AssemblyNet: A Point Cloud Dataset and Benchmark for Predicting Part Directions in an Exploded Layout

Jesper Gaarsdal[1,3*] Joakim Bruslund Haurum[2*] Sune Wolff[1] Claus Brøndgaard Madsen[3]
[1] SynergyXR ApS, Aarhus, Denmark
[2] Visual Analysis and Perception (VAP) Laboratory, Aalborg University & Pioneer Centre for AI, Denmark
[3] Computer Graphics Group, Aalborg University, Denmark
jg@synergyxr.com, joha@create.aau.dk, sw@synergyxr.com, cbm@create.aau.dk

Abstract

Exploded views are powerful tools for visualizing the assembly and disassembly of complex objects, widely used in technical illustrations, assembly instructions, and product presentations. Previous methods for automating the creation of exploded views are either slow and computationally costly or compromise on accuracy. Therefore, the construction of exploded views is typically a manual process. In this paper, we propose a novel approach for automatically predicting the direction of parts in an exploded view using deep learning. To achieve this, we introduce a new dataset, AssemblyNet, which contains point cloud data sampled from 3D models of real-world assemblies, including water pumps, mixed industrial assemblies, and LEGO models. The AssemblyNet dataset includes a total of 44 assemblies, separated into 495 subassemblies with a total of 5420 parts. We provide ground truth labels for regression and classification, representing the directions in which the parts are moved in the exploded views. We also provide performance benchmarks using various state-of-the-art models for shape classification on point clouds and propose a novel two-path network architecture. Project page available at https://github.com/jgaarsdal/AssemblyNet

1. Introduction

Exploded views are commonly used in technical illustrations to visualize the assembly and disassembly of complex machinery or equipment. For product showcasing and instructional videos the exploded views are often rendered as 3D animations, and in recent years augmented- and virtual reality technologies have been used to bring the exploded views even closer to the observer [9, 26]. Traditionally, these illustrations are created manually. Previous studies have presented approaches to create exploded views automatically, often using information from Computer-Aided

*Equal contribution.

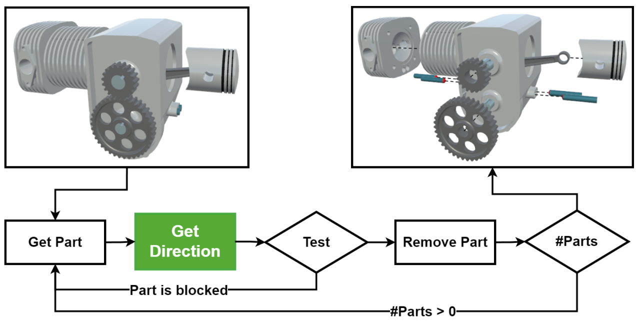


Figure 1. **The assembly-by-disassembly process.** An iterative process of automatically creating an exploded view of an engine assembly with eight exploding parts. The step retrieving a part disassembly direction is highlighted as the focus of this paper.

Design (CAD) models and applying algorithms to calculate distances and directions of parts in an iterative process, such as the one presented in Fig. 1. However, these methods typically need a long time to process, and the time increases significantly, depending on the complexity of the assemblies [2, 17, 24]. Few studies have attempted to create real-time exploded views of assemblies and they do so by sacrificing on the technical accuracy of the visualization or by assembly-specific configuration of parameters [2, 9].

While machine learning methods have been used in the optimization of assembly sequences [29, 36], no previous work has employed it in the generation of explosion layouts. In this paper, we present the open-source *AssemblyNet* dataset consisting of 5420 assembly part samples with annotated part directions. The dataset and proposed methods compliment the iterative process by predicting valid disassembly directions for parts. Our contributions are threefold:

- A publicly available point cloud dataset with annotated directions of parts in exploded view visualizations.
- A direct comparison of commonly used point cloud classification models for predicting part directions.

- A novel two-path point cloud classification model for predicting part directions.

2. Related Works

Automatic generation of exploded layouts. The concept of exploded layouts has been widely used in engineering drawings and technical illustrations for decades, especially since the introduction of CAD systems. Traditionally, these layouts are created manually by domain experts, which is a time-consuming process. With advancements in CAD systems, researchers and industry professionals have been exploring ways to automatically generate exploded layouts and assembly sequences since the early 1990s [23].

Much of the early work in the field relies on calculating contact points and blocking relationships from CAD data which takes a long time to compute even for simple assemblies [1, 17]. This information is then used to determine assembly sequences by iteratively removing parts following an assembly-by-disassembly approach. Recent work has attempted to address the computation times of these methods by simplification, either by reducing the number of directions a part may move in, or by using a simpler representation of the geometry such as a bounding box [9, 30, 35]. Each of these approaches, however, has its limitations, and may lead to false-positive blocking constraints.

While techniques from machine learning have been studied in the context of Assembly Sequence Planning [4,6,28], to the best of our knowledge, there has been no previous studies on using machine learning in the process of creating explosion layouts of assemblies. As such, there are also no available datasets or benchmarks for this task.

Point cloud analysis. With the rapid increase of 3D sensors and LiDAR technology, point cloud data has become increasingly prevalent. In deep learning point clouds have been explored for various applications [5, 13, 16]. 3D point clouds are unordered sets of data points in 3D space, which was one of the early challenges in utilizing them for deep learning. One approach that is used in projection-based networks is to transform the irregular structure of the 3D point cloud into a regular one. Some of these networks use multi-view projection, creating several 2D images from multiple viewpoints which can be processed by traditional 2D CNNs [5, 10, 19, 25]. Another approach is to divide the 3D space of the point cloud into a grid of voxels [18], processing the volumetric data using 3D CNNs. The projection-based methods have achieved high accuracy on classification tasks, but they may lose important details due to the projection. In contrast, voxel-based methods can keep the spatial information by increasing the resolution of the grid, but this may not be feasible due to the memory and computation costs. This limitation can be addressed by an irregular grid structure that offers adaptive resolution, such as octrees [22], but they can still introduce significant computational overhead from tree maintenance and traversal.

Point-based networks process the point cloud directly and retain more of the original information. In 2017, Qi *et al.* [20] proposed PointNet able to directly consume point clouds for classification and segmentation tasks. This was followed by PointNet++ [21] which took the feature extraction on unordered sets from PointNet and employed it in a hierarchical structure to aggregate local and global information. Similarly, Wang *et al.* [32] proposed the Dynamic Graph Convolutional Neural Network (DGCNN), which aggregates local information by constructing a graph from the k-nearest neighbours (kNN) for each point. More recently, Zhao *et al.* [37] proposed Point Transformer which uses the increasingly popular Transformer architecture [31]. In order to capture the relationships between points this model uses self-attention on pairs of neighboring points. For an in-depth review of deep learning for 3D point clouds, we refer to the survey by Guo *et al.* [12].

All of these proposed networks are typically benchmarked for classification and segmentation tasks. For shape classification, the dataset most commonly used is ModelNet40 [33], which contains more than 12 thousand CAD models and their point clouds from 40 classes. For segmentation tasks, the most common datasets are S3DIS [3] and ShapeNetPart [34]. The S3DIS dataset for semantic segmentation of scenes contains 271 rooms made by 3D scanning and each point in the point clouds is labeled as one of 13 classes. The ShapeNetPart dataset for object part segmentation contains more than 16 thousand models from 16 model categories separated into 2-6 parts.

3. AssemblyNet

In this section we present how the data for the AssemblyNet dataset was collected (Sec. 3.1), how the ground truth labels were generated (Sec. 3.2), and how the dataset is constructed (Sec. 3.3).

3.1. Data Collection

The point clouds in the dataset are sampled from 3D models consisting of real assemblies provided by a water pump manufacturing company, as well as various industrial assemblies [*] and LEGO models [*] from online model collections. Due to licensing constraints and non-disclosure agreements the original 3D models are not included in the dataset, instead only the sampled point clouds are available.

The dataset includes a total of 44 assemblies with 16 real assemblies from the manufacturing company, 14 mixed industrial assemblies, and 14 LEGO models. In order to not confuse the neural networks by moving parts in directions where they would be blocked, each of these assemblies were

[*]TurboSquid: https://www.turbosquid.com/
[*]Mecabricks: https://www.mecabricks.com/

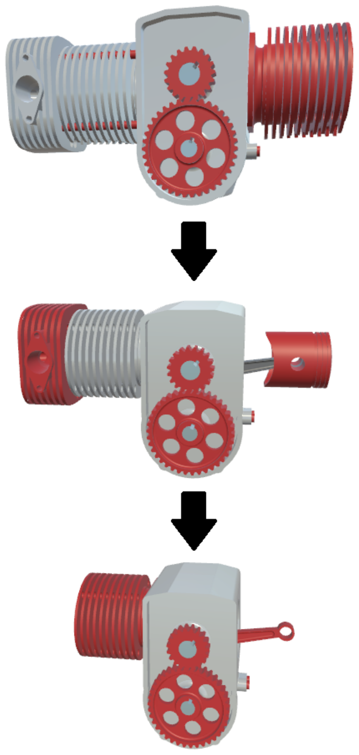


Figure 2. **Example of subassemblies.** An example of how subassemblies are created iteratively by removing parts that are unblocked at each step. In this example, the red parts are the ones that can be removed and that are used as samples in the dataset. As parts are removed from one layer to the next, new parts become unblocked and can be used. Parts that are not blocking other parts can be reused as samples (*e.g.* the gears in this figure).

split into several layers of subassemblies using the Blender 3D editor. An example of these layers is shown in Fig. 2. The layers were created iteratively, by manually removing only the parts that could realistically be removed at each iteration. This separation into layers provides a total of 495 subassemblies with a total of 5420 parts used as the samples in the dataset. The number of parts in a single subassembly ranges from 2-74. Examples of assemblies and subassemblies are shown in the supplementary materials (supp. mat.).

Using the Unity game engine [27], two initial point clouds are sampled for each of the 5420 parts, a point cloud of just the part and a point cloud of the entire subassembly of that sample. The point clouds are sampled using a Monte Carlo method with 20480 points for the subassembly and 10240 for the part. The two point clouds are stored separately but normalized as a single point cloud to preserve the spatial information of the part points relative to the subassembly. In a subsequent processing step, we use the CloudCompare 3D tool [7] to subsample the point clouds to 512 and 1024 point variants. Both variants are included in the dataset. An example of a part- and subassembly point cloud is shown in Fig. 3.

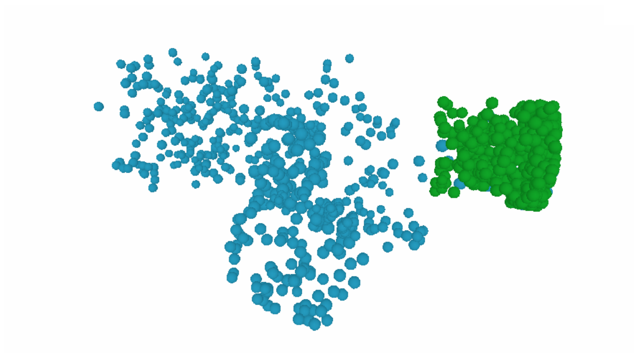


Figure 3. **Example point cloud.** A part point cloud (green) and subassembly point cloud (blue), both with 512 points, shown in the same 3D space.

3.2. Ground Truth

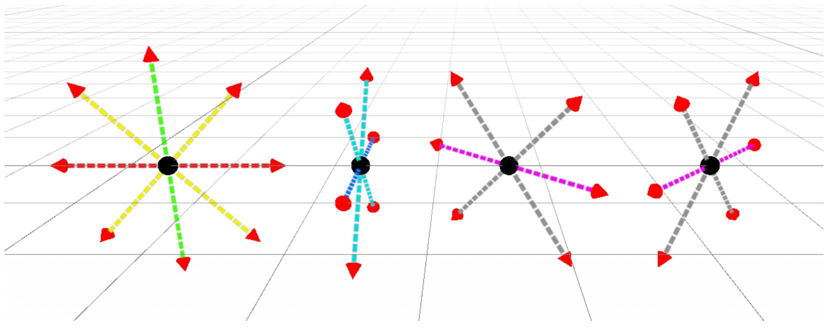
The dataset includes continuous and discrete ground truth values for regression and for classification, respectively. For regression, the ground truth are the normalized 3D directions that parts are moved in. These were created in the Unity game engine by manually separating each part and logging the direction from the start- to the end position. For classification, the number of possible world space directions are discretized to 26 classes representing directions at roughly 45° angles to each other as shown in Fig. 4a. A sample is assigned the class that is closest to its actual direction, using the dot product between the two directions.

Although most part directions follow the global axes ($\pm X, \pm Y, \pm Z$), there is a loss of fidelity with the reduction of possible directions, as seen in the example in Fig. 4b, however, we believe this could be alleviated post-inference by mapping the predicted class to the closest local axis on the oriented bounding box of the part [9, 35].

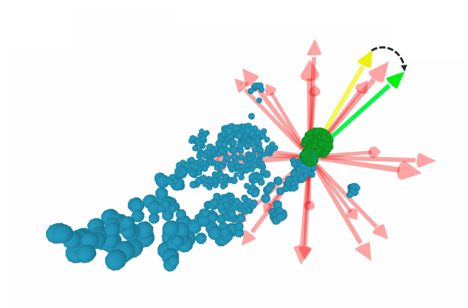
3.3. Dataset Construction

The dataset is created by splitting the data into a training set of 4334 samples ($\approx$ 80%), a validation set of 542 samples ($\approx$ 10%), and a test set of 544 samples ($\approx$ 10%). The splits are created manually following two rules in order to avoid data leakage and contamination: each assembly is represented in each split, each subassembly is only represented in one split. Subassemblies are selected randomly for a split, based on their number of samples.

Looking at the ground truth for classification, the class distribution of each split is shown in Fig. 5. It can be seen that the dataset is heavily skewed towards the cardinal directions, while several classes aren't represented. While imbalanced, this is representative of the real life use cases as these classes represent the global axes ($\pm X, \pm Y, \pm Z$).



(a) The 26 world space directions used as the classification ground truth labels. Here the directions are split into four planes to reduce visual clutter. Two planes are orthogonal and include the six global axes ($\pm X, \pm Y, \pm Z$), two planes are at 45° angles to the first two.



(b) An example of how the regression ground truth values (yellow arrow) are mapped to world space directions for the classification ground truth (green arrow).

Figure 4. **Discretized ground truth directions.** Visualizations of the classification- and regression ground truth directions.

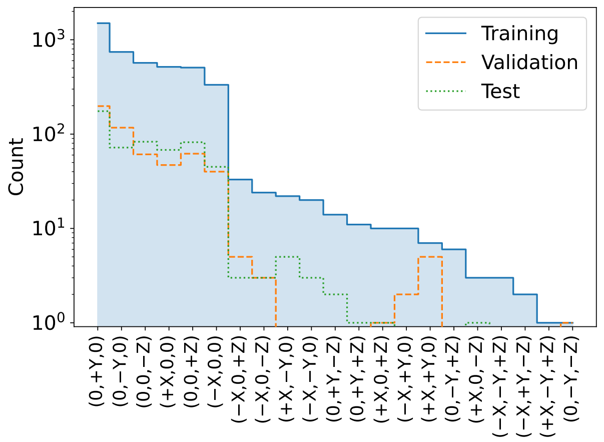


Figure 5. **Discretized direction distribution.** The class distribution of part directions are shown across the three splits. There is a large data imbalance, with an emphasis on the cardinal directions, and five possible part directions not present in any of the splits. Note that the y-axis is on a log-scale.

4. Two-path Network Architecture

In this section, we present a two-path network architecture for the AssemblyNet dataset. By adopting a two-path approach we seek to utilize the fact that each sample comprises two separate point clouds - one for the part and one for the entire subassembly. By processing these point clouds separately, we hope to capture distinct features and spatial relations from each, which may otherwise be overlooked when the point clouds are concatenated.

4.1. Overview

The overall two-path architecture draws inspiration from recent work by Guo *et al*. [11], employing a transformer network approach of encoder- and decoder layers, shown to achieve state-of-the-art performance on point cloud classification and segmentation [37], as well as cross-attention between the two paths. For our benchmarks, we have tested this architecture using two different backbones, one using the EdgeConv layers from DGCCN (referred to as TP-DGCNN) as the encoder layers and one using the point set abstraction layers from PointNet++ (referred to as TP-PointNet++) as the encoder layers. An overview of the network with the DGCNN backbone is shown in Fig. 6.

4.2. Paths

The main path of the network is the part path, processing the point cloud of the specific part in order to capture geometric features and characteristics specific for that part. The subassembly path aims to add context in the terms of the spatial relationship between the part and the rest of the subassembly. For this, we use cross-attention to collect conditioned features that are then concatenated with the encoder features. These concatenated features are then subjected to max- and average pooling to reduce dimensions and capture global features, as in [32], before they are concatenated with the decoder features and passed through the matching decoder layer. For the cross-attention module, we use the features from the part path as the query and the features from the subassembly path as the key and value.

5. Experimental Design

In this section, we describe the approach and protocols used to produce the benchmarks presented in Sec. 6. First, we present the state-of-the-art point cloud classification methods used for the benchmarks and ablation study (Sec. 5.1). Second, we describe the protocol and hyperparameters used for the training of classification and regression models (Sec. 5.2). Lastly, we describe how we evaluate the results of the classification models against the regression models (Sec. 5.3).

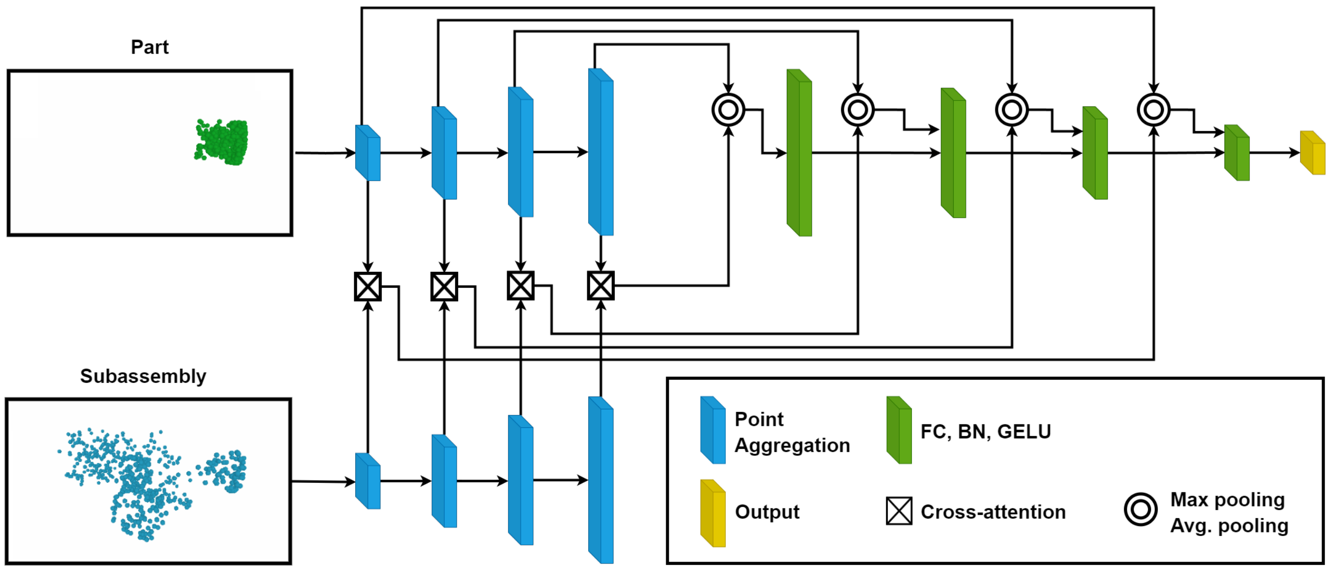


Figure 6. **Overview of the proposed Two-Path Network.** The TP-DGCNN two-path network architecture using the four EdgeConv layers from DGCNN as the encoder (Point Aggregation) and four corresponding decoder layers consisting of a fully connected layer, batch normalization and GELU activation function. The input to the decoder layers are the features from the part encoder layers, the cross-attention between the part- and subassembly features, and the output from the previous decoder layer. The dimensions of the encoder features and the cross-attention features are reduced by max- and average pooling. The size of a layer in the figure illustrates the number of features at that layer. The TP-PointNet++ network shares the same overall architecture but with three encoder layers being the three set abstraction layers from PointNet++ with three corresponding decoder layers.

5.1. Methods

To validate the effectiveness of the AssemblyNet dataset, we present results from four models from the domain of point cloud shape classification and segmentation. The models are selected as they represent different approaches to capturing spatial relationships and geometric features. The PointNet network, proposed by Qi *et al.* [20], applies a shared multi-layer perceptron (MLP) to each point independently, capturing individual point features. Later, Qi *et al.* [21] presented PointNet++, an extension that applies PointNet recursively on nested partitions of the point cloud in order to capture local and global structures at different scales. Presented by Wang *et al.* [32], DGCNN constructs a graph over the point cloud and applies convolutions on its edges to capture local geometric structures. Lastly, Goyal *et al.* [10] recently presented SimpleView, a simple projection-based approach, creating several 2D images of the point cloud from different perspectives and applies 2D convolutions using the ResNet-18 network [14]. For all networks we find that changing the standard activation functions to the GELU activation function [15] improves performance. Furthermore, given that the samples consist of two separate point clouds, we also present results from two variants of a two-path network (Sec. 4), using the best performing of the four base models as a backbone.

5.2. Training Protocol

We base our training on the original DGCNN training protocol [32], as it has been shown to work well for various models [10]. Using DGCNN, for both the classification and regression training settings, we conducted a randomized hyperparameter search with 300 runs per training setting, testing each configuration three times to account for variance. The investigated hyperparameters are shown in Tab. 1. The investigated settings were trained for 20 epochs with early stopping based on the classification validation accuracy or the regression validation loss.

In all training runs, we train the models for 250 epochs with a batch size of 32. For the four base models we concatenate the 512 point part- and subassembly point clouds for a total of 1024 points. For the two-path models we use the 1024 point part- and subassembly point clouds, such that both paths receive 1024 points. The reasoning for this is that each path contains the complete feature extraction from either DGCNN or PointNet++, and PointNet++ in particular is designed for 1024 points as the input. Furthermore, 1024 points is the most widely used input scheme [10]. We train classification networks using a Label-Smoothed Cross-Entropy (LS-CE) loss with a smoothing factor of 0.1, whereas regression networks are trained with an L1 loss.

Previous studies have shown that rotation augmentation generally lowers accuracy and is primarily used for rota-

Hyperparameters	Range	Initial	Final (cls)	Final (reg)
Optimizer	[SGD, Adam]	Adam	Adam	Adam
Learning Rate (SGD)	[0.1, 0.05, 0.01, 0.005, 0.001]	0.1	0.01	0.01
Learning Rate (Adam)	[0.01, 0.005, 0.001, 0.0005, 0.0001]	0.001	0.0005	0.0001
Momentum (SGD)	[0.9, 0.98]	0.98	0.9	0.9
Epsilon (Adam)	[10^{-4}, 10^{-6}, 10^{-8}]	10^{-8}	10^{-4}	10^{-6}
Weight Decay	[10^{-4}, 10^{-5}, 10^{-6}]	10^{-5}	10^{-4}	10^{-5}
kNN value (DGCNN, TP-DGCNN)	[20, 25, 30]	20	20	20
Views (SimpleView)	[6, 9, 12]	6	6	6

Table 1. **Considered hyperparameters and search ranges.** Overview of the searched hyperparameters, the initial values, and the final values for both classification (cls) and regression (reg).

tion invariance [10]. However, as our labels are world space directions, different orientations of the point cloud should yield different predicted directions. To this effect, we use rotation augmentation on both point clouds and ground truth. For classification, we select a random class as the new ground truth and rotate the point cloud from the original direction to the new one. For regression we simply rotate both the point cloud and ground truth by the same random axis and angle. For both, we use a probability of 30%. For other data augmentations, we use random translation, scaling, and dropout, as used in the original DGCNN protocol.

5.3. Evaluation Metrics and Protocol

Due to the intrinsic difference of the classification and regression tasks we report performance using different metrics. For classification models, we use the macro-accuracy, which assigns equal weight to all classes, C, see Eq. (1)

$$\text{Macro Accuracy} = \frac{1}{C}\sum_{i}^{C}\text{Acc}_i, \tag{1}$$

where Acc_i is the accuracy for the ith class, and C is the total number of considered classes.

For regression models, we measure the performance using the average angular similarity measure, see Eq. (2)

$$\text{Average Angular Similarity} = \frac{1}{N}\sum_{i}^{N} 1 - \frac{\theta_i}{\pi}, \tag{2}$$

where θ_i is the angle between the predicted and ground truth directions, and N is the number of data points considered. The angular similarity measure was chosen over the cosine similarity, as it assigns more distinguishable values when two vectors are nearly parallel.

In order to compare the classification and regression models, we map the predictions of the regression models to one of the 26 classes using the same method used for the ground truth labels (as described in Sec. 3.2).

Additionally, due to the unique nature of the AssemblyNet dataset, it is possible to rotate each datapoint to all possible 26 classes, and thereby check how well the networks generalize to all classes. Therefore, for the classification task we use both the original direction distribution (denoted Original-Dir.), as well as the equalized direction distribution (denoted Equal-Dir.) where each data point is evaluated for all 26 orientations.

6. Experimental Results

We report the metric performance for the classification task in Tab. 2 and the regression task in Tab. 3. For the classification setting, we find that the PointNet++ based networks outperform all other networks. In the Original-Dir. evaluation setting the standard PointNet++ performs best, followed by the standard DGCNN. However, when considering the more generalized Equal-Dir. setting we find that the TP-DGCNN and TP-PointNet++ networks outperform their standard variations. This is a clear sign that the two-path networks have merit when predicting the part directions. We also find that the PointNet and SimpleView networks perform poorly. This makes sense as the PointNet network lacks any form of global information, whereas the SimpleView network is limited by only being able to extract global information from fixed perspectives.

In order to better understand the mistakes made by TP-PointNet++, we evaluate the distribution of angular similarities, see Fig. 7. We find that the most common mistakes are either predicting the direct opposite or orthogonal direction of the ground truth, and show examples of this in supp. mat.

For the the regression setting, we find that the standard variants of the PointNet++ and DGCNN networks outperform the two-path variants. This is similar to how the standard variations outperform the two-path networks in the Original-Dir. distribution setting. When casting the regression predictions into the classification settings we find that the performance drops significantly, see Tab. 4. However, we find for all but the Original-Dir. test split, the TP-

Table 2. **Macro accuracy of classification networks.** We evaluate all networks for the classification task across both validation and test splits, as well as the Original-Dir. and Equal-Dir. distributions. We highlight the best performing model per split in **bold**.

Model	Original-Dir.		Equal-Dir.	
	Val.	Test	Val.	Test
PointNet	35.67	53.28	37.74	40.27
PointNet++	**84.02**	**90.20**	60.88	62.82
DGCNN	72.36	87.43	55.76	58.55
SimpleView	67.14	65.84	43.97	45.29
TP-DGCNN	62.25	85.61	58.91	62.20
TP-PointNet++	79.19	78.97	**65.13**	**69.49**

Table 3. **Angular similarity of regression networks.** We evaluate all networks trained for the regression task across both the validation and test splits using the angular similarity measure. We highlight the best performing model per split in **bold**.

Model	Angular Similarity	
	Val.	Test
PointNet	0.724	0.740
PointNet++	**0.902**	**0.906**
DGCNN	0.875	0.874
SimpleView	0.772	0.755
TP-DGCNN	0.857	0.855
TP-PointNet++	0.874	0.900

Table 4. **Macro accuracy of regression networks.** We evaluate all networks trained for the regression task across both the validation and test splits, as well as the Original-Dir. and Equal-Dir. data distributions. The output of the regression networks are mapped to one of the 26 discrete classes as described in Sec. 3.2. We highlight the best performing model per split in **bold**.

Model	Original-Dir.		Equal-Dir.	
	Val.	Test	Val.	Test
PointNet	19.01	18.38	5.59	5.60
PointNet++	39.13	38.66	9.65	9.47
DGCNN	44.02	**43.24**	7.35	7.26
SimpleView	18.06	21.96	6.18	5.71
TP-DGCNN	36.58	34.02	8.27	8.29
TP-PointNet++	**45.51**	41.68	**13.62**	**14.30**

PointNet++ network again outperform all other networks.

7. Ablation Studies

We conduct three ablation studies in order to determine the effect of the proposed rotation augmentation, considering the classification task as an ordinal regression task, and the total point cloud sizes. Unless otherwise noted, all training settings are kept fixed as described in Sec. 5.2 and evaluated using the validation split.

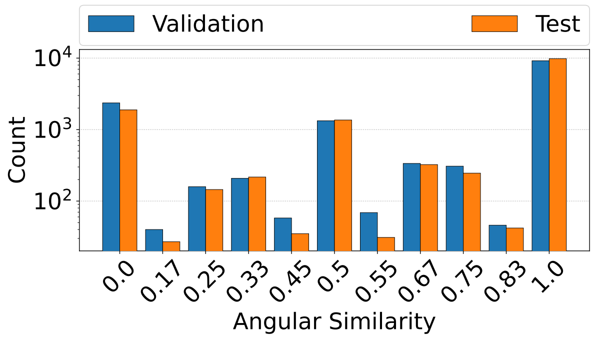


Figure 7. **Distribution of angle similarities for TP-PointNet++ classification predictions.** The angular similarity between predicted and true classes for TP-PointNet++ on the Equal-Dir. evaluation protocol. Note that the y-axis is on log-scale. Most errors occur due to predicting the direct opposite or orthogonal direction.

7.1. Effect of Rotation Augmentation

As mentioned in Sec. 5.2, using rotation augmentation has previously been found to lower the accuracy of point cloud models [10]. However, since the considered task of predicting directions is distinct from the typical point cloud classification and segmentation task, we find it necessary to verify the use of rotations. We find that by applying rotation augmentation during training it is possible to cover the rarer (and even nonexistent) classes. This leads to a direct improvement in the Equal-Dir. evaluation setting, see Tab. 5. However, we also find that not using rotation augmentation, improves performance for all networks except for the PointNet++ variants in the Original-Dir. setting.

7.2. Effect of SORD Ordinal Regression Loss

When discretizing the possible part directions into 26 classes an inherent relative ordering exists between the different classes, as the severity of an incorrect prediction is linked to how many degrees the prediction is off (*e.g.* an error of 180° is worse than an error of 45°). Therefore, the classification task could be interpreted as an ordinal regression task. To check whether the ordinal regression perspective leads to improved metric performance, we compare the LS-CE loss with the SORD loss proposed by Díaz and Marathe [8] using the Cosine Distance as the distance function. As shown in Tab. 6 we find that across all networks the LS-CE loss objective results in a better macro accuracy.

Table 5. **Effect of rotation augmentation.** We determine the effect of using rotation augmentations in the classification setting. We find that when applying rotation augmentation the performance increases for all tested networks for the more general Equal-Dir. direction distribution.

Model	Original-Dir.		Equal-Dir.	
	w/o aug	w/ aug	w/o aug	w/ aug
PointNet	43.48	35.67	6.56	37.74
PointNet++	74.95	84.02	7.96	60.88
DGCNN	76.03	72.36	8.68	55.76
SimpleView	76.02	67.14	8.48	43.97
TP-DGCNN	80.22	62.25	7.59	58.91
TP-PointNet++	69.89	79.19	12.06	65.13

Table 6. **Effect of SORD ordinal regression loss.** We compare the performance in the classification setting when using the classification-based LS-CE loss or the ordinal regression-based SORD loss. We find that the LS-CE loss outperforms SORD across all networks and direction distributions.

Model	Original-Dir.		Equal-Dir.	
	SORD	LS-CE	SORD	LS-CE
PointNet	23.48	35.67	7.98	37.74
PointNet++	51.72	84.02	25.35	60.88
DGCNN	60.56	72.36	20.76	55.76
SimpleView	33.55	67.14	10.45	43.97
TP-DGCNN	51.27	62.25	25.16	58.91
TP-PointNet++	58.04	79.19	36.09	65.13

7.3. Effect of Point Cloud Size

As described in Sec. 5.2 the four state-of-the-art networks are trained with point clouds consisting of 1024 points, while the two-path networks are fed two point clouds each consisting of 1024 points *i.e.* a total of 2048 points. In order to ensure that the improved performance of the two-path networks was not solely from the increase in point cloud sizes we compare the standard and two-path DGCNN and PointNet++ variants when trained with a total of 1024 and 2048 points, see Tab. 7. We find that the two-path networks outperforms the standard variations under the more general Equal-Dir. setting for both point cloud sizes, whereas for the Original-Dir. setting the standard variants perform better. This indicates to us that the improved performance of the two-path networks is from the proposed architecture, and not from a difference in point cloud sizes.

8. Conclusion

Exploded views are integral to effectively visualize complex assemblies, but creating them has been predominantly a manual, time-consuming process, as automated methods have faced challenges in terms of balancing speed and accuracy. Addressing this, we introduce AssemblyNet, a novel dataset containing point cloud data of 44 assemblies, divided into 495 subassemblies with a total of 5420 parts from a combination of industrial 3D models and LEGO models.

Table 7. **Effect of point cloud size.** We compare the performance in the classification setting when using a total point cloud size of 1024 and 2048 points, *i.e.*, 2×512 and 2×1024. We find that for both point cloud sizes the two-path networks achieves best performance in the more general Equal-Dir. setting.

Model	Original-Dir.		Equal-Dir.	
	1024	2048	1024	2048
DGCNN	72.36	72.31	55.76	57.01
PointNet++	84.02	79.49	60.88	64.14
TP-DGCNN	68.62	62.25	59.64	58.91
TP-PointNet++	63.92	79.19	65.85	65.13

The dataset is benchmarked using six different models, four state-of-the-art models for processing 3D point clouds as well as a two variations of a novel two-path network architecture using either DGCNN or PointNet++ as a backbone. This two-path network approach is designed to put an emphasis on the spatial relationship between a part and its subassembly. We present benchmarks for both classification and regression, using macro accuracy and angular similarity, respectively. For classification we consider both the original and an equalized distribution of directions, made possible by rotating each datapoint to all 26 directions. For regression we have also converted predictions and ground truth to the nearest of the 26 classification directions for a better comparison between the two approaches.

When evaluating using the more general equalized direction distribution we find that the best performing model, the TP-PointNet++, achieves an accuracy of 69.49%, outperforming the second best model, PointNet++, by nearly 7 percentage points. We find that a majority of incorrect predictions made by TP-PointNet++ are due to predicting opposite or orthogonal directions. These results clearly indicate that predicting part directions is a non-trivial task, which the current methods cannot handle sufficiently.

By open-sourcing AssemblyNet, the code, and the models, we aim to further the research and development in automated generation of exploded views, an important step in advancing technical illustrations, assembly instructions, and assembly visualization.

Acknowledgements: This work is supported by the Danish Innovation Foundation through its Industrial PhD program.

References

[1] Maneesh Agrawala, Doantam Phan, Julie Heiser, John Haymaker, Jeff Klingner, Pat Hanrahan, and Barbara Tversky. Designing effective step-by-step assembly instructions. *ACM Trans. Graph.*, 22(3):828–837, jul 2003. 2

[2] Zezi Ai, Kirstie Hawkey, and Stephen Brooks. Scale-based exploded views: A selection method for mobile devices. In *Proc. HotMobile*, page 27–32, New York, NY, USA, 2016. ACM. 1

[3] Iro Armeni, Ozan Sener, Amir R. Zamir, Helen Jiang, Ioannis Brilakis, Martin Fischer, and Silvio Savarese. 3d semantic parsing of large-scale indoor spaces. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016. 2

[4] M. V. A. Raju Bahubalendruni, B. B. Biswal, and B. B. V. L. Deepak. Optimal robotic assembly sequence generation using particle swarm optimization. In *Proceedings of the International Conference on Robotics and Artificial Intelligence (ICRAI 2015)*, May 2015. 2

[5] Xiaozhi Chen, Huimin Ma, Ji Wan, Bo Li, and Tian Xia. Multi-view 3d object detection network for autonomous driving. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. 2

[6] Young-Keun Choi, Dong Myung Lee, and Yeong Bin Cho. An approach to multi-criteria assembly sequence planning using genetic algorithms. *The International Journal of Advanced Manufacturing Technology*, 42:180–188, 2009. 2

[7] CloudCompare. Cloudcompare, 2009. [Online] Available at: https://www.cloudcompare.org/. 3

[8] Raul Diaz and Amit Marathe. Soft labels for ordinal regression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 7

[9] Jesper Gaarsdal, Sune Wolff, and Claus B. Madsen. Real-time exploded view animation authoring in vr based on simplified assembly sequence planning. In *2023 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW)*, pages 667–668, 2023. 1, 2, 3

[10] Ankit Goyal, Hei Law, Bowei Liu, Alejandro Newell, and Jia Deng. Revisiting point cloud shape classification with a simple and effective baseline. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 3809–3820. PMLR, 18–24 Jul 2021. 2, 5, 6, 7

[11] Shiyi Guo, Yujie Fu, Zhengda Qian, Zheng Rong, and Yihong Wu. Cross-attention-based feature extraction network for 3d point cloud registration. In *2022 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1–6, 2022. 4

[12] Yulan Guo, Hanyun Wang, Qingyong Hu, Hao Liu, Li Liu, and Mohammed Bennamoun. Deep learning for 3d point clouds: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(12):4338–4364, 2021. 2

[13] Joakim Bruslund Haurum, Moaaz M. J. Allahham, Mathias S. Lynge, Kasper Schøn Henriksen, Ivan A. Nikolov, and Thomas B. Moeslund. Sewer defect classification using synthetic point clouds. In *Proceedings of the 16th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - VISAPP*. INSTICC, SciTePress, 2021. 2

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016. 5

[15] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus), 2023. arXiv - 1606.08415. 5

[16] Ian Lenz, Honglak Lee, and Ashutosh Saxena. Deep learning for detecting robotic grasps. *The International Journal of Robotics Research*, 34(4-5):705–724, 2015. 2

[17] Wilmot Li, Maneesh Agrawala, Brian Curless, and David Salesin. Automated generation of interactive 3d exploded view diagrams. *ACM Trans. Graph.*, 27(3):1–7, aug 2008. 1, 2

[18] Daniel Maturana and Sebastian Scherer. 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, 2015. 2

[19] Seyed Saber Mohammadi, Yiming Wang, and Alessio Del Bue. Pointview-gcn: 3d shape classification with multi-view point clouds. In *2021 IEEE International Conference on Image Processing (ICIP)*, pages 3103–3107, 2021. 2

[20] Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. 2, 5

[21] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. 2, 5

[22] Gernot Riegler, Ali Osman Ulusoy, and Andreas Geiger. Octnet: Learning deep 3d representations at high resolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. 2

[23] Bruce Romney, Cyprien Godard, Michael Goldwasser, and G. Ramkumar. An efficient system for geometric assembly sequence generation and evaluation. In *ASME 1995 15th International Computers in Engineering Conference and the ASME 1995 9th Annual Engineering Database Symposium*, International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, pages 699–712, 09 1995. 2

[24] Shuai Shao, Yufei Xing, Ligang Qu, and Xin Li. An automatic generation method of exploded view based on projection. *Manufacturing Technology Journal*, 21(5):691–699, 2021. 1

[25] Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik Learned-Miller. Multi-view convolutional neural networks for 3d shape recognition. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, December 2015. 2

[26] Markus Tatzgern, Denis Kalkofen, and Dieter Schmalstieg. Dynamic compact visualizations for augmented reality. In *Proc. VR*, pages 3–6. IEEE Computer Society, 2013. 1

[27] Unity Technologies. Unity, 2005. [Online] Available at: https://unity.com/. 3

[28] Hwai-En Tseng, Chien-Cheng Chang, Shih-Chen Lee, and Yu-Ming Huang. Hybrid bidirectional ant colony optimization (hybrid baco): An algorithm for disassembly sequence planning. *Engineering Applications of Artificial Intelligence*, 83:45–56, 2019. 2

[29] Yuan-Jye Tseng, Jian-Yu Chen, and Feng-Yi Huang. A multi-plant assembly sequence planning model with integrated assembly sequence planning and plant assignment using ga. *The International Journal of Advanced Manufacturing Technology*, 48:333–345, 2010. 1

[30] S. S. V. Prasad Varupala, Anil Kumar Gulivindala, and M. V. A. Raju Bahubalendruni. An efficient geometrical feasibility testing method for exploded view generation to perform assembly sequence planning. In Saroj Kumar Acharya and Dipti Prasad Mishra, editors, *Current Advances in Mechanical Engineering*, pages 963–973. Springer Singapore, 2021. 2

[31] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. 2

[32] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon. Dynamic graph cnn for learning on point clouds. *ACM Trans. Graph.*, 38(5), oct 2019. 2, 4, 5

[33] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015. 2

[34] Li Yi, Vladimir G. Kim, Duygu Ceylan, I-Chao Shen, Mengyan Yan, Hao Su, Cewu Lu, Qixing Huang, Alla Sheffer, and Leonidas Guibas. A scalable active framework for region annotation in 3d shape collections. *ACM Trans. Graph.*, 35(6), dec 2016. 2

[35] Jiapeng Yu and Jiahao Zhang. Hierarchical exploded view generation based on recursive assembly sequence planning. *The International Journal of Advanced Manufacturing Technology*, 93(1):1207–1228, 2017. 2, 3

[36] Hanye Zhang, Haijiang Liu, and Lingyu Li. Research on a kind of assembly sequence planning based on immune algorithm and particle swarm optimization algorithm. *The International Journal of Advanced Manufacturing Technology*, 71:795–808, 2014. 1

[37] Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip H.S. Torr, and Vladlen Koltun. Point transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 16259–16268, October 2021. 2, 4