SynMotor: A Benchmark Suite for Object Attribute Regression and Multi-Task Learning

Chengzhi Wu[1], Linxi Qiu[1], Kanran Zhou[1], Julius Pfrommer[2,3] and Jürgen Beyerer[3]
[1]*Institute for Anthropomatics and Robotics, Karlsruhe Institute of Technology, Karlsruhe, Germany*
[2]*Fraunhofer Center for Machine Learning, Karlsruhe, Germany*
[3]*Fraunhofer Institute of Optronics, System Technologies and Image Exploitation IOSB, Karlsruhe, Germany*

Keywords: Computer Vision Benchmark, Object Attribute Regression, Multi-Task Learning.

Abstract: In this paper, we develop a novel benchmark suite including both a 2D synthetic image dataset and a 3D synthetic point cloud dataset. Our work is a sub-task in the framework of a remanufacturing project, in which small electric motors are used as fundamental objects. Apart from the given detection, classification, and segmentation annotations, the key objects also have multiple learnable attributes with ground truth provided. This benchmark can be used for computer vision tasks including 2D/3D detection, classification, segmentation, and multi-attribute learning. It is worth mentioning that most attributes of the motors are quantified as continuously variable rather than binary, which makes our benchmark well-suited for the less explored regression tasks. In addition, appropriate evaluation metrics are adopted or developed for each task and promising baseline results are provided. We hope this benchmark can stimulate more research efforts on the sub-domain of object attribute learning and multi-task learning in the future.

1 INTRODUCTION

Machine learning researchers have developed tremendous inventive network models and algorithms during the past decade. In parallel, a relatively small number of benchmarks have been developed for evaluating and comparing the performance of various models. Datasets and benchmarks play important roles in the development of neural networks and drive research in more challenging directions. A good dataset can boost the development of a certain computer vision domain, *e.g.* , ImageNet (Deng et al., 2009) to image classification, PASCAL VOC (Everingham et al., 2009) and COCO (Lin et al., 2014) to image detection and segmentation, KITTI (Geiger et al., 2012) to point cloud detection, or ShapeNet (Chang et al., 2015) and S3IDS (Armeni et al., 2016) to point cloud segmentation. However, among all the computer vision tasks, the task of attribute regression is less explored due to the scarcity of suitable benchmarks. Current attribute learning methods mostly focus on outdoor pedestrians (Li et al., 2015)(Li et al., 2018) or human facials (Sarafianos et al., 2018)(Kalayeh et al., 2017). The attributes in those datasets are mostly binary (*e.g.* gender, with glasses or not), and only a few attributes are continuous variables (*e.g.* age). On the other hand, regression models with neural networks are mostly used for non-vision data. We think it would be interesting if we could contribute to bridging the gap between attribute learning and regression for the computer vision community.

In this paper, we propose SynMotor, a benchmark that gives the possibility to perform object attribute regression. Multi-task learning and multi-modal learning are also possible with the provided dataset. The benchmark originates from a sub-task within our manufacturing project which aims at the automatic disassembly of small electric motors. A mesh model dataset is first generated with a carefully developed Blender addon, in which the motor specifications are saved as object attributes. Subsequently, synthetic motor datasets of both 2D image dataset and corresponding 3D point cloud dataset are created for deep learning purposes. Apart from the object attributes ground truth, the ground truth label of common computer vision tasks including detection, classification, and segmentation are also provided. Those labels are generated automatically along with the image or point cloud generation, with no manual annotation required. On the other hand, the metrics for common computer vision tasks are already well-developed, while developing metrics for object

Wu, C., Qiu, L., Zhou, K., Pfrommer, J. and Beyerer, J.
SynMotor: A Benchmark Suite for Object Attribute Regression and Multi-Task Learning.
DOI: 10.5220/0011718400003417
In *Proceedings of the 18th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2023) - Volume 4: VISAPP*, pages 529-540
ISBN: 978-989-758-634-7; ISSN: 2184-4321

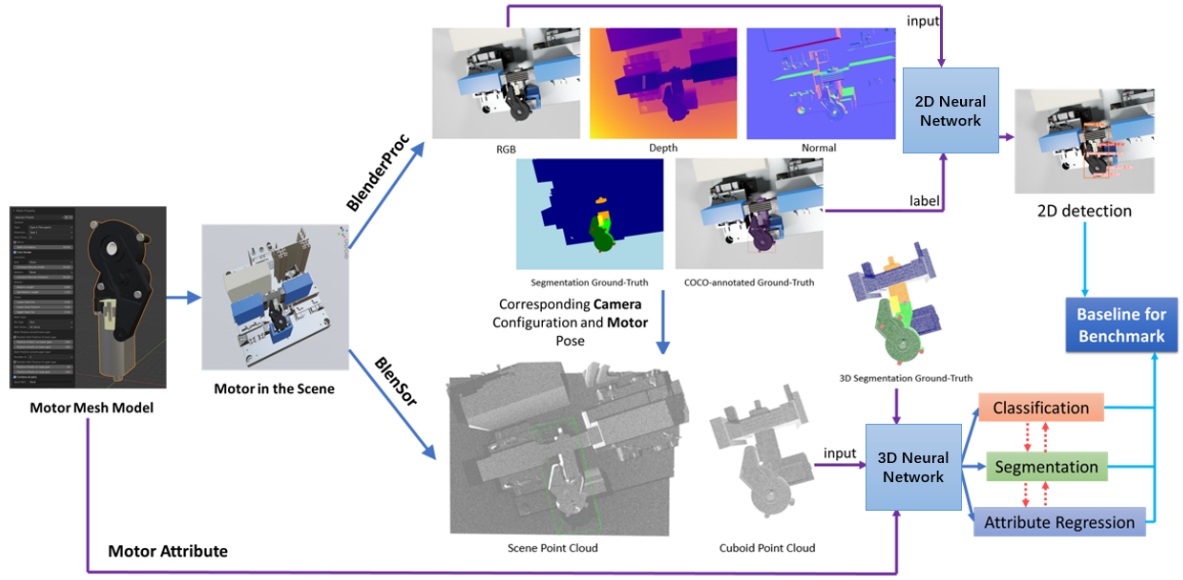


Figure 1: The framework of our work. Synthetic datasets are generated and task metrics are defined. Baseline results of 2D detection and 3D classification, segmentation, and attribute regression are provided.

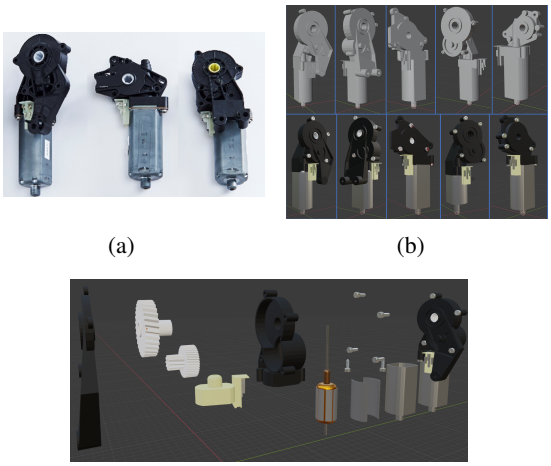
attribute regression is more difficult since the metrics need to be attribute-oriented. In our benchmark, we have designed several metrics for attribute regression given the motor attributes. Baseline results are also provided in the following experiments with some widely recognized networks. An illustrative figure of our framework is given in Figure 1.

The remainder of this paper is structured as follows: Section 2 summarizes the state-of-the-art of 3D dataset creation, multi-task learning, and attribute learning. Section 3 shows a pipeline of creating our synthetic dataset. Section 4 describes the tasks and the developed metrics. Section 5 gives baseline results with some widely recognized network models. Finally, Section 6 summarizes presented results and discusses future work.

2 RELATED WORK

3D dataset. For 3D object dataset, ModelNet (Wu et al., 2015) builds a huge dataset of 3D CAD models and provides the ModelNet40 benchmark for tasks including 3D shape classification and retrieval. Another similar work of ShapeNet (Chang et al., 2015) provides more detailed semantic annotations, its subsequent work of PartNet (Mo et al., 2019) additionally offers fine-grained semantic segmentation information for a subset of the models. (Tremblay et al., 2018) creates a dataset for object pose estimation. A large dataset of 3D-printing models is provided in Thingi10K (Zhou and Jacobson, 2016), while a more recent ABC dataset (Koch et al., 2019) collects over 1 million CAD models including many mechanical components. Regarding 3D scenes, KITTI (Geiger et al., 2012) uses a vehicle-mounted platform outfitted with a variety of sensors to capture and record road information. Data for autonomous driving tasks like 3D object detection and 3D tracking are collected. While (Ros et al., 2016) and (Khan et al., 2019) generate synthetic datasets for the segmentation and detection of objects in virtual urban scenes, (Le Hoang-An et al., 2021) generates images from virtual garden scenes. SynthCity (Griffiths and Boehm, 2019) generates point clouds of urban scenes using Blender. (Pierdicca et al., 2019) also uses Blender but for the generation of point clouds of historical objects. For indoor scenes, SUN-RGBD (Song et al., 2015), S3DIS (Armeni et al., 2016), and ScanNet (Dai et al., 2017) use different cameras to scan rooms to get 3D indoor point cloud dataset.

Multi-task Learning. Overfeat (Sermanet et al., 2014) trains on classification, localization, and detection tasks simultaneously using a single shared convolutional network. (Eigen and Fergus, 2015) implements depth prediction, normal estimation, and semantic labeling using a single multi-scale convolutional network. (Kendall et al., 2018) optimizes the weight of loss during multi-task training according to the uncertainty of each task, thereby realizing the simultaneous learning of classification and regression tasks of different orders of magnitude. MultiNet (Teichmann et al., 2018) and UberNet (Kokkinos, 2017)

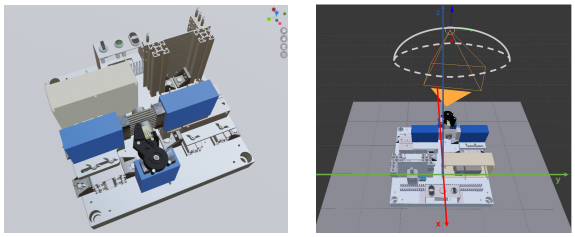


(a) (b) (c)

Figure 2: Real-world motors (a) and generated demo motors. (b) Upper row: no textures added; bottom row: textures added and rendered. (c) An explosion figure of a generated motor. The original assembled motor model is also shown at the rightmost.

offer methods for jointly performing classification, detection, and semantic segmentation using a unified architecture and achieving very efficient results. The above approaches work on the fundamental principle of a global feature extractor comprised of convolutional layers shared by all tasks and a different output branch for each task. In contrast to previous approaches, the strategy used in (Misra et al., 2016) and (Gao et al., 2019) is to have a separate network for each task while exchanging parameters between their parallel layers. Ruder (Ruder et al., 2019) divides each layer into task blocks and information sharing blocks of previous layers, and the input of each layer is a linear combination of these two blocks, so that when learning, you can choose to focus more on the relevance of the task or the task itself. PAD-Net (Xu et al., 2018) uses multi-task learning making preliminary predictions for depth, scene, and surface normal estimation, then combining these predictions to obtain the refined depth and scene parsing results.

Attribute Learning. Attribute learning refers to the process of discovering relevant attributes based on known logical principles. Visual attribute recognition has become an important research area due to its high-level semantic information. Previous work like (Russakovsky and Fei-Fei, 2010) establishes transfer learning by learning visual attributes such as colors, shapes, or textures of images, and makes connections between semantically unrelated categories. Attribute learning has a wide range of applications in the recognition of pedestrians. (Li et al., 2015) suggests two deep learning models for surveillance scenarios



(a) (b)

Figure 3: (a) Simulated scene in Blender, with a clamping system. (b) Camera setting.

to recognize based on a single attribute and multiple attributes respectively. PGDM (Li et al., 2018) analyses the salient features of the pedestrian's body structure, it is helpful for the recognition of pedestrian attributes. DIAA (Sarafianos et al., 2018) utilizes multiscale visual attention and weighted focal loss for person attribute recognition. With the widespread popularity of posting selfies in social software, the practice of recognizing facial attributes is growing, (Liu et al., 2015) provides a cascaded deep learning framework for jointly predicting attributes and localizing faces. (Kalayeh et al., 2017) leverages the information obtained from semantic segmentation to improve facial feature prediction, (Liu et al., 2018) obtains equivalent discriminative ability for face attribute recognition on CelebA and LFWA (Huang et al., 2008) datasets by learning disentangled but complementing face features with minimal supervision.

3 SYNTHETIC DATASET GENERATION

3.1 Mesh Model Generation

A Blender addon is created for the easy generation of synthetic motor mesh models. As open-source software, Blender (Bonatti et al., 2016) is a proven tool that performs well in modeling shapes and creating highly customizable addons. Our addon is able to generate motor mesh models with various specifications and save them in desired file formats. Each component of a generated motor can also be saved separately. The generated models contain the following components: (i) Pole Pot; (ii) Electric Connection; (iii) Gear Container; (iv) Cover; (v) Side Screws and (vi) Cover screws. Those are the six main categories we need to perform segmentation on for the first step of disassembly. Additionally, the following inner components have also been generated : (vii) Magnets; (viii) Armature; (ix) Lower Gear and (x) Upper Gear, as presented in Figure 2(c). To generate

Table 1: Object attributes and their notations and ranges. The x/y/z values of a key point location are treated as three separate attributes. Some of the attributes are only valid under certain conditions. The attributes are used in different designed metrics. SRE stands for size relative error, GLE stands for gear location error, MRE stands for motor rotation error, SLE stands for screw location error. See a detailed explanation regarding these metrics in Section 4.2.

Notation	Attribute	Range	Validity	Involved metric
T	Motor type	$\{0,1,2,3,4\}$	always	Cls. accuracy
N_s	Number of cover screws	$\{3,4,5\}$	always	Cls. accuracy
L_b	Bottom length	$6.2 \sim 8.0$	always	SRE
L_{sb}	Sub-bottom length	$0.6 \sim 2.0$	always	SRE
D_{lg}	Lower gear region diameter	$3.5 \sim 4.5$	always	SRE
D_{ug}	Upper gear region diameter	$5.0 \sim 6.5$	if $T = 0/1/2$	SRE
X_{lg}, Y_{lg}, Z_{lg}	Lower gear center location (xyz)	$(1.7 \sim 2.3, 1.0, 10.6 \sim 14.2)$	always	$\text{GLE}_{xyz}, \text{GLE}_{xz}$
X_{ug}, Y_{ug}, Z_{ug}	Upper gear center location (xyz)	$(1.7 \sim 2.5, 0.3 \sim 0.5, 13.5 \sim 17.3)$	if $T = 0/1/2$	$\text{GLE}_{xyz}, \text{GLE}_{xz}$
R_x, R_y, R_z	Motor rotation (rx/ry/rz)	$(\pm 15°, \pm 5°, \pm 5°)$	always	MRE
$X_{s_1}, Y_{s_1}, Z_{s_1}$	Cover screw 1 location (xyz)		always	$\text{SLE}_{xyz}, \text{SLE}_{xz}$
$X_{s_2}, Y_{s_2}, Z_{s_2}$	Cover screw 2 location (xyz)	$(-4.9 \sim 5.0,$	always	$\text{SLE}_{xyz}, \text{SLE}_{xz}$
$X_{s_3}, Y_{s_3}, Z_{s_3}$	Cover screw 3 location (xyz)	$-3 \sim -1.4,$	always	$\text{SLE}_{xyz}, \text{SLE}_{xz}$
$X_{s_4}, Y_{s_4}, Z_{s_4}$	Cover screw 4 location (xyz)	$8.6 \sim 20.7)$	if $N_s = 4/5$	$\text{SLE}_{xyz}, \text{SLE}_{xz}$
$X_{s_5}, Y_{s_5}, Z_{s_5}$	Cover screw 5 location (xyz)		if $N_s = 5$	$\text{SLE}_{xyz}, \text{SLE}_{xz}$

motors with various specifications, we provide lots of parameter options that control the type, size, position, and rotation of different parts of motor, *e.g.* screw position, gear size, or pole pot length. Figure 2(b) shows ten generated demo motors with different parameters and an exploded view of a demo motor. All the individual components mentioned above are modeled separately as illustrated.

3.2 Image and Point Cloud Generation

The generated mesh models are further used to create synthetic image and point cloud datasets. As shown in Figure 3(a), a simulated scene is built in Blender for it. Apart from the lights and cameras, to make the scene more realistic, a model of the real-world clamping system and a background panel have been added additionally. Three light sources with random changes in light intensity are placed in the scene. The camera rotates randomly on top of the scene within a certain view range yet always towards the motor, as illustrated in Figure 3(b). To create an image dataset, apart from the scene images rendered by Blender directly, BlenderProc (Denninger et al., 2019) is used to generate corresponding depth images, normal images, and segmentation ground truth images. Detection ground truth of bounding boxes of motor and screws are also provided. On the 3D synthetic data side, BlenSor (Gschwandtner et al., 2011) is used to simulate the sensors to create point cloud data as well as to generate their segmentation ground truth. 3D bounding boxes are also given. Moreover, for better learning of key objects, we additionally provide corresponding sub-point clouds for each scene by cropping random cuboid regions around the motors, but make sure to include them. The current version of our benchmark focuses more on the undismantled motors, hence inner components will not be investigated. However, any researcher interested in this part is free to use the provided addon and scripts to generate corresponding datasets according to their own needs.

3.3 Dataset Details

We have created a synthetic motor mesh dataset of 1000 motor mesh models with different specifications. They are placed in the same simulated scene but with random camera settings, random light conditions, and random mild translations and rotations. With these 1000 scenes, 1000 sets of images and 1000 point clouds are generated respectively. 80% of the data are randomly selected as the training data, while the other 20% are used as the test data. To ensure the correspondence between images and point clouds for each scene, the camera information has been saved and shared between BlenderProc and Blensor.

Note that although we only generated 1000 motors for the dataset creation, it is possible to create a much larger dataset with more motor models following a same pipeline. Scripts are used for batch generation, hence the generation process requires no exhausting labor work at all. The only manual work is to set some hyper-parameters. The dataset generation scripts will also be released. To give a better sense of how long it takes to generate a dataset of a certain size, we report the time it spent for creating our datasets of 1000 motors. To generate 1000 motor mesh models, it took around 8 hours. To generate 1000 sets of images, it took around 12 hours. To generate 1000 point clouds, it took around 21 hours. These results are based on a

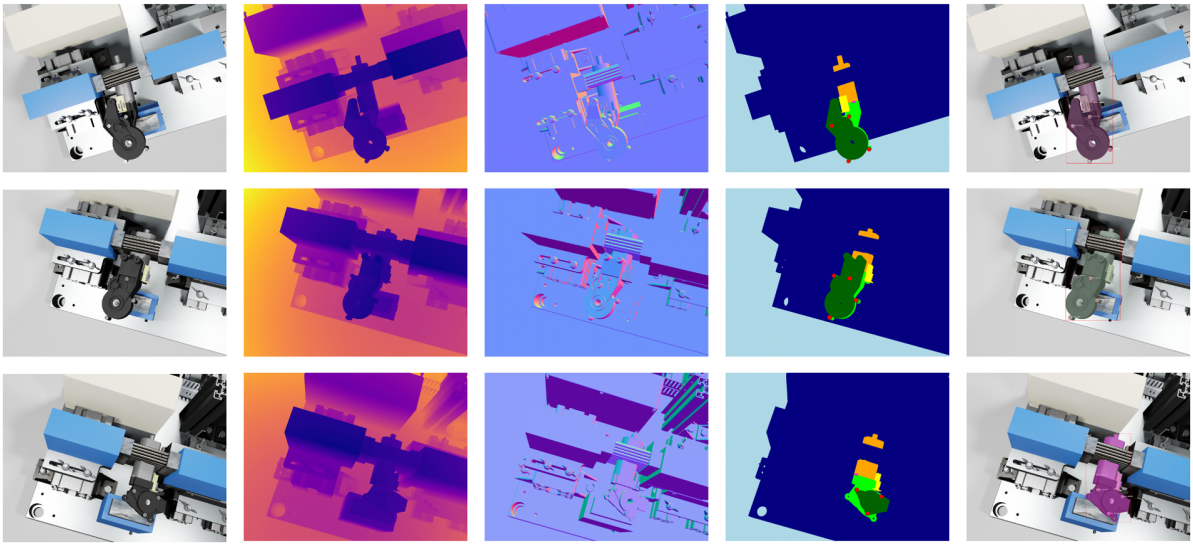


Figure 4: Demos of synthetic image data.

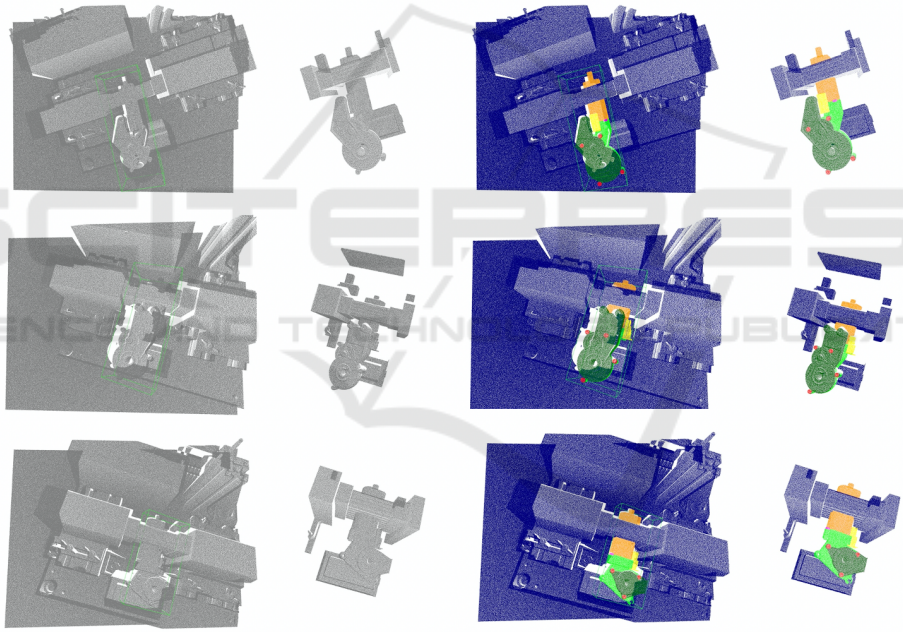


Figure 5: Demos of synthetic point cloud data.

desktop with an Intel Core i5 CPU with 16GB RAM, and a GTX 1080 GPU. Faster processing can surely be achieved with better processors and GPUs.

Our datasets are organized as follows.

Mesh Model Dataset. Part obj files and assembled full shape obj files are both saved, with corresponding material files. Apart from that, all attributes are saved in a csv file. We select 30 key attributes as the main learning targets for the benchmark and have pre-processed them for more convenient usage. The 30 attributes are given in Table 1. Attribute *Type* and *Number of cover screws* are suitable for classification tasks, while the other 28 attributes are suitable for attribute learning. Note that some attributes are invalid in some cases, *e.g.* , the xyz coordinates of 5th cover screw when the motor only has 4 cover screws. While the attribute values are set to zero when they are invalid, an additional binary mask csv file is provided for all 28 attributes of all 1000 motors. In our benchmark, all 28 attributes are continuous variables and hence are suitable for attribute regression.

Image Dataset. 1000 sets of images are provided. In each set, there are one rgb image, one depth image, one normal image, and one segmentation ground truth image. Detection ground truth of 2D bounding boxes are provided in the COCO fashion, with a visualized detection result supplemented. Demos of generated images are given in Figure 4.

Point Cloud Dataset. 1000 sets of point clouds are provided. In each set, there are one scene point cloud and one cropped point cloud which focuses on the motor region. 3D bounding boxes are provided. Segmentation ground truth labels are also provided. Figure 5 gives a demo of generated point clouds colored in their segmentation ground truth.

4 DESIGNED TASKS AND METRICS

4.1 Common Computer Vision Tasks

Detection. The detection task is to detect the positions of the key objects in the scene and to draw 2D or 3D bounding boxes around each object of interest in RGB images or point clouds. For 2D detection, the widely used mean average precision (mAP) was originally proposed in the VOC challenge (Everingham et al., 2009). We use an advanced version in COCO (Lin et al., 2014) which further considers different IoU thresholds. The two metrics are (i) mAP with IoU threshold of 0.5; (ii) average mAP with IoU threshold of $0.5, 0.55, 0.6, \ldots, 0.95$. For 3D detection, we require a 3D bounding box overlap of 70% for computing the precision-recall curve and the mAP.

Classification. The task of classification is to classify the key object in the scene into the prior-defined categories. There are 5 types of motors in our dataset, the main differences between them are the number of gears and the shape of covers. The classification task can be performed on both 2D dataset and 3D dataset. The well-known classification accuracy is used as the metric for the classification task. This is a relatively easy task in our setting since we only have 5 categories for classification and none of them is a tail category.

Segmentation. Segmentation on 2D images is the process of assigning a label to every pixel in the image such that pixels with the same label share certain characteristics. 3D point cloud segmentation is the process of classifying point clouds into different regions, so that the points in the same isolated region have similar properties. Common segmentation metrics are category-wise IoU and overall mIoU. In our case, the overall mIoU metric is used. Moreover, since the screw categories are the key categories in real-world applications, an additional metric of screw mIoU is performed.

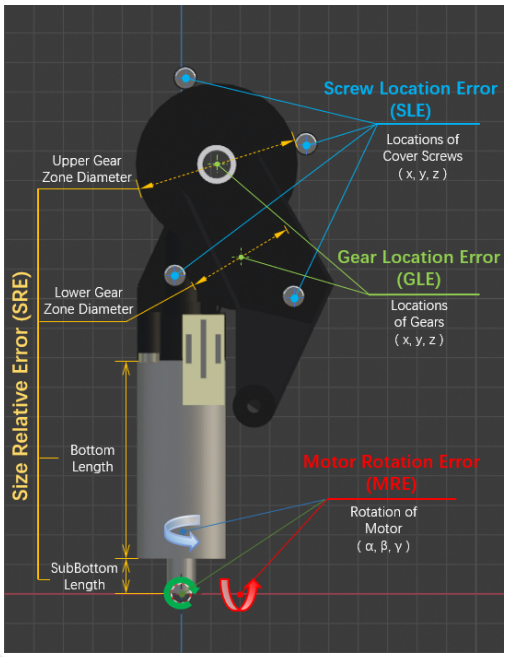


Figure 6: Object attribute regression metrics.

4.2 Object Attribute Regression

When generating the motor mesh models, their detailed geometric attributes are also saved, including the length of the pole pot, the diameter of the gear region, the positions of the screws, etc., which provides the possibility for regression learning. Evaluation metrics for assessing these learned attributes are proposed with consideration on multi-perspectives, including object size, object orientation, and object key point positions. It is worth noting that since the attributes are not always valid in all scenes, a binary mask is used for the metrics to get more accurate error information. In the following definition of each metric, for a more clear description, we use A to denote the union set of involved attributes, with a corresponding binary mask M_a. Note that the following metrics are only used for evaluation, the loss used for the training is computed with a masked mean square error (MSE) between the predicted results and the ground truth.

Size Relative Error (SRE). This metric evaluates the predicted overall motor size using four key attributes which represent the main body of motors: the lengths of the bottom and the sub-bottom part, *i.e.* pole pot, and the diameters of the gear regions. Denote A as the union set of those attributes: $A \in \{L_b, L_{sb}, D_{lg}, D_{ug}\}$ with a corresponding binary mask M_a, for a batch of N motor point clouds with ground truth A^*, the SRE metric is given as

$$\mathrm{SRE} = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{1}{\|M_{a_i}\|_1}\sum_{A_i}\frac{|M_{a_i}A_i - A_i^*|}{1 - M_{a_i} + A_i^*}\right) \tag{1}$$

where $\sum M_{a_i}$ counts the valid attribute number in ith motor and $1 - M_{a_i}$ is served as a safety term to ensure the denominator is greater than 0 When $A_i^* = 0$. For a certain attribute of a certain motor that is invalid, its ground truth value and mask value are both 0, *i.e.* $A_i^* = M_{a_i} = 0$, its size relative error is computed as zero and this attribute will not be counted in the denominator.

Note that we compute a mean error over the instance level for SRE. We believe this can represent the metric information better. The following three metrics are performed over the batch level.

Gear Location Error (GLE). This metric evaluates the distance error between the predicted location of gear region center and its ground truth. Involved attributes are the center point coordinate values. Denote A_l and A_u as the union sets of those attributes: $A_l \in \{X_{lg}, Y_{lg}, Z_{lg}\}$ and $A_u \in \{X_{ug}, Y_{ug}, Z_{ug}\}$ with masks M_{al} and M_{au}, for a batch of N motor point clouds, the GLE_{xyz} metric is given as

$$\mathrm{GLE}_{xyz} = \frac{\sum_{i=1}^{N}\left(\sqrt{\sum_{A_{l_i}}\left(M_{al_i}A_{l_i} - A_{l_i}^*\right)^2} + \sqrt{\sum_{A_{u_i}}\left(M_{au_i}A_{u_i} - A_{u_i}^*\right)^2}\right)}{\sum_{i=1}^{N}\left(\|M_{al_i}\|_1/3 + \|M_{au_i}\|_1/3\right)} \tag{2}$$

where $M_{au_i}/3$ means for the 3D coordinate of each center point, the mask should only be counted for one time. In real-world applications of disassembly, the small error along the motor normal direction (in our case, the Y axis) is sometimes irrelevant. We further provide another metric that only considers the distance error on the XZ plane with projected points. By redefining $A_l \in \{X_{lg}, Z_{lg}\}$ and $A_u \in \{X_{ug}, Z_{ug}\}$, a similar metric of GLE_{xz} is given as

$$\mathrm{GLE}_{xz} = \frac{\sum_{i=1}^{N}\left(\sqrt{\sum_{A_{l_i}}\left(M_{al_i}A_{l_i} - A_{l_i}^*\right)^2} + \sqrt{\sum_{A_{u_i}}\left(M_{au_i}A_{u_i} - A_{u_i}^*\right)^2}\right)}{\sum_{i=1}^{N}\left(\|M_{al_i}\|_1/2 + \|M_{au_i}\|_1/2\right)} \tag{3}$$

Motor Rotation Error (MRE). This metric evaluates the absolute motor rotation error. Involved attributes are the motor rotations along three axes. Denote A as the union set of those involved attributes: $A \in \{R_x, R_y, R_z\}$. Since the rotation attributes are always valid, the mask M_a is unnecessary for computation. The metric MRE is defined as

$$\mathrm{MRE} = \frac{1}{3N}\sum_{i=1}^{N}\sum_{A_i}|A_i - A_i^*| \tag{4}$$

Screw Location Error (SLE). This metric evaluates the distance error between the predicted cover screw positions and their ground truth. Involved attributes are the screw position coordinate values. Denote A_j as the union set of those attributes: $A_j \in \{X_{s_j}, Y_{s_j}, Z_{s_j}\}$ with masks M_{aj} where $j = 1,2,3,4,5$. For a batch of N motor point clouds, the SLE_{xyz} metric is given as

$$\mathrm{SLE}_{xyz} = \frac{\sum_{i=1}^{N}\sum_{j=1}^{5}\sqrt{\sum_{A_{j_i}}\left(M_{aj_i}A_{j_i} - A_{j_i}^*\right)^2}}{\sum_{i=1}^{N}\sum_{j=1}^{5}\|M_{aj_i}\|_1/3} \tag{5}$$

where $M_{aj_i}/3$ means for the 3D coordinate of each screw, the mask should only be counted for one time. Same as GLE, a metric SLE_{xz} only considers the XZ plane is defined with $A_j \in \{X_{s_j}, Z_{s_j}\}$:

$$\mathrm{SLE}_{xz} = \frac{\sum_{i=1}^{N}\sum_{j=1}^{5}\sqrt{\sum_{A_{j_i}}\left(M_{aj_i}A_{j_i} - A_{j_i}^*\right)^2}}{\sum_{i=1}^{N}\sum_{j=1}^{5}\|M_{aj_i}\|_1/2} \tag{6}$$

4.3 Multi-Task Learning

Multi-task learning (Ruder, 2017)(Zhang and Yang, 2017) means solving multiple learning tasks simultaneously, while exploiting the commonalities and differences between tasks. This can improve the learning efficiency and prediction accuracy of task-specific models compared to training the models individually. While acceptable performance can be obtained by focusing on a single task, the information that might help in getting better performance is possibly ignored. Specifically, the information comes from training signals on related tasks. By sharing representations between related tasks, the generalization ability of the model can be improved on the original task.

In our case, the above tasks can be performed simultaneously with a same backbone network. For example, classification and segmentation can be performed at the same time. Or as mentioned in the last subsection, the classification, detection, and segmentation results may be used as additional input for the regression task. Multi-modal learning is also possible by using both 2D and 3D data.

Table 2: 3D classification and segmentation baseline results.

Model	Loss	Classification	Segmentation (mIoU)								
		(Accuracy)	Overall	Background	Cover	Gear Container	Charger	Bottom	Side Screw	Cover Screw	Screw
DGCNN (cls)	L_c	100	-	-	-	-	-	-	-	-	-
DGCNN (seg)	L_s	-	93.24	99.89	98.63	95.80	97.32	98.29	79.72	83.10	82.73
DGCNN (cls+seg)	$L_c + L_s$	100	92.47	99.87	98.43	95.41	96.97	98.15	77.05	81.38	80.83
DGCNN (cls+seg)	$\sqrt{L_c L_s}$	100	92.41	99.87	98.40	95.35	96.87	98.18	76.94	81.20	80.66

Table 3: 2D Detection baseline results.

Model	mAP (IoU 0.5-0.95)	mAP (IoU 0.5)
YOLOv5n	75.3	93.6
YOLOv5s	77.8	94.2
YOLOv5m	82.7	95.9
YOLOv5l	86.7	96.4
YOLOv5x	89.1	96.8

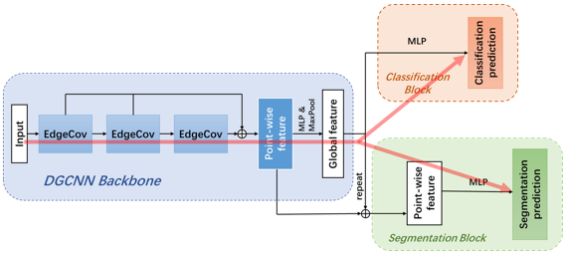


Figure 7: 3D classification and segmentation framework for baseline results.

5 BASELINE RESULTS

5.1 2D Detection

Since there is no such network architecture for 3D point cloud detection as widely recognized as YOLO series for 2D image detection, for the detection task, we give baseline results on the 2D image dataset. YOLO models of different sizes are used. All the experiments are performed with same parameter settings. The input image resolution is 640, the batch size is 16. The optimizer SGD is used with an epoch number of 200. The learning rate starts at 0.01 and decays to 0.001 with a linear decay. The mAP performances of all models are given in Table 3. It shows that the YOLO framework achieves remarkable performance on detection tasks. The mAP performance also improves when a larger network model is used.

5.2 3D Classification and Segmentation

In the past five years, a variety of network models have been proposed for point cloud data. Multi-view based (Lawin et al., 2017)(Boulch et al., 2017) and volumetric-based methods (Maturana and Scherer, 2015)(Jiang et al., 2018) are mostly used in the early years. Since the pioneer work of PointNet (Qi et al., 2017a), point-based methods which include point-wise MLP (Qi et al., 2017b), point convolution-based methods (Wu et al., 2019)(Thomas et al., 2019) and graph-based methods (Wang et al., 2019)(Chen et al., 2021)(Liang et al., 2020), gradually became the main choice. Recent work even adapt the idea of transformer (Vaswani et al., 2017) for point cloud learning, *e.g.* , PCT (Guo et al., 2021) and PT (Zhao et al., 2020)(Engel et al., 2021). Among all those methods, PointNet++ (Qi et al., 2017b) and DGCNN (Wang et al., 2019) are recognized as two key work in the domain. In this paper, we use DGCNN as the network backbone to produce baseline results.

We use an architecture of three successive EdgeConv blocks (key block in DGCNN) to learn point-wise features, a linear layer and a max pooling layer is applied subsequently to compute a global feature. For the classification task, the global feature is processed with several followed dense layers to get a classification prediction; while for the segmentation task, the global feature is repeated and concatenated with point-wise features to process through several other dense layers to predict point-wise labels. An illustrative figure is given in Figure 7. Note that although two tail blocks are different, the encoder block is identical. It is possible to use a shared encoder for both classification and segmentation tasks. In this case, by adding the losses together and performing gradient backpropagation at the same time, we can train on the classification task and the segmentation task simultaneously. This is a simple way of performing multi-task learning. The red line indicates how the information flows during one training step. The encoder block is co-trained with both tasks, while the respective task tail blocks are trained in parallel.

The settings of these three experiments are identical. We use a sub-point cloud size of 2048 points for batch training. The batch size is 16. The optimizer AdamW is used with an epoch number of 100. The learning rate starts with 1×10^{-3} and decays to 1×10^{-5} with a cosine annealing schedule. In the EdgeConv blocks, we use $K = 32$ when selecting neighbor points. Table 2 indicates that the DGCNN global feature allows for complete identification of the motor types since it is a relatively easy task. However, the segmentation results from the

Table 4: 3D regression baseline results. T stands for motor type, N_s stands for the number of cover screws. SRE stands for size relative error, GLE stands for gear location error, MRE stands for motor rotation error, SLE stands for screw location error. Results on all four kinds of methods are presented. Results with or without using the additional segmentation task in the meta-block are both presented.

Method	with seg	Classification ↑		Segmentation ↑		Regression ↓					
		T accuracy	N_s accuracy	mIoU	Screw mIoU	SRE	GLE_{xyz}	GLE_{xz}	MRE	SLE_{xyz}	SLE_{xz}
Separate	no	100	80.86	-	-	6.20%	0.3745	0.3741	0.9398	0.6171	0.6138
	yes	100	82.52	90.27	74.84	6.20%	0.3745	0.3741	0.9398	0.6171	0.6138
Pre-train	no	100	80.86	-	-	5.56%	0.3423	0.3418	0.8237	0.5615	0.5585
	yes	100	82.52	90.27	74.84	**5.41%**	0.3500	0.3499	**0.8039**	0.5556	0.5536
Parallel	no	100	75.98	-	-	5.66%	**0.3199**	**0.3196**	0.8233	**0.5500**	**0.5477**
	yes	100	76.40	88.65	71.36	5.81%	0.3759	0.3757	0.8518	0.5869	0.5829
Iterative	no	100	81.52	**-**	**-**	6.03%	0.4043	0.4040	0.9386	0.6100	0.6053
	yes	100	**82.82**	**90.49**	**75.63**	5.84%	0.3721	0.3719	0.9444	0.5759	0.5740

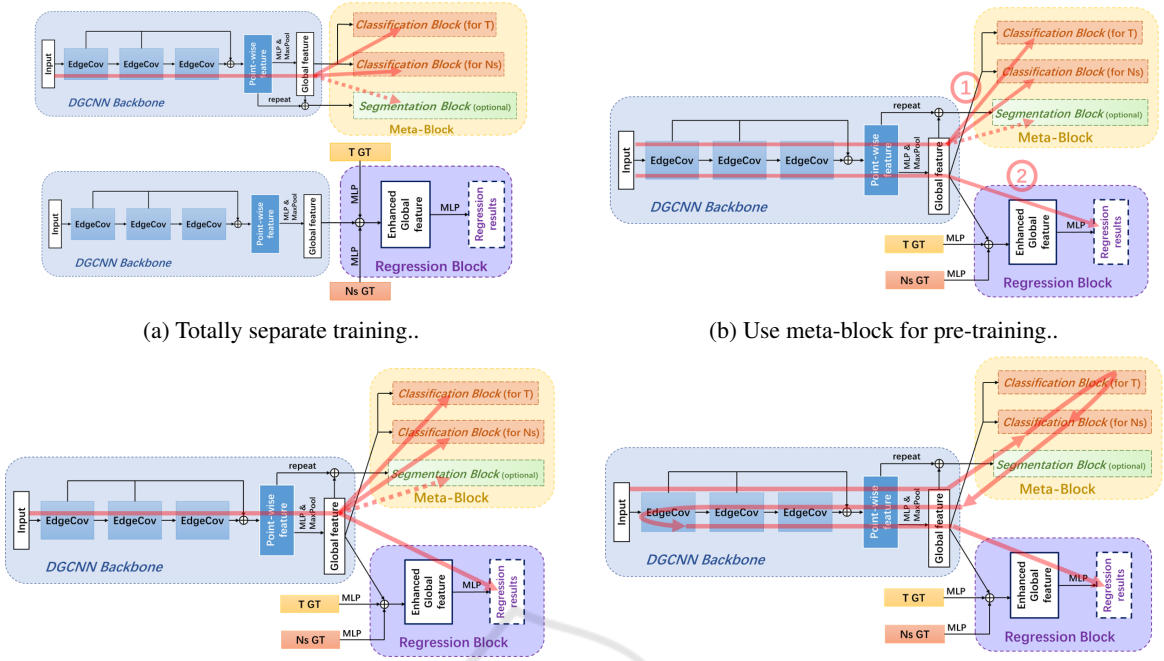
multi-task training model are not as good as that from individual training. Using loss weights to balance the gradient information from two tails blocks may improve the results. We would like to leave this problem to other researchers that are interested in this topic.

5.3 3D Object Attribute Regression

Same as in the previous subsection, DGCNN is used as the backbone for encoding point-wise features and a global feature. Moreover, as illustrated in Figure 8, we consider the classification-segmentation parallel training introduced in the previous subsection as a meta-block. Note that since the one-hot attributes T and N_s indicate the validity of some other attributes, apart from the block of motor type classification, another block for classifying the number of cover screws has also been included in the meta-block. The regression tasks compose a second tail block. In our case, we use a simple method of concatenating the global feature with one-hot features that have been encoded with MLPs, and then directly perform MLP on the enhanced global feature to get a 28-dimensional vector, which indicates the regression results of 28 attributes.

With the proposed architecture, there are several possible ways to train those blocks. (i) **Totally separate training:** the encoder and the meta-block are trained as one network first, and then another same-structure encoder and the regression block are trained as another totally separate network. No information is shared between two trainings. (ii) **Use meta-block for pre-training:** this method is similar to the last one, but the encoder weights in the second step will be initialized with the weights from the first step. The meta-block is used for pre-training. (iii) **Encoder-shared yet tail blocks trained in parallel:** the encoder is shared between two tail blocks and three blocks compose one joint network. All tasks are trained in parallel. (iv) **Encoder-shared and tail blocks trained iteratively:** in each training step, the encoder is firstly connected with the meta-block. We compute the loss, perform gradient backpropagation and update model weights with these two blocks. Then the same encoder connects with the regression block in a switch manner, the input is reprocessed with the weight-updated encoder to get new encoded representations which are used for computing the regression loss. We then again perform gradient backpropagation and weight update in these two blocks. This action performs iteratively. The red line in Figure 8 indicates how the information flows during one training step.

The experiments of the four methods used same settings. We set a sub-point cloud size of 2048 points for batch training. The batch size is 16. The optimizer AdamW is used with an epoch number of 100. The learning rate starts with 1×10^{-3} and decays to 1×10^{-5} with a cosine annealing schedule. In the EdgeConv blocks, we use $K = 32$ when selecting neighbor points. As shown in Table 4, in most cases, performing attribute regression with semantic segmentation decreases the regression error. Regression can also help in determining the number of cover screws. Among them, the separate-based method is the most simple one and it achieves the worst regression results. The pretrain-based method significantly improves the regression results. Overall, the parallel-based method achieves better performance in most regression metrics. However, since losses are merged together during the training, the classification and segmentation performances drop slightly with the parallel-based method. The iterative training-based approach achieves the best results for segmentation, but does not achieve the best results in regression. There are surely some other better ways of designing the architecture, *e.g.* , post-processing on the segmentation results for better N_s classification results, or using 2D detection results as the supplementary input. We would like to leave this as an open question for other interested researchers.



(a) Totally separate training..

(b) Use meta-block for pre-training..

(c) Encoder-shared yet tail blocks trained in parallel..

(d) Encoder-shared and tail blocks trained iteratively..

Figure 8: Four different training methods for baseline results on the 3D regression framework. The red lines indicate how the information flows inside these frameworks.

6 CONCLUSION

In this paper, a benchmark is proposed using synthetic 2D and 3D dataset, in which motors are the key objects. Motor attributes are saved during the dataset generation and are suitable for the less explored attribute regression task. Apart from the common computer vision tasks including classification, detection, and segmentation, several metrics have been designed especially for the regression task. Baseline results on several tasks are also provided. We hope this work could contribute to the attribute regression or multi-task learning domain in the computer vision community and inspire the development of other novel algorithms in the future.

ACKNOWLEDGEMENTS

The project AgiProbot is funded by the Carl Zeiss Foundation.

REFERENCES

Armeni, I., Sener, O., Zamir, A. R., Jiang, H., Brilakis, I. K., Fischer, M., and Savarese, S. (2016). 3d semantic parsing of large-scale indoor spaces. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1534–1543.

Bonatti, C., Crovisier, S., Diaz, L., and Wilkinson, A. (2016). What is... a blender? *arXiv preprint arXiv:1608.02848*.

Boulch, A., Le Saux, B., and Audebert, N. (2017). Unstructured point cloud semantic labeling using deep segmentation networks. *3DOR@ Eurographics*, 3.

Chang, A. X., Funkhouser, T. A., Guibas, L. J., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., Xiao, J., Yi, L., and Yu, F. (2015). Shapenet: An information-rich 3d model repository. *ArXiv*, abs/1512.03012.

Chen, C., Fragonara, L. Z., and Tsourdos, A. (2021). Gapnet: Graph attention based point neural network for exploiting local feature of point cloud. *Neurocomputing*, 438:122–132.

Dai, A., Chang, A. X., Savva, M., Halber, M., Funkhouser, T. A., and Nießner, M. (2017). Scannet: Richly-annotated 3d reconstructions of indoor scenes. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2432–2443.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255.

Denninger, M., Sundermeyer, M., Winkelbauer, D., Zidan, Y., Olefir, D., Elbadrawy, M., Lodhi, A., and

Katam, H. (2019). Blenderproc. *arXiv preprint arXiv:1911.01911*.

Eigen, D. and Fergus, R. (2015). Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 2650–2658.

Engel, N., Belagiannis, V., and Dietmayer, K. C. J. (2021). Point transformer. *IEEE Access*, 9:134826–134840.

Everingham, M., Gool, L. V., Williams, C. K. I., Winn, J. M., and Zisserman, A. (2009). The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88:303–338.

Gao, Y., She, Q., Ma, J., Zhao, M., Liu, W., and Yuille, A. L. (2019). Nddr-cnn: Layerwise feature fusing in multi-task cnns by neural discriminative dimensionality reduction. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3200–3209.

Geiger, A., Lenz, P., and Urtasun, R. (2012). Are we ready for autonomous driving? the kitti vision benchmark suite. *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3354–3361.

Griffiths, D. and Boehm, J. (2019). Synthcity: A large scale synthetic point cloud. *ArXiv*, abs/1907.04758.

Gschwandtner, M., Kwitt, R., Uhl, A., and Pree, W. (2011). Blensor: Blender sensor simulation toolbox. In *International Symposium on Visual Computing*, pages 199–208. Springer.

Guo, M.-H., Cai, J., Liu, Z.-N., Mu, T.-J., Martin, R. R., and Hu, S. (2021). Pct: Point cloud transformer. *Comput. Vis. Media*, 7:187–199.

Huang, G. B., Mattar, M., Berg, T., and Learned-Miller, E. (2008). Labeled faces in the wild: A database for studying face recognition in unconstrained environments. In *Workshop on faces in'Real-Life'Images: detection, alignment, and recognition*.

Jiang, M., Wu, Y., Zhao, T., Zhao, Z., and Lu, C. (2018). Pointsift: A sift-like network module for 3d point cloud semantic segmentation. *arXiv preprint arXiv:1807.00652*.

Kalayeh, M. M., Gong, B., and Shah, M. (2017). Improving facial attribute prediction using semantic segmentation. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4227–4235.

Kendall, A., Gal, Y., and Cipolla, R. (2018). Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7482–7491.

Khan, S., Phan, B., Salay, R., and Czarnecki, K. (2019). Procsy: Procedural synthetic dataset generation towards influence factor studies of semantic segmentation networks. In *CVPRW*, pages 88–96.

Koch, S., Matveev, A., Jiang, Z., Williams, F., Artemov, A., Burnaev, E., Alexa, M., Zorin, D., and Panozzo, D. (2019). Abc: A big cad model dataset for geometric deep learning. *CVPR*, pages 9593–9603.

Kokkinos, I. (2017). Ubernet: Training a universal convolutional neural network for low-, mid-, and high-level vision using diverse datasets and limited memory. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5454–5463.

Lawin, F. J., Danelljan, M., Tosteberg, P., Bhat, G., Khan, F. S., and Felsberg, M. (2017). Deep projective 3d semantic segmentation. In *International Conference on Computer Analysis of Images and Patterns*, pages 95–107. Springer.

Le Hoang-An, Mensink, T., Das, P., Karaoglu, S., and Gevers, T. (2021). Eden: Multimodal synthetic dataset of enclosed garden scenes. *2021 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1578–1588.

Li, D., Chen, X., and Huang, K. (2015). Multi-attribute learning for pedestrian attribute recognition in surveillance scenarios. *2015 3rd IAPR Asian Conference on Pattern Recognition (ACPR)*, pages 111–115.

Li, D., Chen, X., Zhang, Z., and Huang, K. (2018). Pose guided deep model for pedestrian attribute recognition in surveillance scenarios. *2018 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1–6.

Liang, Z., Yang, M., Li, H., and Wang, C. (2020). 3d instance embedding learning with a structure-aware loss function for point cloud segmentation. *IEEE Robotics and Automation Letters*, 5:4915–4922.

Lin, T.-Y., Maire, M., Belongie, S. J., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. (2014). Microsoft coco: Common objects in context. In *ECCV*.

Liu, Y., Wei, F., Shao, J., Sheng, L., Yan, J., and Wang, X. (2018). Exploring disentangled feature representation beyond face identification. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2080–2089.

Liu, Z., Luo, P., Wang, X., and Tang, X. (2015). Deep learning face attributes in the wild. *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 3730–3738.

Maturana, D. and Scherer, S. (2015). Voxnet: A 3d convolutional neural network for real-time object recognition. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 922–928. IEEE.

Misra, I., Shrivastava, A., Gupta, A. K., and Hebert, M. (2016). Cross-stitch networks for multi-task learning. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3994–4003.

Mo, K., Zhu, S., Chang, A. X., Yi, L., Tripathi, S., Guibas, L. J., and Su, H. (2019). Partnet: A large-scale benchmark for fine-grained and hierarchical part-level 3d object understanding. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 909–918.

Pierdicca, R., Mameli, M., Malinverni, E., Paolanti, M., and Frontoni, E. (2019). Automatic generation of point cloud synthetic dataset for historical building representation. In *AVR*, pages 203–219.

Qi, C., Su, H., Mo, K., and Guibas, L. (2017a). Pointnet:

Deep learning on point sets for 3d classification and segmentation. *CVPR*, pages 77–85.

Qi, C. R., Yi, L., Su, H., and Guibas, L. J. (2017b). PointNet++: Deep hierarchical feature learning on point sets in a metric space. *arXiv preprint arXiv:1706.02413*.

Ros, G., Sellart, L., Materzynska, J., Vázquez, D., and López, A. M. (2016). The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes. *CVPR*, pages 3234–3243.

Ruder, S. (2017). An overview of multi-task learning in deep neural networks. *ArXiv*, abs/1706.05098.

Ruder, S., Bingel, J., Augenstein, I., and Søgaard, A. (2019). Latent multi-task architecture learning. In *AAAI*.

Russakovsky, O. and Fei-Fei, L. (2010). Attribute learning in large-scale datasets. In *ECCV Workshops*.

Sarafianos, N., Xu, X., and Kakadiaris, I. (2018). Deep imbalanced attribute classification using visual attention aggregation. *ArXiv*, abs/1807.03903.

Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R., and LeCun, Y. (2014). Overfeat: Integrated recognition, localization and detection using convolutional networks. *CoRR*, abs/1312.6229.

Song, S., Lichtenberg, S. P., and Xiao, J. (2015). Sun rgb-d: A rgb-d scene understanding benchmark suite. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 567–576.

Teichmann, M., Weber, M., Zöllner, J. M., Cipolla, R., and Urtasun, R. (2018). Multinet: Real-time joint semantic reasoning for autonomous driving. *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 1013–1020.

Thomas, H., Qi, C. R., Deschaud, J.-E., Marcotegui, B., Goulette, F., and Guibas, L. J. (2019). Kpconv: Flexible and deformable convolution for point clouds. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6411–6420.

Tremblay, J., To, T., and Birchfield, S. (2018). Falling things: A synthetic dataset for 3d object detection and pose estimation. *CVPRW*, pages 2119–21193.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Wang, Y., Sun, Y., Liu, Z., Sarma, S. E., Bronstein, M. M., and Solomon, J. M. (2019). Dynamic graph cnn for learning on point clouds. *Acm Transactions On Graphics (tog)*, 38(5):1–12.

Wu, W., Qi, Z., and Li, F. (2019). Pointconv: Deep convolutional networks on 3d point clouds. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9613–9622.

Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., and Xiao, J. (2015). 3d shapenets: A deep representation for volumetric shapes. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1912–1920.

Xu, D., Ouyang, W., Wang, X., and Sebe, N. (2018). Padnet: Multi-tasks guided prediction-and-distillation network for simultaneous depth estimation and scene parsing. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 675–684.

Zhang, Y. and Yang, Q. (2017). A survey on multi-task learning. *ArXiv*, abs/1707.08114.

Zhao, H., Jiang, L., Jia, J., Torr, P. H. S., and Koltun, V. (2020). Point transformer. *ArXiv*, abs/2012.09164.

Zhou, Q. and Jacobson, A. (2016). Thingi10k: A dataset of 10, 000 3d-printing models. *ArXiv*, abs/1605.04797.