RGL-NET: A Recurrent Graph Learning framework for Progressive Part Assembly

Abhinav Narayan Harish
CVIG Lab, IIT Gandhinagar
abhinav.narayan@iitgn.ac.in

Rajendra Nagar
IIT Jodhpur
rn@iitj.ac.in

Shanmuganathan Raman
CVIG Lab, IIT Gandhinagar
shanmuga@iitgn.ac.in

Abstract

Autonomous assembly of objects is an essential task in robotics and 3D computer vision. It has been studied extensively in robotics as a problem of motion planning, actuator control and obstacle avoidance. However, the task of developing a generalized framework for assembly robust to structural variants remains relatively unexplored. In this work, we tackle this problem using a recurrent graph learning framework considering inter-part relations and the progressive update of the part pose. Our network can learn more plausible predictions of shape structure by accounting for priorly assembled parts. Compared to the current state-of-the-art, our network yields up to 10% improvement in part accuracy and up to 15% improvement in connectivity accuracy on the PartNet [23] dataset. Moreover, our resulting latent space facilitates exciting applications such as shape recovery from the point-cloud components. We conduct extensive experiments to justify our design choices and demonstrate the effectiveness of the proposed framework.

1. Introduction

Automated assembly requires a structural and functional understanding of object parts to place them in their appropriate locations. In a chair, a square-shaped structure could be its base or its back. A long cuboid part could be its legs. However, imparting this assembly skill to machines is still an open problem in vision and robotics.

To ensure smooth and collision-free assembly, we must accurately estimate the pose of each part. In robotics literature, there are a few works that attempt this problem. Choi *et al*. [6] develop a pose estimation scheme to register point cloud to incomplete depth maps. Suarez *et al*. [30] assemble an IKEA chair by hard-coding motion trajectories onto robotic arm manipulators. However, none of the prior works can be generalized to assembling everyday objects where we do not have access to the global structure of the assembled shape. In this work, we assemble a shape from its part point

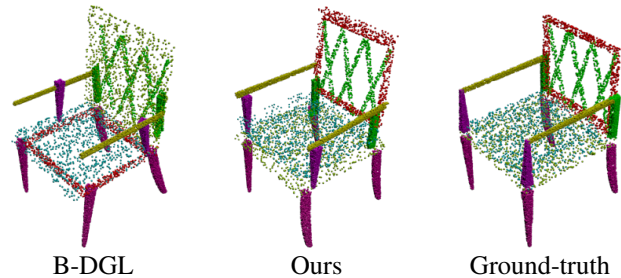B-DGL   Ours   Ground-truth

Figure 1: Our progressive Part Assembly scheme reduces inter-part confusion. Dynamic Graph Learning [11] (B-DGL) mistakes the green coloured chair seat for its back.

clouds without any prior semantic knowledge. Instead of manually configuring per-part pose, we explore relations that can be generalized across shapes in a category.

A study conducted in 2003 on designing assembly instructions [2] uncovers that humans prefer sequential assembly instructions - split into step-by-step instructions. This could be an assembly diagram illustrating how each part connects with the other or an instructional video. However, designing detailed diagrams can become cumbersome for the designer. In some instances, intricate designs are often unnecessary. Understanding the assembly progression can provide information of the subsequent part poses. This work demonstrates that a linear ordering of part elements can significantly improve part placement and inter-part connectivity.

Being a recently formulated research problem, only a few works tackle this problem in a similar setting as ours. Li *et al*. [19] assemble a shape from its component point cloud using an image-based prior. In Coalesce [37], the authors develop a framework for assembly and joint synthesis using translation and scaling of component point clouds. Huang *et al*. [11] attempt this task without semantic knowledge of parts using a dynamic graph learning framework.

However, none of these prior works have explored progressive assembly strategies. They transform all parts at once without leveraging information that previous part placements
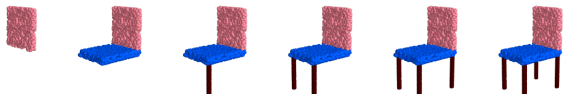
Figure 2: Top-down order for a chair in PartNet [23].

can offer. This can result in confusion among structurally similar components. For instance, a chair seat may have a very similar structure to the chair back, resulting in its incorrect placement (Figure 1). By incorporating an assembly progression (Figure 2), we reduce inter-part confusion and increase the network's capacity to learn intricate structures.

Similar to [11], we account for structural variety by incorporating random noise and allowing our network to explore the ground truth space using the minimum-over-N (MoN) [8] loss. Further, we analyze our network performance at various dimensions of random noise. Our analysis reveals that our framework can generalize well even at the zero randomness setting. Overall, our progressive scheme demonstrates up to 10% improvement in part accuracy and up to 15% improvement in connectivity accuracy over dynamic graph learning [11] on PartNet [23]. Moreover, our standalone framework can achieve up to 6% improvement over this baseline, demonstrating its efficacy. Our ablation studies address the critical aspects of our scheme, including the architectural design and the optimal order for part placement.

In summary, our major contributions are -

- We propose a novel recurrent graph learning framework for assembly which significantly improves part-placement and inter-part connectivity.

- Our framework yields competitive performance even in the absence of random exploration.

- We demonstrate qualitatively the potency of our latent space by utilizing it to recover shape without access to its global structure.

- We investigate a variety of ways of ordering part components, and experimentally establish the optimality of our choice.

2. Related Work

Part Based 3D Modelling. We can decompose complex 3D shapes into simple part structures, which can construct novel shapes. One of pioneering works in this direction was by Funkhouser *et al*. [9], who attempted this problem using an intelligent scissoring of parts components. The subsequent works [3, 16, 13] utilize probabilistic graphical models to encode semantic part relationships. The authors of [4] demonstrate the construction of high-quality CAD models using noisy data from sensors and a 3D shape database.

Recent works leverage the power of deep neural networks for shape-modelling. ComplementMe [31] introduces a weakly supervised approach in the absence of consistent semantic segmentation and labels. The authors of [7] create an autoencoder for a latent space to factorize a shape into its parts, allowing for part-level shape manipulation.

Most of the prior works in this domain either assume known part semantics or depend on an existing shape repository. We make no such assumption and assemble a variable number of parts during testing.

Structural Shape Generation. With the advent of deep-learning and the development of large scale shape datasets [23, 36], shape generation has garnered the interest of the vision community. GRASS [18] and StructureNet [21] compress shape structure into a latent space, taking into account inter-part relationships. PT2PC [22] generates 3D shapes conditioned on the part-tree decomposition. ShapeAssembly [15] uses a procedural programmatic representation for connecting part cuboids. SAGNET [34] develops a structural aware generative model, catering to pairwise relationships and encoding structure and geometry separately. SDM-NET [10] extends this approach to meshes through a controlled generation of fine-grained geometry.

Few of these prior works model shape generation as an assembly of point cloud components. Inspired by Seq2Seq networks for machine translation, PQ-NET [33] develops a sequential encoding and decoding scheme for regressing shape parameters. PageNet [17] utilizes a partwise-VAE to regress the transformation parameters of a 3D shape.

Instead of generating a new point cloud structure, we transform the existing point clouds of shape components using a rigid transformation. This is more relevant to real-world vision and robotics applications.

Part Component Assembly. Automated part assembly is a long-standing problem in robotics, emphasizing 6D pose estimation, motion planning and actuator control. Shao *et al*. [28] utilize fixtures to reduce the complexity of the assembly space. Zakka *et al*. [38] generalize assembly to unseen categories using shape descriptors. The authors of [20] utilize reinforcement learning to incorporate parameters like force and torque into assembly. Several other works formulate assembly as a motion planning problem [12, 14].

We tackle the problem closely aligned to computer vision, wherein we estimate the 6D pose from part point clouds without prior semantic knowledge. In this domain, [19, 11] formulate a similar problem to ours. Li *et al*. [19] utilize a two-stage pipeline of image segmentation followed by part assembly. The authors of [11] utilize a dynamic graph framework to assemble a shape. However, unlike these prior works, we incorporate progressive assembly to encode information, significantly improving part-placement.
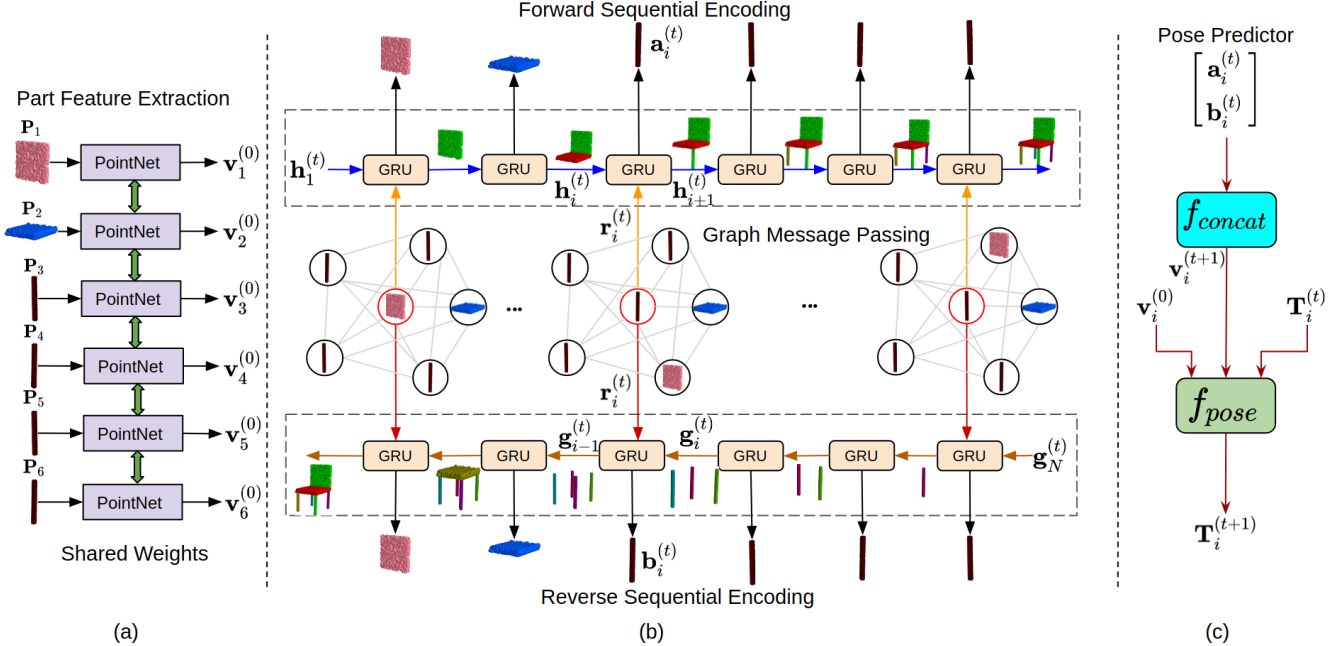
Figure 3: One iteration of our Recurrent Graph Learning framework. (a) We process part features and compute a graph message. (b) The message is encoded sequentially in our bidirectional GRU framework. (c) The features generated by the forward and reverse GRU are used to regress part-pose. We use three such iterations in our framework.

3. Proposed Method

Consider an ordered set of N non-overlapping point clouds components of a 3D shape, $\mathcal{P} = (\mathbf{P}_1, \mathbf{P}_2, \ldots, \mathbf{P}_N)$, where $\mathbf{P}_i \in \mathbb{R}^{N_d \times 3}$, and N_d, represents the number of points per 3D shape. We predict part poses $(\mathbf{q}_i, \mathbf{c}_i)$, where, $\mathbf{q}_i \in \mathbb{R}^4$ given $\|\mathbf{q}_i\|_2 = 1$ represents the quaternion and $\mathbf{c}_i \in \mathbb{R}^3$ represents the translation. The complete assembled shape is $\mathcal{S} = \mathbf{T}_1(\mathbf{P}_1) \cup \mathbf{T}_2(\mathbf{P}_2) \cup \cdots \cup \mathbf{T}_N(\mathbf{P}_N)$. Here, $\mathbf{T}_i(.)$ represents joint $SE(3)$ transformation arising from $(\mathbf{q}_i, \mathbf{c}_i)$.

To assemble a shape, we utilize an iterative network composed of a graph learning backbone [11] and a progressive assembly encoder. We call this framework Recurrent Graph Learning (RGL). The graph learning backbone accounts for inter-part relations to comprehend contextual information. Progressive encoding accumulates a prior using the shape structure of already assembled parts. We provide the complete pipeline of our framework in Figure 3.

3.1. Graph Learning Backbone

We model the inter-part relations, using a time-varying dynamic graph with set of vertices $\mathcal{V}^{(t)}$ and edges $\mathcal{E}^{(t)}$. The nodes of the graph $\mathcal{V}^{(t)} = \{\mathbf{v}_1^{(t)}, \mathbf{v}_2^{(t)}, \ldots, \mathbf{v}_N^{(t)}\}$ are the features of each part $\mathbf{P}_i$ at time step t of the iterative network. The graph is complete with a self-loop, i.e., $(i, j) \in \mathcal{E}^{(t)} \ \forall \ (i, j) \in [N] \times [N]$. Here, $[N]$ denotes the set of first N natural numbers $\{1, 2, \ldots, N\}$. We initialize the features $\mathbf{v}_i^{(0)} \in \mathbb{R}^{256}$ using a shared PointNet [26] encoder

on the point-cloud $\mathbf{P}_i$. At time step t, we model the edge message $\mathbf{e}_{ij}^{(t)} \in \mathbb{R}^{256}$ between the i-th and j-th nodes as,

$$\mathbf{e}_{ij}^{(t)} = f_{\text{edge}} \left(\begin{bmatrix} \mathbf{v}_i^{(t)} \\ \mathbf{v}_j^{(t)} \end{bmatrix} \right). \quad (1)$$

During assembly, distinct part-pairs may bear a different relationship. For instance, the four legs of a chair could be strongly dependent on each other and less influenced by the position of the chair back. To account for this, we use an attention mechanism [32]. Accordingly, we compute the overall message received by $\mathbf{v}_i^{(t)}$ as a weighted combination of edge messages from all possible nodes $\mathbf{v}_j^{(t)}$.

$$\mathbf{m}_i^{(t)} = \frac{\sum_{j=1}^{N} w_{ij}^{(t)} \mathbf{e}_{ij}^{(t)}}{\sum_{j=1}^{N} w_{ij}^{(t)}}. \quad (2)$$

Here, $w_{ij}^{(t)}$ represents the scalar attention weight between nodes $\mathbf{v}_i^{(t)}$ and $\mathbf{v}_j^{(t)}$. Among the many possible ways to compute attention, we observe that using features extracted from part poses $\mathbf{T}_i^{(t)}$ and $\mathbf{T}_j^{(t)}$ yield good results.

$$w_{ij}^{(t)} = f_{rel}(f_{feat}(\mathbf{T}_i^{(t)}), f_{feat}(\mathbf{T}_j^{(t)})), \ \forall t > 0. \quad (3)$$

Here, f_{feat} processes part-poses and returns a 128D feature. f_{rel} operates on these features to return the scalar w_{ij}. At the initial time step, $w_{ij}^{(0)} = 1$ and $\mathbf{T}_k^{(0)}(\mathbf{P}_k) = \mathbf{P}_k, \ \forall k \in [N]$.

3.2. Progressive Message Encoding

We identified two choices for progressive assembly - a) update the part features one at a time and use the updated features for relational reasoning with subsequent parts, b) storing the assembly information in a recurrent unit. We reject the first option because - i) we face the problem of vanishing and exploding gradients for parts occurring at the beginning of the sequence, ii) the parts at the end receive more supervision than the parts occurring at the beginning. Instead, we utilize a bidirectional gated recurrent unit (GRU) to store the prior assembly. This ensures smoother gradient flow. Moreover, its bidirectional nature distributes information fairly across the sequence.

We model the network by two ordered sets of hidden states $\mathcal{H}^{(t)} = \{\mathbf{h}_1^{(t)}, \mathbf{h}_2^{(t)}, \ldots, \mathbf{h}_N^{(t)}\}$ and $\mathcal{G}^{(t)} = \{\mathbf{g}_1^{(t)}, \mathbf{g}_2^{(t)}, \ldots, \mathbf{g}_N^{(t)}\}$ for the forward and backward recurrent units, respectively. Here, $\mathbf{h}_k^{(t)}, \mathbf{g}_k^{(t)} \in \mathbb{R}^{256}, \forall k \in [N]$. We allow our network to explore the ground truth space by encoding noise in the initial hidden state.

$$\mathbf{h}_1^{(t)} = \mathbf{g}_N^{(t)} = \begin{bmatrix} \mathbf{z}^\top & \mathbf{0}^\top \end{bmatrix}^\top. \qquad (4)$$

Here, $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ represents the random noise vector. We keep the initial forward and reverse hidden states the same so that both learn similar shape structures. While regressing the part pose of a current part $\mathbf{P}_i$, we take into account its current features and the received part message. The recurrent input, $\mathbf{r}_i^{(t)} = \begin{bmatrix} \mathbf{v}_i^{(t)} \\ \mathbf{m}_i^{(t)} \end{bmatrix}$ gives the network a context of the relative and absolute orientation of each part. We incorporate this information onto the prior using f_{hidden}.

$$\mathbf{h}_{i+1}^{(t)} = f_{hidden}(\mathbf{r}_i^{(t)}, \mathbf{h}_i^{(t)}) \qquad (5)$$

$$\mathbf{g}_{i-1}^{(t)} = f_{hidden}(\mathbf{r}_i^{(t)}, \mathbf{g}_i^{(t)}). \qquad (6)$$

Correspondingly, for each part we obtain two outputs, $\mathbf{a}_i^{(t)}$ and $\mathbf{b}_i^{(t)}$ through forward and reverse encoding, respectively.

$$\mathbf{a}_i^{(t)} = f_{out}(\mathbf{r}_i^{(t)}, \mathbf{h}_i^{(t)}) \qquad (7)$$

$$\mathbf{b}_i^{(t)} = f_{out}(\mathbf{r}_i^{(t)}, \mathbf{g}_i^{(t)}). \qquad (8)$$

We model the updated features $\mathbf{v}_i^{(t+1)}$ by processing $\mathbf{a}_i^{(t)}$, $\mathbf{b}_i^{(t)} \in \mathbb{R}^{256}$ using a function f_{concat}.

$$\mathbf{v}_i^{(t+1)} = f_{concat}\left(\begin{bmatrix} \mathbf{a}_i^{(t)} \\ \mathbf{b}_i^{(t)} \end{bmatrix}\right). \qquad (9)$$

This step aims to reduce the bias occurring due to part location in the sequence; parts appearing at the beginning of the first sequence would occur at the end of the second and vice-versa. Using these updated features, we can regress the pose for each part. We also utilize the original features $\mathbf{v}_i^{(0)}$ and previously extracted part-pose $\mathbf{T}_i^{(t)}$, to pass on information extracted in previous time-steps.

$$\mathbf{T}_i^{(t+1)} = f_{pose}(\mathbf{v}_i^{(t+1)}, \mathbf{v}_i^{(0)}, \mathbf{T}_i^{(t)}). \qquad (10)$$

In our implementation, f_{out} and f_{hidden} are the transfer functions of the GRU block. f_{rel}, f_{edge}, f_{feat}, f_{pose} and f_{concat} are Multi-Layer-Perceptron's (MLP's). Overall, we utilize three time steps of the RGL framework.

4. Experiments

In this section, we demonstrate the merits of our sequential strategy through a variety of experiments. We also justify our design choices through extensive ablation studies.

4.1. Dataset

Due to the unavailability of a large scale real-world dataset for this task, we utilize the synthetic PartNet [23] dataset containing fine-grained part segmentation. We use the three largest categories - i) chair, ii) table and iii) lamp with the predefined train (70%), validation (10%) and test (20%) splits. Each component part contains 1000 points, sampled from part meshes using farthest point sampling. To ensure invariance to rigid transformations, we transform each point cloud into its canonical space using PCA [25].

4.2. Loss Functions

To explore structural variations, we incorporate the MoN loss [8], along with random noise $\mathbf{z}_j$ in the initial hidden state. Considering our overall network as f and the optimal pose-extractor as f^*, we define the MoN loss as,

$$\mathcal{L}_{mon} = \min_{j \in [N]} \mathcal{L}\left(f\left(\mathcal{P}, \mathbf{z}_j\right), f^*(\mathcal{P})\right). \qquad (11)$$

Here, $\mathbf{z}_j \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \forall j \in [N]$, are IID random noise vectors. The loss function, $\mathcal{L}$, is split into three categories similar to [11] for global and part-wise structural integrity.

The translation is supervised by an L2 loss $\mathcal{L}_t$ between the predicted and ground-truth part center ($\mathbf{c}_i^*$, $\mathbf{c}_i^*$).

$$\mathcal{L}_t = \sum_{i=1}^{N} \|\mathbf{c}_i - \mathbf{c}_i^*\|_2^2. \qquad (12)$$

The rotation is supervised by calculating Chamfer distance [8] between the rotated point cloud $\mathbf{q}_i(\mathbf{P}_i)$ and the ground-truth point cloud $\mathbf{q}_i^*(\mathbf{P}_i)$ (Equation (13)).

$$\mathcal{L}_r = \sum_{i=1}^{N} d_c(\mathbf{q}_i(\mathbf{P}_i), \mathbf{q}_i^*(\mathbf{P}_i)). \qquad (13)$$

		B-Global [27, 17]	B-LSTM [33]	B-Complement [31]	B-DGL [11]	Ours without MoN	Ours (Complete)
SCD↓	Chair	0.0146	0.0131	0.0241	**0.0091**	0.0101	**0.0087**
	Table	0.0112	0.0125	0.0298	**0.0050**	0.0053	**0.0048**
	Lamp	0.0079	**0.0077**	0.0150	0.0093	0.0088	**0.0072**
PA↑	Chair	15.70	21.77	8.78	39.00	**42.84**	**49.06**
	Table	15.37	28.64	2.32	**49.51**	49.15	**54.16**
	Lamp	22.61	20.78	12.67	**33.33**	31.66	**37.56**
CA↑	Chair	9.90	6.80	9.19	23.87	**28.74**	**32.26**
	Table	33.84	22.56	15.57	**39.96**	39.71	**42.15**
	Lamp	18.60	14.05	26.56	41.70	**46.28**	**57.34**

Table 1: Quantitative comparison with baseline methods. **Green** represents the best and **Blue** represents the second best.

Here, $d_c(\mathcal{X}, \mathcal{Y})$ is the Chamfer distance between the two point sets $\mathcal{X}$ and $\mathcal{Y}$, defined in Equation (14).

$$d_c(\mathcal{X}, \mathcal{Y}) = \sum_{\mathbf{x} \in \mathcal{X}} \min_{\mathbf{y} \in \mathcal{Y}} \|\mathbf{x} - \mathbf{y}\|_2^2 + \sum_{\mathbf{y} \in \mathcal{Y}} \min_{\mathbf{x} \in \mathcal{X}} \|\mathbf{x} - \mathbf{y}\|_2^2. \quad (14)$$

Lastly, the shape-cd-loss $\mathcal{L}_s$ (Equation (15)), ensures the overall quality of the generated assembly $\mathcal{S}$ by computing its Chamfer distance from the ground truth assembly $\mathcal{S}^*$.

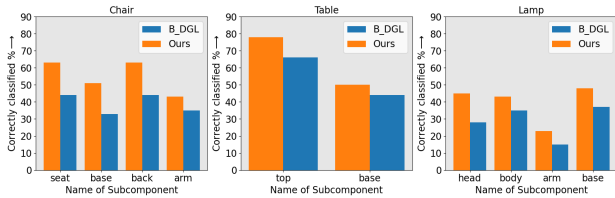$$\mathcal{L}_s = d_c(\mathcal{S}, \mathcal{S}^*). \quad (15)$$

Figure 4: Comparison of our method with B-DGL [11] on the most common sub-components of each category.

4.3. Evaluation Metrics

We measure the network performance by generating a variety of shapes and finding the closest shape to the ground truth using minimum matching distance [1]. For better comparison, we utilize *part accuracy* (PA) , *connectivity accuracy* (CA) and *shape Chamfer distance* (SCD), used by [11]. *Shape Chamfer distance* is defined in Equation (15). We define the remaining terms below.

Part Accuracy. This metric (Equation (16)) measures the fraction of $SE(3)$ transformed parts $\mathbf{T}_i(\mathbf{P}_i)$ that lie below a threshold Chamfer distance τ_p from the ground truth $\mathbf{T}_i^*(\mathbf{P}_i)$. Here, $\mathbb{1}$ represents the indicator function.

$$PA = \frac{1}{N} \sum_{i=1}^{N} \mathbb{1}\big(d_c(\mathbf{T}_i(\mathbf{P}_i), \mathbf{T}_i^*(\mathbf{P}_i)) < \tau_p\big). \quad (16)$$

Connectivity Accuracy. We incorporate connectivity accuracy (Equation (17)), to measure the quality of inter-part connections. For each connected-part pair $(\mathbf{P}_i, \mathbf{P}_j)$, we define the contact c_{ij}^* as a point on $\mathbf{P}_i$ that is closest to $\mathbf{P}_j$. Similarly, contact point c_{ji}^* is the point on $\mathbf{P}_j$ that is closest to $\mathbf{P}_i$. (c_{ij}^*, c_{ji}^*) are transformed into their corresponding part canonical space as (c_{ij}, c_{ji}). Then, connectivity accuracy is,

$$CA = \frac{1}{|\mathcal{C}|} \sum_{\{c_{ij}, c_{ji}\} \in \mathcal{C}} \mathbb{1}\left(\|\mathbf{T}_i(c_{ij}) - \mathbf{T}_j(c_{ji})\|_2^2 < \tau_c\right). \quad (17)$$

Here, $\mathcal{C}$ represents the set of all possible contact point pairs $\{c_{ij}, c_{ji}\}$. During evaluation, τ_c and τ_p are set to 0.01.

4.4. Results and Comparisons

The only direct baseline to our work is Dynamic Graph Learning (B-DGL) [11]. We also compare our results with three other baselines: B-LSTM [33], B-Global [17, 27] and B-Complement [31] used by B-DGL. As we were unable to reproduce the results of B-Complement accurately, we exclude it from our qualitative comparison.

In Table 1, we observe that the most improvement in part accuracy($\approx 10\%$) occurs in the chair category. This could be due to four distinct components of the chair - back, seat, leg and arm, which merits our progressive assembly framework. The improvement is $\approx 4.5\%$ on the table category, which has only two such distinct components - table-top and table-base. On the lamp category, progressive assembly helps to ensure better connectivity accuracy, which is 15% above B-DGL.

Figure 4 shows that the improvement is distributed across the most common subcategories of a shape. Among these, the chair-seat, chair-back and table-top are well-performing subcategories. On the other hand, structurally diverse components like the chair arm and table base have lower accuracy's.

Our qualitative results reflect a few key aspects which our progressive framework improves .We further highlight these qualitative results in our supplementary video.

Inter-Part Confusion. In Figure 5(a), we observe that the chair arm and leg share a very similar structure. B-
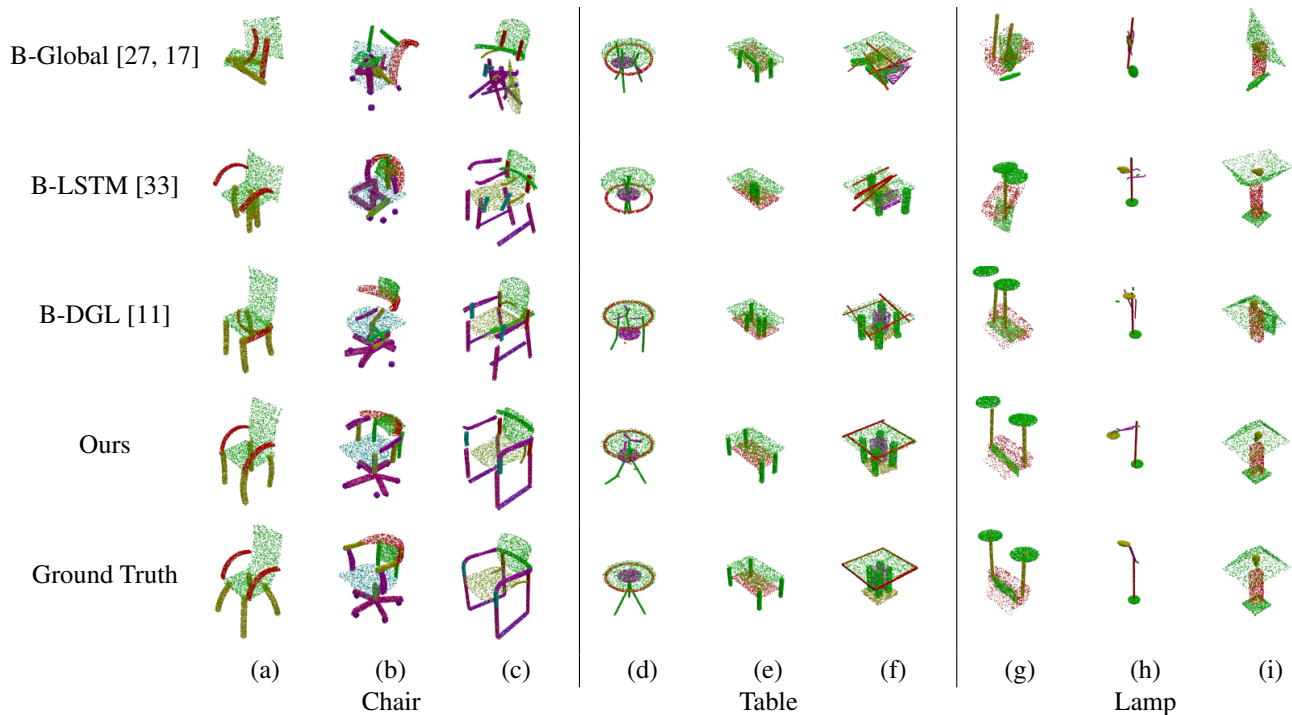
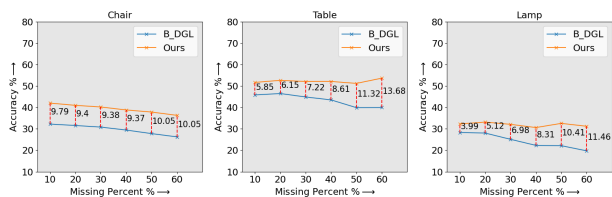Figure 5: Qualitative comparison with baseline methods on 9 different shapes, (a)-(i) of PartNet [23].

Figure 6: Comparison of our method with B-DGL [11] with varying percentage of missing parts.

DGL misinterprets the unconventionally structured red chair handles. Our framework is able to correctly place this part.

Better Connectivity. Our method better understands fine-grained part connections. In Figure 5(f), our network is able to correctly predict the four bars around the table-top. In the lamp in Figure 5(i), our network is able to predict the light bulb and its cover correctly.

Rotational Symmetry. Predicting rotational symmetry is a challenging task which our network handles very well. In Figure 5(b) the star shaped chair legs are correctly predicted.

4.5. Performance with Missing Parts

Often, a packaging defect can result in missing parts during assembly. In this scenario, we want our algorithm to predict plausible results so the deficiency can be identified.

This is not without a few caveats. By randomly choosing a candidate for deletion, pivotal parts like the chair-seat could be removed, affecting the quality of assembly. Instead, we order parts according to their volume and delete a certain percentage of the smallest volume parts. We utilize this strategy as smaller parts are more likely to be misplaced.

Figure 7: Sample result with missing chair legs. Notice how our method approximately positions the curved back.

For part deletion, we compute the volume of the axis-aligned-bounding-box of the point cloud. Further, each part belonging to a part-group (ex:-chair legs) is assigned the minimum volume among all its members. This accounts for point-cloud sampling variations. Moreover, we do not use partial part-groups. For instance, if we delete one chair leg, the rest of the chair legs are also deleted.

In Figure 6, we observe that the accuracy increases at specific deletion percentages. This could be due to the removal of incorrectly placed smaller volume parts. The increasing

difference with the baseline B-DGL shows that our algorithm is more robust at higher deletion percentages. In Figure 7, we provide a sample visual result on a chair at 60% deletion.

4.6. Shape Recovery from Latent Space

An exciting application of our latent space is shape recovery. Unlike shape-autoencoding [35, 5, 24], we do not have access to the global shape structure. Instead, we recover shape structure from the component point clouds. We utilize the point cloud decoder of TreeGAN [29] and train it without the discriminator separately on the two largest categories, chair and table, using the last hidden state of our GRU. During training, we optimize *shape Chamfer distance* (Equation (15)) and train it independently of assembly.

In Figure 8, we observe that for the four-leg table, our recovery maintains structural integrity. On the chair, our network gives a good outline of the structure, with a sparser distribution of points around the chair legs. This could be due to storing a detailed shape in a smaller dimension.

This experiment gives an insight into our progressive scheme. Our latent state carries coarse structure information required in the subsequent steps. The reconstruction is reasonably accurate given the hidden state has not been constrained during assembly and the diversity of PartNet [23].
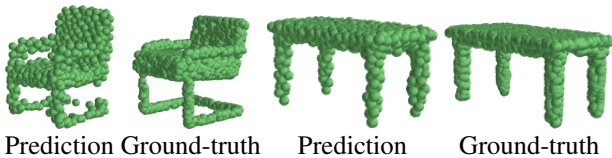

Prediction   Ground-truth   Prediction   Ground-truth

Figure 8: Results on shape recovery from hidden state. Our method can recover coarse structure of the shape.

4.7. Bounds of Performance

Introducing random noise to our network architecture allows us to generate structural variations. In this section, we monitor the performance of our network to varying amounts of randomness and establish a bound on part accuracy. We do not modify the network architecture but change the dimension of random noise in Equation (4), keeping the dimension of the hidden state $\mathbf{h}_1^{(t)}$ and $\mathbf{g}_N^{(t)}$ fixed.

To better quantify the variations, we introduce the term variability V_E as the difference between its best and worst performance. Analogous to Equation (11), we define maximum matching distance as the network's worst performance over E iterations. Then, considering our network as a function, f, the ground truth as f^*, and random noise as $\mathbf{z}_j$,

$$V_E = \max_{j \in [E]} \mathcal{L}\left(f\left(\mathcal{P}, \mathbf{z}_j\right), f^*(\mathcal{P})\right) - \min_{j \in [E]} \mathcal{L}\left(f\left(\mathcal{P}, \mathbf{z}_j\right), f^*(\mathcal{P})\right). \tag{18}$$
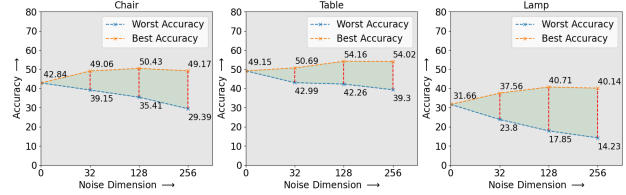

Figure 9: Performance of our network on varying dimension of random noise. Our results at zero noise are comparable to B-DGL [11] (Chair: 39.00, Table: 49.51, Lamp: 33.33).

To replicate a practical scenario, we choose $E = 10$ and experiment with noise dimensions of 0, 32, 128 and 256.

Our results in Figure 9 demonstrate that increasing the random noise allows the network to explore more structural varieties; however, it results in a decreasing infimum. Also, at a given noise dimension, the lamp category shows the highest variability. This could be attributed to its structural diversity and smaller dataset size.

We customize our network design choices based on this analysis. For optimal performance, our network must balance accuracy ($PA \uparrow$) and variability ($V_E \downarrow$). Accordingly, we choose the noise dimension as 32 for the chair and lamp category and 128 for the table category.

Performance in Absence of Random Noise. Incorporating MoN [8] loss during training explores structural varieties, leading to better overall performance. However, it comes at the cost of increased training time and variability (V_E). Figure 9 reflects an additional benefit of our progressive scheme; our results are competitive even in the absence of random noise. In this setting, our network can be trained without the MoN loss, which is $\times 2.5$ faster and has no tradeoff on variability ($V_E = 0$). In Table 1, we note that these results are comparable to B-DGL trained with 5 iterations of MoN.

4.8. Ablation Studies

In this section, we provide an experimental justification of our design choices. In particular, we consider two major aspects - i) structural variations of our architecture and ii) optimal sequence for assembly. We provide details of each configuration in our supplementary file.

Architectural Variants. We construct a diverse set of architecture variants to justify our design choices. We use a unidirectional RNN in both i) bottom to top and ii) top to bottom ordering, iii) we initialize the subsequent hidden state, $(\mathbf{h}_1^{(t+1)} = \mathbf{h}_N^{(t)}$ and $\mathbf{g}_N^{(t+1)} = \mathbf{g}_1^{(t)})$, iv) we add noise to the pose decoder instead of the hidden state, v) we evaluate our recurrent backbone without graph learning, and vi) we pass the precomputed graph-message after sequential encoding.

In Table 2, we observe that the bidirectional GRU incorporates more context compared to its unidirectional

	SCD ↓	PA ↑	CA ↑
(i) Bottom to Top Encoding	**0.0086**	46.42	29.66
(ii) Top to Bottom Encoding	0.0101	44.81	28.85
(iii) Initialize hidden states	0.0095	46.74	29.60
(iv) Noise in Pose Decoder	0.0098	46.31	31.19
(v) Without Graph Learning	0.0092	45.36	31.78
(vi) Sequential before Graph	0.0091	48.13	30.54
(vii) Ours (Complete)	0.0087	**49.06**	**32.26**

Table 2: Ablation study of structural variants.

	SCD ↓	PA ↑	CA ↑
(i) Volume order	0.0119	36.13	22.01
(ii) Group Connectivity Order	0.0118	36.62	22.07
(iii) Part Connectivity Order	0.0114	37.46	25.19
(iv) Central - Part Connectivity	0.0102	43.04	28.65
(v) Random order	0.0158	30.91	19.04
(vi) Top to Bottom Order	**0.0087**	**49.06**	**32.26**

Table 4: Ablation study of the different orders used for assembling a shape.

counterpart. Using bottom-to-top encoding performs better ($PA = 46.42$) than top-to-bottom ($PA = 44.81$) encoding. This could be because the chair legs are closer to the seat, and fixing the seat earlier in the sequence can better predict the chair arm and back.

Our standalone framework can predict parts better ($PA = 45.36$) than the B-DGL ($PA = 39.00$), highlighting the merits of progressive assembly. It is noteworthy to observe that initializing hidden states of the subsequent time-steps $t > 1$ negatively impacts part accuracy ($PA = 46.74$). This could be because using random noise at each step better explores structural variations than initializing them with the previous hidden state. Also, exploring global structural variations by introducing the noise in the hidden state ($PA = 49.06$) results in better performance than part-wise randomness, i.e, placing noise in the pose-decoder ($PA = 46.31$).

We also analyze the importance of different loss functions by removing each separately and training with the remaining losses. In Table 3, we observe that $\mathcal{L}_t$ is the most significant for accurate part placement. Among the remaining losses, $\mathcal{L}_r$ helps improve connectivity between parts (CA), and $\mathcal{L}_s$ helps optimize the overall shape structure (SCD).

	SCD ↓	PA ↑	CA ↑
(i) Without $\mathcal{L}_s$	0.0098	48.62	30.85
(ii) Without $\mathcal{L}_t$	0.0091	16.35	14.21
(iii) Without $\mathcal{L}_r$	**0.0078**	48.72	29.85
(iv) Ours (Complete)	0.0087	**49.06**	**32.26**

Table 3: Removing individual loss functions.

Optimal Order for Assembly. As our assembly strategy is progressive, studying the interplay between ordering and the resulting part placement is crucial. However, the number of possible arrangements grows exponentially with the number of parts. Theoretically, there could exist an order which produces better assembly results than ours. Identifying this global optimum ordering is beyond the scope of this experiment. Instead, we consider a few intuitive choices and determine the best one among those - i) we consider volume ordering, i.e., parts ordered from minimum to maximum vol-

ume, ii) we group similar parts together, start from a random group and iteratively append neighbouring groups, iii) we start from a random part and iteratively append neighbouring parts (part-connectivity), iv) we follow part-connectivity, however, beginning at the part with maximum neighbours, v) and lastly, we evaluate random ordering.

The results in Table 4 show that among our considered choices, the top-down ordering ($CA = 32.26$) of parts is the most optimal and random arrangement performs the worst ($CA = 19.04$). Among the other choices, part connectivity ensures better connectivity ($CA = 25.19$) compared to group-wise ordering ($CA = 22.07$). Moreover, starting from the most connected part further improves connectivity accuracy ($CA = 28.65$). However, there is not much difference in following volume ordering ($CA = 22.01$) and group connectivity ordering ($CA = 22.07$).

5. Conclusion and Future Work

We proposed a novel progressive approach to assemble shapes given their part point clouds which can better predict part locations and inter-part connectivity. We showed the potency of our latent space by utilizing it to recover shape structure. Furthermore, our experiments demonstrated that part ordering could play a crucial role in assembly automation. Future works may develop a reordering framework to arrange randomly-ordered parts into a consistent top to bottom order. Another possible study could explore the application of our latent space in retrieving part-connectivity.

We would also like to incorporate constraints such as symmetry into our progressive strategy. In a chair, we could assemble its back, followed by its seat and legs ensuring global shape symmetry constraints at each step. This would reduce the dimensionality of the assembly space.

6. Acknowledgements

References

[1] Panos Achlioptas, Olga Diamanti, Ioannis Mitliagkas, and Leonidas Guibas. Learning representations and generative models for 3d point clouds. *International conference on machine learning*, pages 40–49, 2018.

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

[3] Siddhartha Chaudhuri, Evangelos Kalogerakis, Leonidas Guibas, and Vladlen Koltun. Probabilistic reasoning for assembly-based 3d modeling. In *ACM SIGGRAPH 2011 papers*, pages 1–10. 2011.

[4] Siddhartha Chaudhuri and Vladlen Koltun. Data-driven suggestions for creativity support in 3d modeling. *ACM SIGGRAPH Asia 2010 papers*, pages 1–10, 2010.

[5] Zhiqin Chen, Kangxue Yin, Matthew Fisher, Siddhartha Chaudhuri, and Hao Zhang. Bae-net: Branched autoencoder for shape co-segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8490–8499, 2019.

[6] Changhyun Choi, Yuichi Taguchi, Oncel Tuzel, Ming-Yu Liu, and Srikumar Ramalingam. Voting-based pose estimation for robotic assembly using a 3d sensor. *2012 IEEE International Conference on Robotics and Automation*, pages 1724–1731, 2012.

[7] Anastasia Dubrovina, Fei Xia, Panos Achlioptas, Mira Shalah, Raphaël Groscot, and Leonidas J Guibas. Composite shape modeling via latent space factorization. *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8140–8149, 2019.

[8] Haoqiang Fan, Hao Su, and Leonidas J Guibas. A point set generation network for 3d object reconstruction from a single image. *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 605–613, 2017.

[9] Thomas Funkhouser, Michael Kazhdan, Philip Shilane, Patrick Min, William Kiefer, Ayellet Tal, Szymon Rusinkiewicz, and David Dobkin. Modeling by example. *ACM transactions on graphics (TOG)*, 23(3):652–663, 2004.

[10] Lin Gao, Jie Yang, Tong Wu, Yu-Jie Yuan, Hongbo Fu, Yu-Kun Lai, and Hao Zhang. Sdm-net: Deep generative network for structured deformable mesh. *ACM Transactions on Graphics (TOG)*, 38(6):1–15, 2019.

[11] Jialei Huang, Guanqi Zhan, Qingnan Fan, Kaichun Mo, Lin Shao, Baoquan Chen, Leonidas Guibas, and Hao Dong. Generative 3d part assembly via dynamic graph learning. *The IEEE Conference on Neural Information Processing Systems (NeurIPS)*, 2020.

[12] Seth A Hutchinson and Avinash C Kak. Extending the classical ai planning paradigm to robotic assembly planning. *Proceedings., IEEE International Conference on Robotics and Automation*, pages 182–189, 1990.

[13] Prakhar Jaiswal, Jinmiao Huang, and Rahul Rai. Assembly-based conceptual 3d modeling with unlabeled components using probabilistic factor graph. *Computer-Aided Design*, 74:45–54, 2016.

[14] Pablo Jiménez. Survey on assembly sequencing: a combinatorial and geometrical perspective. *Journal of Intelligent Manufacturing*, 24(2):235–250, 2013.

[15] R Kenny Jones, Theresa Barton, Xianghao Xu, Kai Wang, Ellen Jiang, Paul Guerrero, Niloy J Mitra, and Daniel Ritchie. Shapeassembly: Learning to generate programs for 3d shape structure synthesis. *ACM Transactions on Graphics (TOG)*, 39(6):1–20, 2020.

[16] Evangelos Kalogerakis, Siddhartha Chaudhuri, Daphne Koller, and Vladlen Koltun. A probabilistic model for component-based shape synthesis. *ACM Transactions on Graphics (TOG)*, 31(4):1–11, 2012.

[17] Jun Li, Chengjie Niu, and Kai Xu. Learning part generation and assembly for structure-aware shape synthesis. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(07):11362–11369, 2020.

[18] Jun Li, Kai Xu, Siddhartha Chaudhuri, Ersin Yumer, Hao Zhang, and Leonidas Guibas. Grass: Generative recursive autoencoders for shape structures. *ACM Transactions on Graphics (TOG)*, 36(4):1–14, 2017.

[19] Yichen Li, Kaichun Mo, Lin Shao, Minhyuk Sung, and Leonidas Guibas. Learning 3d part assembly from a single image. In *European Conference on Computer Vision*, pages 664–682. Springer, 2020.

[20] Jianlan Luo, Eugen Solowjow, Chengtao Wen, Juan Aparicio Ojea, Alice M Agogino, Aviv Tamar, and Pieter Abbeel. Reinforcement learning on variable impedance controller for high-precision robotic assembly. *2019 International Conference on Robotics and Automation (ICRA)*, pages 3080–3087, 2019.

[21] Kaichun Mo, Paul Guerrero, Li Yi, Hao Su, Peter Wonka, Niloy Mitra, and Leonidas J Guibas. Structurenet: Hierarchical graph networks for 3d shape generation. *arXiv preprint arXiv:1908.00575*, 2019.

[22] Kaichun Mo, He Wang, Xinchen Yan, and Leonidas Guibas. Pt2pc: Learning to generate 3d point cloud shapes from part tree conditions. In *European Conference on Computer Vision*, pages 683–701. Springer, 2020.

[23] Kaichun Mo, Shilin Zhu, Angel X Chang, Li Yi, Subarna Tripathi, Leonidas J Guibas, and Hao Su. Partnet: A large-scale benchmark for fine-grained and hierarchical part-level 3d object understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 909–918, 2019.

[24] Jiahao Pang, Duanshun Li, and Dong Tian. Tearingnet: Point cloud autoencoder to learn topology-friendly representations. *arXiv preprint arXiv:2006.10187*, 2020.

[25] Karl Pearson. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572, 1901.

[26] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017.

[27] Nadav Schor, Oren Katzir, Hao Zhang, and Daniel Cohen-Or. Componet: Learning to generate the unseen by part synthesis

and composition. *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8759–8768, 2019.

[28] Lin Shao, Toki Migimatsu, and Jeannette Bohg. Learning to scaffold the development of robotic manipulation skills. *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5671–5677, 2020.

[29] Dong Wook Shu, Sung Woo Park, and Junseok Kwon. 3d point cloud generative adversarial network based on tree structured graph convolutions. *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3859–3868, 2019.

[30] Francisco Suárez-Ruiz, Xian Zhou, and Quang-Cuong Pham. Can robots assemble an ikea chair? *Science Robotics*, 3(17):eaat6385, 2018.

[31] Minhyuk Sung, Hao Su, Vladimir G Kim, Siddhartha Chaudhuri, and Leonidas Guibas. Complementme: Weakly-supervised component suggestions for 3d modeling. *ACM Transactions on Graphics (TOG)*, 36(6):1–12, 2017.

[32] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. pages 5998–6008, 2017.

[33] Rundi Wu, Yixin Zhuang, Kai Xu, Hao Zhang, and Baoquan Chen. Pq-net: A generative part seq2seq network for 3d shapes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 829–838, 2020.

[34] Zhijie Wu, Xiang Wang, Di Lin, Dani Lischinski, Daniel Cohen-Or, and Hui Huang. Sagnet: Structure-aware generative network for 3d-shape modeling. *ACM Transactions on Graphics (TOG)*, 38(4):1–14, 2019.

[35] Yaoqing Yang, Chen Feng, Yiru Shen, and Dong Tian. Foldingnet: Point cloud auto-encoder via deep grid deformation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 206–215, 2018.

[36] 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 Transactions on Graphics (ToG)*, 35(6):1–12, 2016.

[37] Kangxue Yin, Zhiqin Chen, Siddhartha Chaudhuri, Matthew Fisher, Vladimir Kim, and Hao Zhang. Coalesce: Component assembly by learning to synthesize connections. *arXiv preprint arXiv:2008.01936*, 2020.

[38] Kevin Zakka, Andy Zeng, Johnny Lee, and Shuran Song. Form2fit: Learning shape priors for generalizable assembly from disassembly. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9404–9410. IEEE, 2020.