yeung, "Revisiting point cloud classification: A new benchmark dataset and classification model on real-world data," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 1588–1597.
[37] B.-S. Hua, Q.-H. Pham, D. T. Nguyen, M.-K. Tran, L.-F. Yu, and S.-K. Yeung, "SceneNN: A scene meshes dataset with aNNotations," in *Proc. 4th Int. Conf. 3D Vis. (3DV)*, Oct. 2016, pp. 92–101.
[38] J. Sauder and B. Sievers, "Self-supervised deep learning on point clouds by reconstructing space," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 12962–12972.
[39] J. Park, H. Seo, and E. Yang, "PC-adapter: Topology-aware adapter for efficient domain adaption on point clouds with rectified pseudo-label," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2023, pp. 11530–11540.

Huantao Ren received the B.S. degree in electronic information engineering from Hangzhou Dianzi University, Hangzhou, China, in 2019, and the M.S. degree in electrical engineering from Syracuse University, Syracuse, NY, USA, in 2021, where she is currently pursuing the Ph.D. degree with the Department of Electrical Engineering and Computer Science.

Her research interests include point cloud segmentation and classification, domain generalization, and few-shot learning.



Minmin Yang received the B.S. degree in computer science from Ningbo University of Technology, Ningbo, China, in 2019, and the M.S. degree in computer science from Syracuse University, Syracuse, NY, USA, in 2021, where she is currently pursuing the Ph.D. degree in the Department of Electrical Engineering and Computer Science.

Her research interests include point cloud analysis, few-shot learning, and zero-shot learning.

Senem Velipasalar (Senior Member, IEEE) received the B.S. degree in electrical and electronic engineering from Bogazici University, Istanbul, Türkey, in 1999, the M.S. degree in electrical sciences and computer engineering from Brown University, Providence, RI, USA, in 2001, and the M.A. and Ph.D. degrees in electrical engineering from Princeton University, Princeton, NJ, USA, in 2004 and 2007, respectively.

She is currently a Professor with the Department of Electrical Engineering and Computer Science, Syracuse University, Syracuse, NY. Her research has been on mobile camera applications and applications of machine learning to different types of sensor data, including visible-range and thermal cameras, 3-D sensors, MRI, and fNIRS.

Dr. Velipasalar is a member of the Editorial Board of the IEEE TRANSACTIONS ON IMAGE PROCESSING and *Springer Journal of Signal Processing Systems*. She received the Faculty Early Career Development Award (CAREER) from the National Science Foundation in 2011.