BIGCHAIR: BImodal Graph-CHAracter learnIng for Retrieval

Stanford CS224N Custom Project

Sithipont (Pino) Cholsaipant
Department of Computer Science
Stanford University
pschol@stanford.edu

Sidney Hough
Department of Computer Science
Stanford University
shough@stanford.edu

Julian Quevedo
Department of Computer Science
Stanford University
julianq@stanford.edu

Abstract

Language models that encode structural information about 3D shapes may facilitate efficient and expressive retrieval of complex 3D information and aid humans in the model design process. Language-shape fusion can be achieved with a joint embedding space that enforces small distances between embeddings of descriptions generated by a text encoder and matching shape embeddings generated by a shape encoder. Approaches that voxelize 3D shapes and encode them with various convolutional neural networks suffer from information loss and wasted computation by applying the same convolution over voxels of various densities. As such, we pass raw 3D shape data in the form of meshes through a graph neural network (GNN). We additionally improve the text encoder by augmenting description data with aspect-aware embeddings that capture the most critical properties of 3D shapes.

1 Key Information to include

- Mentor: Jiajun Wu
- External Collaborators (if you have any): No
- Sharing project: No

2 Introduction

Artificial intelligence has the potential to augment human creative facilities. 3D modeling is one creative domain that is relatively unexplored by machine learning research, but stands to gain immensely from intelligent assistance. Modeling software is notoriously difficult to learn and tedious to use. Model databases rely on fine-grained human annotation for search.

If someone describes an idea for an interesting new chair, any designer worth their salt should be able to visualize said chair in their mind's eye, as well as apply transformations on a whim: rotate it, squish it, or change its color. Similarly, machine learning models should be able to leverage natural language to reason about 3D objects. One means of multimodal synthesis is via the development of a joint embedding space with contrastive learning that encodes semantics from multiple modalities. Contrastive learning gets around the need to embark on massive data annotation efforts, instead having models learn directly from natural language descriptions [1]. Joint embedding spaces have proven particularly effective in the translation of images to text, as in CLIP [2].

Limited work has applied contrastive learning to joint text-3D-shape learning [3]. Thus, in this paper, we learn joint representations for natural language descriptions and 3D shapes with a shape encoder and a separate text encoder. We improve upon existing approaches by encoding shapes as raw meshes instead of voxelizing them, which results in improved fidelity of shape inputs. We additionally help our text encoder by augmenting text inputs with context-aware descriptions that capture key geometric information. To test the quality of our embedding space, we focus on quantitative and qualitative retrieval performance, and observe that our mesh encoder is capable of expressing a wide variety of 3D shape properties as described in text.

3 Related Work

3.1 3D Representations for Machine Learning

3D shapes can be represented in a variety of ways, including voxels and point clouds [4]. Deep learning for 3D shapes has traditionally leveraged voxel-based representations, which are 3D extensions of pixels. For instance, 3D ShapeNets [5] extend convolutional deep belief networks to voxels, representing shapes as probability distributions of binary variables on a 3D voxel grid. VoxNet [6] is a 3D-CNN architecture that learns over occupancy grids. To enforce rotational invariance, VoxNet augments their data with rotations of the same shape and pools predictions for all rotations at test time.

Point clouds are a relatively less structured, unordered means of representing shapes. An early work with point clouds is PointNet [7], which applies a symmetric function on elements of an orderless point set containing (x, y, z) coordinates and transforms inputs into a canonical features space to accomplish transformation invariance. PointCNN is an approach that learns a convolutional neural network over point clouds using special $\mathcal{X}$-Conv operators that learn a transformation designed to weight and permute input features, with the goal of achieving equivariance to different point orders [8].

Voxel-based methods suffer from high computational costs and memory requirements, in part due to the fact that variable resolution is not possible. Applying the same convolution operation over voxels representing heterogeneous surfaces can lead to information loss or wasted computation [9]. Point-based methods, on the other hand, lack point connectivity, making nearest neighbor search expensive. To combat the challenges inherent in both of these methods, our work leverages mesh-based representations. We convert raw shape meshes into graphs and learn variants of graph convolutional networks over them, allowing for arbitrary resolution, efficient vertex sampling, guaranteed transformation invariance, and faithfulness to the raw shapes being represented. For example, highly planar sections of a surface with little curvature can be modelled with few, sparsely distributed graph nodes, while sections with high curvature can be represented precisely with dense subgraphs.

3.2 Graph Learning

Graph machine learning is an emerging field that focuses on connected data such as social networks. Graph neural networks (GNNs) [10] update node representations over several layers. Each node defines its own computation graph, aggregating and transforming messages from its neighbors in convolutional layers. Importantly, GNNs are always either invariant or equivariant to node orderings, so the final representation for a mesh will be fixed regardless of vertex ordering. This means shapes that are rotated, scaled, or otherwise transformed will be one and the same to a GNN if represented by an orderless mesh.

We base our approaches on a type of GNN known as the Graph Attention Network (GAT) [11]. GAT layers computes attention over node features, effectively allowing nodes to assign weight to more important neighbors. The following equation shows an update function for a node h_i, using attention coefficients α to compute linear combinations of neighbor node features, aggregating these neighbors and applying a nonlinearity σ.

$$h_i' = \sigma(\sum_{j \in \mathcal{N}(i)} \alpha_{ij} \mathbf{W} h_j) \tag{1}$$

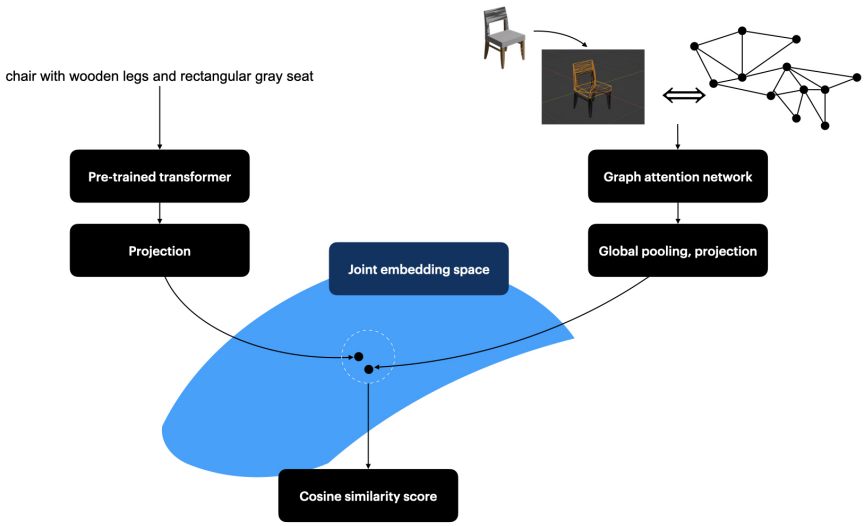


Figure 1: Diagram of our model architecture.

3.3 Contrastive Learning

As oppososed to discriminative learning, contrastive learning involves comparing samples. Input pairs are constructed by some notion of similarity; this similarity comes from the data itself, eliminating labelling needs. The goal of contrastive learning is to create a representation space in which similar examples are close together and dissimilar examples are far apart. In our case, input pairs are constructed from 3D shapes and their corresponding natural language descriptions.

CLIP (Contrastive Language-Image Pre-training) [2] is a largely successful application of contrastive pretraining to establishing a relationship between images and text. The authors jointly train an image encoder and text encoder to predict correct text-image input pairs, simply taking a dot product in the learned joint embedding space to calculate similarity. Text2Shape [3], a model that operates on text and 3D shapes, also creates a multimodal embedding space to push semantically similar text and shapes together, using slightly more complex association and metric learning losses. Text2Shape is the primary basis of our work. Our contributions improve the expressiveness and flexibility of their data representation and encoders.

4 Approach

We jointly train a text encoder and a mesh encoder with a symmetric cross-entropy loss. The model takes as input a batch of shape and description pairs (multiple descriptions can map to a single shape). After retrieving text and shape embeddings from the text encoder and mesh encoder respectively, we take the dot product of these embeddings to produce mesh logits over descriptions and description logits over meshes. To train, we compute the cross-entropy between the probability distribution obtained from these logits and target distributions. Targets are one-hot vectors for description logits that should match to single meshes, and are uniform distributions for mesh logits that can match with multiple descriptions.

4.1 Text encoder

We fine-tune CLIP's [2] pretrained language model, a masked self-attention Transformer made available by Hugging Face. We feed its output into a linear readout layer and normalize the results which gives us final description embeddings. We use CLIP's Transformer because we expect it to have helpful biases from its previous experience with multimodal data and contrastive pretraining.

Aspect-aware features: Shape descriptions contain crucial geometric information that, from a human perspective, are the determinants of matching shapes. For instance, in the description "A gray colored

chair that has a curved back seat..," our model should be particularly sensitive to the phrases "gray colored chair" and "curved back seat." In a comparable approach to that of DAE-GAN [12], we augment our textual data by extracting noun-adjective pairs from descriptions. We then concatenate these pairs and send them through the text encoder, resulting in what we call the *descriptive context*. We separately encode the sentence as a whole, which we call *global context*. Following CLIP [2], we choose to encode each input as the as the output embedding of `[EOS]`. Finally, we compute the final description embedding by concatenating the two context embeddings and passing them through a 2-layer MLP.

4.2 Mesh encoder

Shapes in our dataset are encoded as vertices belonging to faces in their raw form. We convert this data into a set of triangle meshes. For each vertex, we extract node features as that vertex's coordinates and RGB color.

We consider with a few different versions of the mesh encoder. All build off of a 3-layer GAT. After we obtain final node representations, we pass them into two pooling layers: a global mean pool, which averages features of all nodes, and/or a global max pool, which takes the max at all feature indices. These poolings obtain a graph-level embedding that we concatenate, submit to an MLP, and normalize to obtain a final embedding for the mesh that we compare with description embeddings.

Directional Mesh Encoder: In addition to a vanilla GAT, we build a custom architecture that intersperses GAT convolutional layers with an edge convolutional operator known as EdgeConv [13]. This operator learns a simple 2-layer MLP over the difference in a node's features and its neighbors' to update that node. More precisely, given a node v with current embedding $\mathbf{v}$ and a neighbor node $u \in N(v)$ with current embedding $\mathbf{u}$, the expression $\mathbf{v}||\mathbf{u} - \mathbf{v}$ ($||$ refers to concatenation) is passed into the MLP. For each v, this is done for all nodes $u \in N(v)$, and the MLP outputs for each u is summed and taken as the new embedding for v.

$$\mathbf{v} := \sum_{u \in N(v)} MLP(\mathbf{v}||\mathbf{u} - \mathbf{v})$$

Because we encode vertex coordinates as node features, this expression, which includes the term $\mathbf{u} - \mathbf{v}$, essentially computes and transforms distances between vertices. Graphs lack an inherent coordinate system, but we are dealing with a task where physical positioning and distances are vital. Using the EdgeConv operation may equip the graph neural network with better positional understanding.

5 Experiments

5.1 Data

The ShapeNet dataset [14] consists of over 50,000 richly annotated 3D meshes. Text2Shape uses a subset of this dataset, consisting of 8,447 and 6,591 models of tables and chairs, respectively. Each model is paired with, on average, 5 corresponding natural language descriptions, making for 75,344 descriptions in total. While Text2Shape creates voxelizations of these models, we instead use unaltered meshes from the original ShapeNet dataset.

5.2 Evaluation method

Our evaluation metric is top-5 model retrieval accuracy (RR@5). RR@5 is the percentage of descriptions whose corresponding mesh's similarity score is among the top 5 scores. Intuitively, if one is searching a database for a shape and types in a qualitative description, the "right" mesh need not be the top-scoring one; a user could glance around the page.

During training, we evaluate RR@5 on a subset of the shuffled train set, chosen so that the size of this subset is equal to the size of the test set. This allows for fairer comparison, since more samples and more options make it harder for the model to do perfect retrieval.

RR@5 does not capture the suitability of other top-scoring meshes for given descriptions, so it may be an underestimate of our model's practical performance. This being the case, we also conducted brief qualitative investigation of model results. We randomly sampled top-5 matching mesh predictions for descriptions in our dataset, and considered the extent to which the retrieved meshes aligned with descriptors.

5.3 Experimental details

To reduce training time, we used a subset of the dataset used by Text2Shape, with 4000 samples in the train set and 500 samples in the test set. In nearly all experiments our models were trained over 30 epochs, with a batch size of 32 and embedding dimension of 128. One text-encoder and mesh-encoder combination, CLIP Transformer + Aspect and GAT, had not obviously converged after 30 epochs, so we let it train for 64 epochs. We used Adam for optimization, with a learning rate of 5×10^{-5}, weight decay factor of 0.2 and $\beta = (0.9, 0.98)$.

The text encoder and mesh encoder pairs we experimented with are as follows. "Aspect" indicates that we used concatenated aspect-aware features. "Deeper" CLIP Transformer is the pre-trained model we otherwise used with one additional ReLU and linear layer.

Text encoder	**Mesh encoder**
CLIP Transformer	GAT
CLIP Transformer + Aspect	GAT
Deeper CLIP Transformer + Aspect	GAT
CLIP Transformer	Directional Mesh Encoder
CLIP Transformer + Aspect	Directional Mesh Encoder

In earlier experiments, we ran an implementation of GraphSAGE [15] as our mesh encoder in combination with CLIP's Transformer. It performed barely better than random, so we have excluded this run from further analysis.

5.4 Results

Retrieval accuracy.

Model	**Train RR@5**	**Test RR@5**
Random Retrieval	$\approx 1\%$	$\approx 1\%$
CLIP Transformer, GAT (Baseline)	3.84%	2.67%
CLIP Transformer + Aspect, GAT (30 epochs)	$\mathbf{6.68}\%$	$\mathbf{4.44}\%$
CLIP Transformer + Aspect, GAT (64 epochs)	$\mathbf{9.84}\%$	$\mathbf{5.20}\%$
CLIP Transformer + Deep Aspect, GAT	4.96%	3.84%
CLIP Transformer, Directional	4.48%	3.48%
CLIP Transformer + Aspect, Directional	5.16%	3.92%

Table 1: Train and test recall rates in top-5. All models were trained for 30 epochs unless otherwise stated.

Our best performing encoder combination, Deeper CLIP Transformer + Aspect and GAT, achieved 4.44% on test, which is 4.4x better than random chance. In contrast, Text2Shape achieved 2.37% RR@5 over a test set of size 1,500, doing 8x better than random chance.

Because of CLIP Transformer + Aspect and GAT's promising performance, we decided perform an additional extended training run. After training for 64 epochs total, this configuration achieved 5.20% test retrieval accuracy, 5.2x better than random chance.

Directional Mesh Encoder fared worse than a simple GAT on the task. This may be due to the fact that it is roughly twice as deep as our vanilla GAT and is experiencing over-smoothing, a common phenomenon in deep graph neural networks [16]. It may also be the result of an over-presence in dropout layers with too high a dropout rate, making it difficult for the mesh encoder to learn.

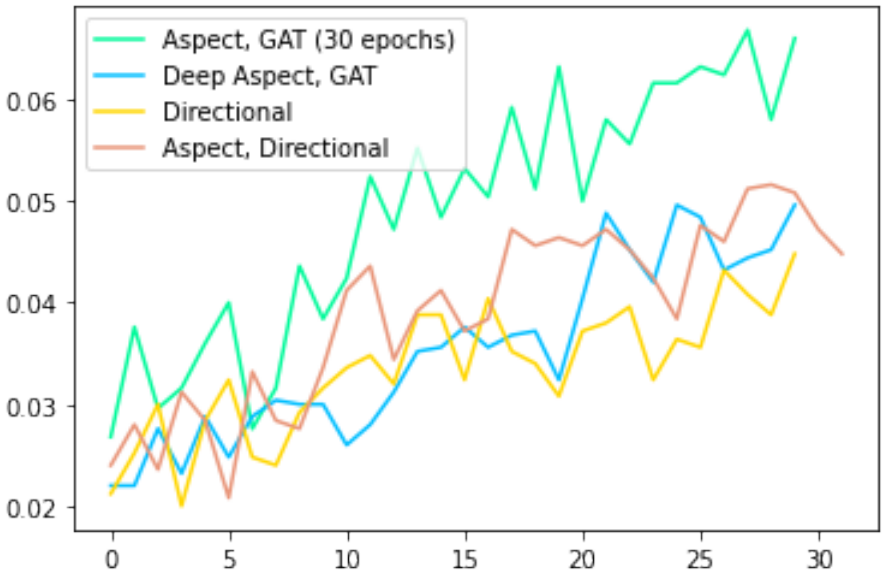
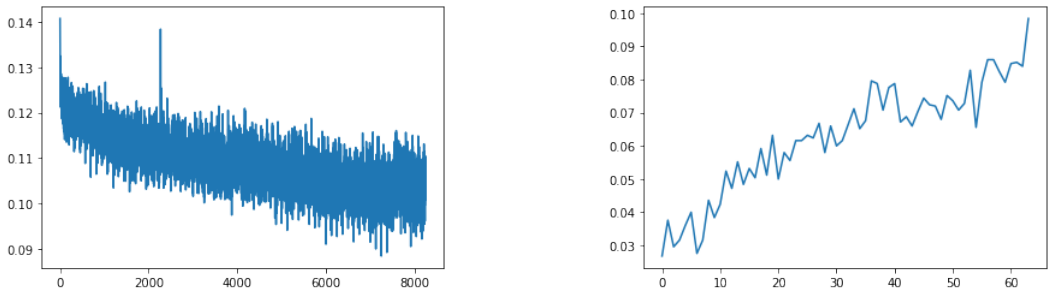


Figure 2: Train accuracies per epoch for each model (excluding baseline and CLIP Transformer + Aspect, GAT (64 epochs).



(a) Loss plot by batch

(b) Train accuracy plot by epoch

Figure 3: Plots for CLIP Transformer + Aspect, GAT (64 epochs). Loss continues to decline and training accuracy continues to increase even after 64 epochs, suggesting that additional training may further increase performance.

Across both architecture configurations, we see that including description-aware features unambiguously improves retrieval performance. We tentatively attribute this result to the high information value conferred by the added descriptive context; the model must increasingly pay attention to noun-adjective pairs since that is all the data available in the aspect-aware features.

Qualitative evaluation.

To evaluate our best model's qualitative performance, we retrieved the top-5 scoring meshes for a given description. To do so, we embedded the query description using the CLIP Transformer + Aspect text encoder. Then, we generate embeddings for all meshes in the validation set using GAT. We then compute dot-product similarity scores between the query embedding and each of the meshes, which are then softmaxed in order to produce a probability distribution. The meshes with the top-5 probabilities for two different descriptions are displayed above in Figure 4.

The first description mentions several key features: a cushioned seat and back, no arm supports, and wooden legs dark brown in color. The fourth-highest scoring chair is an almost-perfect qualitative match, while the others have a subset of at least one or two of the features.

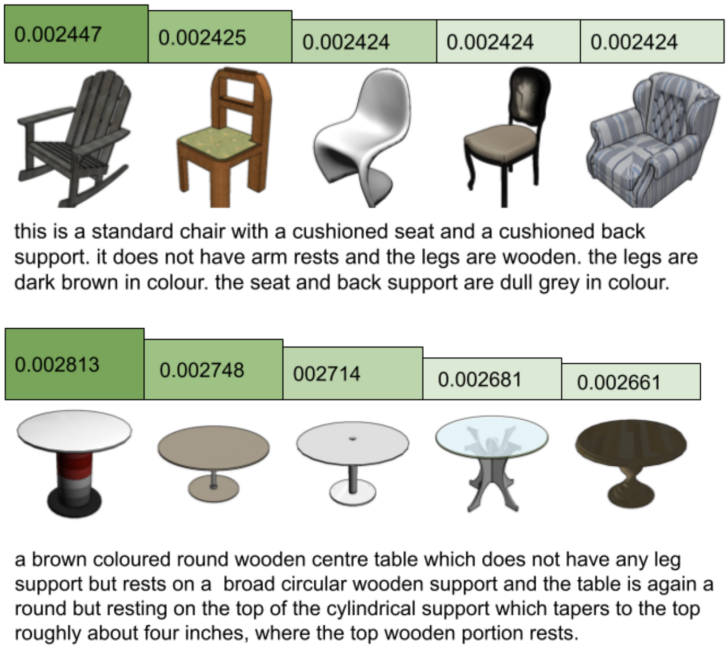


Figure 4: Randomly retrieved examples from ShapeNet and the top 5 probabilities put on each shape matching the given description by the CLIP Transformer + Aspect and GAT model combination. These probabilities are from a distribution over 500 possible meshes.

The second description calls for a round wooden table which rests on a tapered cylindrical support. All of the retrieved tables are in fact round, and all but the fourth have cylinder-like bases. The fifth-highest scoring one is even wooden, a very strong match for the description.

Overall, the ability of our model to retrieve meshes with appropriate physical features demonstrates the expressive embedding space learned by our models.

6 Analysis

The model retrieves matching models reliably. It competes with Text2Shape despite the fact that it learns over a novel shape representation that is arguably more complex. Text2Shape also trained with batch size more comparable to the size of the evaluation set it used. Having more negative examples in training could be useful for pushing non-similar examples far apart and in particular learning hard negatives.

The loss and accuracy plots for the extended training run of CLIP Transformer + Aspect with GAT are in Figure 3b. While loss steadily decreases in the long term, the loss curve is quite noisy. This, we believe, is because contrastive learning is highly dependent on batch size. A larger batch size may smooth learning because there are far more negative examples per batch. In our case, each batch of size N contains N meshes, each paired with 5 matching descriptions. The contrastive loss high cosine similarity among within-mesh descriptions, and low cosine similarity for descriptions of differing meshes. Thus, each mesh embedding must maximize its distance from $5(N-1)$ negative examples in order to improve performance. As N increases, the increased constraints each description embedding provide extra supervision, aiding the learning process.

CLIP, for example, was trained with a very large batch size of 32,768. This allowed for numerous negative examples per batch. With our limited GPU memory, however, we were only able to fit at most 8 examples in each batch. This vastly smaller batch size meant that the negative examples in each batch were far less representative of the entire dataset. We were able to partially circumvent this using gradient caching [17]. This allowed us to emulate a larger batch size of $N = 32$ at the cost of increased runtime.

Qualitatively, the model understands basic descriptors, such as "round." Some of the descriptions in the dataset are relatively complex and include multiple aspects. The model appears to jointly optimize by returning some shapes that match some of the aspects but not others. We observed that some chairs are repeatedly retrieved by the model despite sharing low qualitative similarity with textual descriptions, such as the gray cushioned chair in Figure 4. One hypothesis for this behavior is that some chairs are complex and therefore meet many descriptions, so the model learns to always include in top probability meshes. Expanding the size of the dataset on which we train and evaluate might alleviate this problem.

7 Conclusion

In this paper, we developed a joint embedding space for natural language and 3D shapes by training a text encoder and mesh encoder in tandem. We see that the graph is a powerful abstraction for 3D objects, and that GNNs produce shape embeddings compatible with language descriptions. When coupled with expressive pre-trained transformers and aspect-aware features, GNN-based mesh encoders perform well on shape retrieval. Aspect-aware features proved to be an important asset for our model, while increasing model depth and number of layers appeared less effective. This is perhaps a sign that standard models such as pre-trained transformers and GATs are already sufficient. Greater data augmentation techniques and feature selection may be more important in this domain.

Future work that trains our model for longer and on larger datasets/batch sizes has the potential to produce even greater results. Our dataset was comprised solely of tables and chairs; it is unknown to what extent our methodology transfers to other 3D shapes. We are also interested in conducting generative experiments analogous to those of Text2Shape leveraging embeddings from our joint representation space.

8 Acknowledgements

We would like to thank Casey Manning who taught us how to write Blender scripts that were used to bake vertex colors to our 3D object models in our preprocessing pipeline.

References

[1] Phuc H. Le-Khac, Graham Healy, and Alan F. Smeaton. Contrastive representation learning: A framework and review. *IEEE Access*, 8:193907–193934, 2020.

[2] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. *CoRR*, abs/2103.00020, 2021.

[3] Kevin Chen, Christopher B. Choy, Manolis Savva, Angel X. Chang, Thomas A. Funkhouser, and Silvio Savarese. Text2shape: Generating shapes from natural language by learning joint embeddings. *CoRR*, abs/1803.08495, 2018.

[4] Yun-Peng Xiao, Yu-Kun Lai, Fang-Lue Zhang, Chunpeng Li, and Lin Gao. A survey on deep geometry learning: From a representation perspective. *CoRR*, abs/2002.07995, 2020.

[5] Zhirong Wu, Shuran Song, Aditya Khosla, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets for 2.5d object recognition and next-best-view prediction. *CoRR*, abs/1406.5670, 2014.

[6] 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.

[7] Charles Ruizhongtai Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *CoRR*, abs/1612.00593, 2016.

[8] Yangyan Li, Rui Bu, Mingchao Sun, and Baoquan Chen. Pointcnn. *CoRR*, abs/1801.07791, 2018.

[9] Weijing Shi and Ragunathan Rajkumar. Point-gnn: Graph neural network for 3d object detection in a point cloud. *CoRR*, abs/2003.01251, 2020.

[10] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2009.

[11] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks, 2018.

[12] Shulan Ruan, Yong Zhang, Kun Zhang, Yanbo Fan, Fan Tang, Qi Liu, and Enhong Chen. DAE-GAN: dynamic aspect-aware GAN for text-to-image synthesis. *CoRR*, abs/2108.12141, 2021.

[13] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon. Dynamic graph CNN for learning on point clouds. *CoRR*, abs/1801.07829, 2018.

[14] Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. ShapeNet: An Information-Rich 3D Model Repository. Technical Report arXiv:1512.03012 [cs.GR], Stanford University — Princeton University — Toyota Technological Institute at Chicago, 2015.

[15] William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. *CoRR*, abs/1706.02216, 2017.

[16] Chen Cai and Yusu Wang. A note on over-smoothing for graph neural networks. *CoRR*, abs/2006.13318, 2020.

[17] Jiawei Han Jamie Callan Luyu Gao, Yunyi Zhang. Scaling deep contrastive learning batch size under memory limited setup. In *Proceedings of the 6th Workshop on Representation Learning for NLP*, 2021.