Self-supervised Learning on Point Clouds using Discriminators on Approximate Convex Decompositions

Abstract

Since annotating 3D data is expensive, several recent works have proposed pretraining models on self-supervised tasks before transferring representations to smaller datasets of downstream applications. Recent work [6] proposed a simple self-supervised task of first decomposing an object into approximate convex polyhedra followed by metric learning on this approximate segmentation. They showed encouraging transfer learning results and a focus on higher-level semantics of an object with this approach.

Inspired by the success of [6], we proposed a new self-supervised algorithm which applies discriminators on point clouds segmented as approximate convex polyhedra. Our key idea is to perform a random perturbation (such as rotation, scaling, dropping) of a single approximate polyhedra, and teach a model to discriminate between the perturbed and original 3D objects. We conduct several experiments testing our approach on unsupervised shape classification & few-shot part segmentation, and notice competitive performance of individual perturbations compared to [6]. Our best method interpolates the losses from our proposed task and the method in [6], outperforming re-runs of [6].

1. Introduction

Deep learning on 3D data is crucial for the success of many important real-world applications like animation, medical imaging, self-driving cars, robotics, etc. Unfortunately, these algorithms tend to be data hungry and annotating rich 3D data is difficult and expensive. The unavailability of large training sets for fundamental 3D vision problems of shape classification and part segmentation has resulted in a step towards using self-supervised approaches to learn 3D shape representations. An important aspect of this approach is identifying a self-supervision task that would benefit downstream tasks. A promising approach in this direction is Gadhela et al. 2020 [6] who perform metric learning on Approximate Convex Decompositions (ACD) of 3D point clouds. The idea in ACD is to automatically decompose 3D objects into constituent maximal convex polyhedra, which capture structural properties of the shape. We believe ACD is a great technique to "discretize" point-clouds, converting a set of 1000s of points to a small set of 20-30 convex polyhedra with semantically meaningful properties.

Inspired by [6], we leverage ACD as a preprocessing step for a new self-supervised learning algorithm. Our key idea is to apply systematic deformations (such as rotation, scaling, dropping) on individual convex polyhedra derived from ACD. These deformations help maintain the local consistency of point clouds within the convex polyhedra, but perturb the high-level structural properties of the 3D object. To use these perturbed objects, we train a model to discriminate between unperturbed objects and perturbed objects using a binary classifier on top of point cloud representations. This discriminator approach resembles self-supervised learning in natural language processing using discriminators [3] as well as discriminators in Generative Adversarial Networks [1, 8].

We conduct several experiments to evaluate two aspects of our approach: 1) how easy is the proposed self-supervised task for neural networks? 2) how well does the approach transfer to downstream tasks like unsupervised shape classification and few-shot part segmentation? Our results indicate that the dropping and rotation perturbations are harder to learn than scaling, but also lead to better downstream performance. Individual perturbations perform competitively to the algorithm proposed by [6]. An interpolation between the losses from our method and [6] gets the best performance, beating re-runs of [6].

Proposed vs accomplished

- ~~Read and understand the codebase of [6]~~
- ~~Modify codebase of [6] to perform systematic perturbations to the convex polyhedra for ShapeNet objects~~
- ~~Modify codebase of [6] to perform classification of real vs fake objects~~
- Modify codebase to perform reconstruction of original point clouds using chamfer distance
- ~~Conduct experiments analyzing difficulty of our~~

~~self-supervised algorithm, its transfer success to other tasks like segmentation, shape classification~~

2. Related Work

Representation learning on 3D point sets: Traditional deep learning methods are designed for densely structured input data but 3D point clouds are unordered sets of vectors and hence harder to train on directly. Volumetric methods [10, 20, 12, 23] and multi-view methods [19, 14, 24] have been proposed that use traditional deep learning models on pooled features from rendered views. But these methods struggle with larger point clouds and when point clouds have varying density. They do not perform well on tasks like per-part segmentation either. Hence in the recent years, works like PointNet [16] have shown that point features can be processed directly to obtain point level features that are permutation invariant and these features can then be pooled to obtain global features. But all these methods rely heavily on labeled data for learning downstream tasks such as classification and segmentation. Hence label-efficient methods are desirable and an active field of research. To this end many Self-supervised learning based methods have been suggested whose main aim is to train a network on tasks that allow models to learn intermediate representations that can help with training downstream tasks.

Self Supervised Learning (SSL): SSL has shown a lot of promise in fields like Computer Vision [5, 13, 7] and NLP [4, 21, 22] for training on tasks with fewer labels. SSL is a variation of unsupervised learning that exploits tasks providing "free" data labels to learn intermediate representations. SSL can be seen as a pipeline consisting of two steps: 1. Pre-train a network on a task with unlabeled data and, 2. Fine-tune this network for downstream tasks. The advantage of this pipeline is that fine-tuning on the downstream task can be attained by much fewer labeled data as our learned intermediate representation already encodes some generic features relevant to the downstream task. Some papers in computer vision have proposed the task of predicting the spatial transformations between parts of images [7, 13] where they re-use the labels generated by transformed pixel arrangement. We draw inspiration from such approaches and design our tasks by adding different types of perturbations to objects. But working directly on all the points in a point cloud is costly, so we look at methods that allow us to decompose the point clouds into smaller regions which can then be treated individually.

Approximate Convex Decomposition: 3D shapes can be represented as union of multiple convex components, but performing exact convex decomposition leads to a high number of convex parts while being very expensive computationally. Hence we look at Approximate Convex Decomposition(ACD) which allows concavity up to a certain tolerance. Volumetric Hierarchical Approximate Convex Decomposition (V-HACD) [11] is one such method that computes convex decomposition of the shapes from their volumetric representation obtained after voxelisation of the point cloud.

[6] decompose a shape using V-HACD and then assign component labels to the points in the original ShapeNet mesh using nearest neighbour matching with points sampled from each of the components. Here they formulate their self-supervised task as a metric learning problem on point embeddings using contrastive loss [9]. [15] and [18] implement a similar approach where the self-supervised task is to reconstruct a point cloud after random spatial transformations are applied to the decomposed parts. Similar to prior work, we leverage ACD to identify semantically meaningful parts of an object. However, unlike prior work we apply different types of perturbations (like scaling, rotation, dropping) to an *individual* ACD component in 50% of the dataset, and train a classifier over PointNet++ features to identify whether a shape is real or fake.

3. Description of method and Architecture

Overview: We propose a new self-supervised task for 3D objects, inspired from recent work in NLP literature on using discriminators for self-supervised learning [3]. Our approach is to systematically deform individual convex polyhedra in the approximate convex decompositions of 3D point clouds and train our model to classify whether the final shape is real or has been tampered with. We believe the information learned as a part of the classification process helps the model perform well on various downstream tasks where annotating 3D data is expensive.

Approximate Convex Decomposition (*creation of data for self-supervised learning*): We use Approximate Convex Decomposition's of 3D point cloud representations for our Self-supervised learning task similar to [6]. Compared to exact convex representations using ACD ensures lower-level properties of shape are intact & consistent, so that the model can focus on higher-level semantics. ACD created using Volumteric Hierarchical Approximate Convex Decomposition [11] has multiple advantages over other methods. The first step is to convert point cloud representations of the shape into Occupancy grids. Once the voxelization is complete, the volume is split into half along one of the three axes. This is done recursively until the realisation of desired number of components.

The perturbations: We systematically deform a single random convex polyhedra which is later classified as a part of the training objective. We apply three types of perturbations

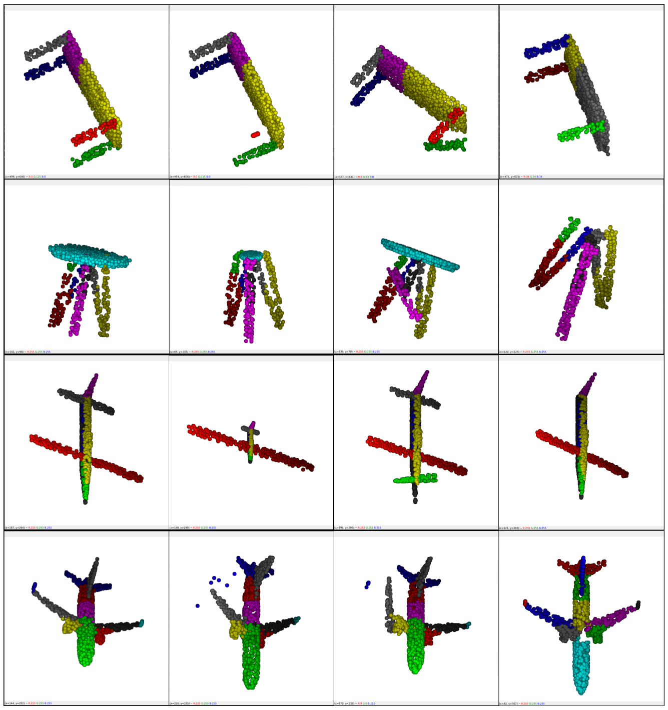

Figure 1. Different kinds of perturbations (at a convex polyhedra level) applied to four different point clouds (rows of the figure). The four columns represent — 1. the original ACD of the 3D Point Clouds 2. SCALE perturbation 3. ROTATE perturbation 4. DROP perturbation. Since these perturbations lead to unnatural objects (which are still locally consistent), we expect models trained to discriminate between real/fake objects will learn meaningful higher-level semantics of 3D objects.

with multiple hyperparameters to create different types of deformations as shown in Figure 1.

We first select a random polyhedra from the set of ACD components and apply one of three perturbations (DROP, SCALE, ROTATE) to it. For the DROP perturbation we completely drop the selected segment from the input creating a deformed 3D point cloud representation. As seen in the fourth column of Figure 1, parts of the objects like a table leg, stool head, tail wings have been removed (compared to first column). For the SCALE perturbation we re-size the randomly selected segment (with respect to its centroid) by a randomly chosen amount amongst [0.125, 0.25, 4, 8]. As seen in the second column of Figure 1, a table leg, stool head, fuselage or plane wings have been compressed relative to rest of the object. Finally, for the ROTATE perturbation, we simply rotate the convex polyhedra through the centroid along either the x, y or z axis by a randomly selected rotation amount among the choices [45, 90, 135, 180, 225, 270, 315] degrees. In the third column of Figure 1 the table leg, stool leg, plane wing / cockpit have been rotated relative to the rest of the object.

Discriminator: We apply a systematic deformation to random 50% of the examples in the minibatch. After systematic deformation over approximate convex polyhedra, our self-supervised training objective is a discriminative task that distinguishes the real objects from deformed ones. We extract features from three different parts of the PointNet++ model [17] (similar to the feature extraction setup for ModelNet40 experiments in [6]), pool representations from individual points using average pooling, and classify them as real or fake. The loss function used is binary cross entropy, after a linear projection layer on the feature set. We also experiment with interpolating the original ACD loss from [6] with our proposed discriminator.

4. Experimental Setup

Model configurations: To test the efficacy of our method, we evaluate nine different configurations of the system ablating components of our proposed method:

- CONTRAST (*baseline*): the original method proposed by [6], where point embeddings from the same ACD component are encouraged to be close to each other in vector space.
- PERTURB-DROP: DROP perturbation applied to 50% dataset, and model trained to classify real vs fake.
- PERTURB-SCALE: SCALE perturbation applied to 50% dataset, and model trained to classify real vs fake.
- PERTURB-ROTATE: ROTATE perturbation applied to 50% dataset, and model trained to classify real vs fake.
- PERTURB-ALL: For every shape in a random 50% of the dataset, one of the three perturbations is chosen uniformly randomly and applied. Model trained to classify real vs fake.
- CONTRAST + λ PERTURB-ALL: Both loss functions (CONTRAST & PERTURB-ALL) are used, interpolated by constant λ ($\lambda = 0.1, 1.0, 10.0, 100.0$).

Evaluation: We evaluate our self-supervised models with the goal of answering two research questions:

1) *How good are the models at the self-supervised task?* This evaluation gives us a sense of how easy/difficult the discrimative task is. To perform this experiment, we simply measure the binary classification accuracy of the model on a held-out set undergoing the same perturbation.

2) how well do these learned representations transfer to downstream tasks? This type of evaluation measure the actual usefulness of this self-supervised method to real-world 3D tasks. We use a setup identical to [6], testing the

Configuration	**fewshot ShapeNetSeg segmentation (k=5)** class avg. mIoU	 instance avg. mIoU
CONTRAST [6]	72.300 ± 1.800	-
CONTRAST (*our re-run*)	73.051 ± 0.203	74.261 ± 0.055
PERTURB-DROP	**73.616** ± 0.081	72.369 ± 0.069
PERTURB-SCALE	71.960 ± 0.071	73.260 ± 0.080
PERTURB-ROTATE	**73.700** ± 0.170	72.829 ± 0.080
PERTURB-ALL	72.813 ± 0.031	70.177 ± 0.090
CONTRAST + 0.1 * PERTURB-ALL	**73.764** ± 0.234	72.626 ± 0.077
CONTRAST + 1.0 * PERTURB-ALL	72.937 ± 0.094	73.631 ± 0.133
CONTRAST + 10 * PERTURB-ALL	70.207 ± 0.139	68.805 ± 0.093
CONTRAST + 100 * PERTURB-ALL	66.307 ± 0.121	63.620 ± 0.055

Table 1. Few shot semantic segmentation results on ShapeNetSeg, with $k = 5$, training each model for 9 epochs (default setting). Each result has been averaged across five different random seeds. Our results indicate that the DROP and ROTATE perturbations are more effective than SCALE, with best performance using an interpolation between CONTRAST and PERTURB-ALL.

Configuration	Discr. Accuracy
CONTRAST	N/A
PERTURB-DROP	63.6%
PERTURB-SCALE	91.7%
PERTURB-ROTATE	77.8%
PERTURB-ALL	74.9%
CONTRAST + 0.1 * PERTURB-ALL	73.4%
CONTRAST + 1.0 * PERTURB-ALL	74.7%
CONTRAST + 10 * PERTURB-ALL	75.5%
CONTRAST + 100 * PERTURB-ALL	75.7%

Table 2. Binary classification accuracy of the self-supervised discriminative task proposed in this project (classifying real shapes vs fake shapes) using early stopping in first 35 epochs. DROP is the hardest perturbation to identify, whereas SCALE is the easiest.

Configuration	ModelNet40 Acc. Train	 Dev	 Test
Random Weights [6]	-	-	78.0
CONTRAST [6]**	-	-	89.1
CONTRAST (*our re-run*, 5 epochs)	90.7	87.4	86.1
CONTRAST (*our re-run*, 95 epochs)	92.3	88.0	87.7
CONTRAST (*our re-run*, best epoch)	93.4	87.7	87.3
CONTRAST (*our re-run*, 5 epochs)	90.7	87.4	**86.1**
PERTURB-DROP	87.5	83.7	81.4
PERTURB-SCALE	91.8	83.6	81.2
PERTURB-ROTATE	90.6	85.8	**84.6**
PERTURB-ALL	90.7	84.1	82.9
CONTRAST + 0.1 * PERTURB-ALL	93.2	89.1	**88.2**
CONTRAST + 1.0 * PERTURB-ALL	95.2	88.7	86.6
CONTRAST + 10 * PERTURB-ALL	93.9	88.2	85.2
CONTRAST + 100 * PERTURB-ALL	93.8	86.3	84.5

Table 3. Unsupervised shape classification accuracy on ModelNet40 for different configurations of our model all trained for five epochs. Individually, ROTATE perturbations perform best. Overall, an interpolation between CONTRAST and PERTURB-ALL with $\lambda = 0.1$ gets the optimal performance. (**Note: we were unable to reproduce the numbers reported in [6] despite using their codebase, we've reported the numbers we obtained after re-running the code in the CONTRAST rows.)

model's performance on (i) unsupervised shape classification on ModelNet40 [23]; (ii) few-shot semantic segmentation of **ShapeNetSeg** part of ShapeNetCore [2]. We re-use the codebase[1] of [6] and keep identical hyperparameters[2] (please refer to their codebase and paper for more details).

5. Results

Reproducing Results from [6]: First, we attempt to reproduce the ModelNet40 and ShapeNetSeg results from [6]. We are successful in reproducing results on ShapeNetSeg, as seen in the CONTRAST rows of Table 1. However, we were unable to match the results on ModelNet40 with default hyperparameters. As seen in the CONTRAST rows of Table 3, we notice 2-3% inferior performance compared to [6] for a variety of checkpoints, possibly indicating overfitting. For a fair comparison and computational reasons, we hence evaluate on ModelNet40 using 5 epochs of self-supervised training on different configurations.

DROP perturbations hardest to discriminate, SCALE perturbation is the easiest: We present our first set of

[1] https://git.io/J3N9z

[2] Only batch size is halved to 16 due to GPU memory, but we also do this for the baseline system CONTRAST.

results in Table 2, measuring binary classification accuracy on our proposed self-supervised task (on a held-out validation set). We train all models to 35 epochs, and measure best validation accuracy. We find that the model struggles to classify the DROP perturbations (only 63.6% accuracy), but finds it much easier to classify SCALE (91.7%) and ROTATE perturbations (77.8%). We also notice a gentle increase in classification accuracy as we interpolate more strongly with PERTURB-ALL (73.4% to 75.7%). Note that we expect all accuracy scores to improve with more epochs of training — DROP / ROTATE jump to 67% / 79% with 50-60 epochs. However, training is expensive (45 to 60 minutes per epoch using single 1080Ti GPU), so we could only run more epochs for a limited set of configurations.

ROTATE perturbation performs best on both downstream tasks, SCALE is the worst: We see a similar trend on both downstream tasks, unsupervised shape classification in Table 3 and few-shot part segmentation in Table 1. For the individual perturbation models, we find ROTATE to be the most effective (84.6% on ModelNet40, 73.7 class average mIoU on ShapeNetSeg), followed by DROP (81.4% / 73.6). SCALE is the least effective perturbation (only 81.2% / 72.0). Perhaps, the SCALE perturbations are too easy to learn meaningful representations ("ease" defined by results in Table 2), and DROP is too hard / ambiguous for the model. We believe ROTATE has the appropriate difficulty, teaching the model good transferable representations. Note that in Table 3 we see that all individual perturbations beat the random weights performance from [6], so we believe all configurations learn some meaningful representations.

Interpolated models work best on both downstream tasks: In both Table 3 and Table 1 we find that the interpolation constant $\lambda = 0.1$ works best on both classification and segmentation (88.2% on ModelNet40, 73.8 class average mIoU on ShapeNetSeg). Performance generally degrades with higher interpolation constants (or higher weights on the PERTURB-ALL term). Performance with $\lambda = 0.1$ exceeds the baseline CONTRAST, which gets 86.1% on ModelNet40 and 73.0 on ShapeNetSeg. In fact, it's even higher than the ModelNet40 performance we get on CONTRAST for any of our re-runs with more epochs (86.1% - 87.7%). Note that [6] report higher scores on ModelNet40 (of 89.1%) but we could not reproduce their numbers as we have discussed earlier in this section.

6. Conclusion & Future Work

In this project we presented a new self-supervised learning algorithm where we trained a discriminator to classify between real and fake shapes. Our fake shapes were constructed by systematic perturbations of individual polyhedra obtained by segmenting point clouds using approximate convex decomposition. Overall our results were encouraging — 1) all individual perturbations learn some meaningful representations, beating a random weights baseline; 2) ROTATE is the most effective perturbation, SCALE is least effective; 3) An interpolation between the loss function from [6] and our PERTURB-ALL configuration gets the best result on both datasets, outperforming the numbers obtained by re-running the codebase of [6].

There are a number of research directions which can be explored as future work. An obvious next step would be exploring the effect of interpolating just ROTATE with CONTRAST, since it is the most effective perturbation. Another simple direction is checking the effect of modifying the perturbation hyperparameters — for instance, perhaps SCALE amounts closer to 1 will be more effective. Two other interesting directions to explore (with more implementation work) could be — 1) ensuring real/fake versions of same object are in same minibatch, or perhaps classifying a real shape from a pool of fake versions of same object; 2) using chamfer distance losses to reconstruct the original point cloud from a perturbed shape.

References

[1] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *International conference on machine learning*, pages 214–223. PMLR, 2017. 1

[2] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015. 4

[3] Kevin Clark, Minh-Thang Luong, Quoc V Le, and Christopher D Manning. Electra: Pre-training text encoders as discriminators rather than generators. In *International Conference on Learning Representations*, 2020. 1, 2

[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018. 2

[5] Carl Doersch, Abhinav Gupta, and Alexei A Efros. Unsupervised visual representation learning by context prediction. In *Proceedings of the IEEE international conference on computer vision*, pages 1422–1430, 2015. 2

[6] Matheus Gadelha, Aruni RoyChowdhury, Gopal Sharma, Evangelos Kalogerakis, Liangliang Cao, Erik Learned-Miller, Rui Wang, and Subhransu Maji. Label-efficient learning on point clouds using approximate convex decompositions. In *European Conference on Computer Vision*, pages 473–491. Springer, 2020. 1, 2, 3, 4, 5

[7] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations. *arXiv preprint arXiv:1803.07728*, 2018. 2

[8] Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *arXiv preprint arXiv:1406.2661*, 2014. 1

[9] R. Hadsell, S. Chopra, and Y. LeCun. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 1735–1742, 2006. 2

[10] Roman Klokov and Victor Lempitsky. Escape from cells: Deep kd-networks for the recognition of 3d point cloud models. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 863–872, 2017. 2

[11] Khaled Mamou, E Lengyel, and AK Peters. Volumetric hierarchical approximate convex decomposition. In *Game Engine Gems 3*, pages 141–158. AK Peters, 2016. 2

[12] Daniel Maturana and S. Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 922–928, 2015. 2

[13] Mehdi Noroozi and Paolo Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *European conference on computer vision*, pages 69–84. Springer, 2016. 2

[14] Deepak Pathak, Ross Girshick, Piotr Dollár, Trevor Darrell, and Bharath Hariharan. Learning features by watching objects move. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2701–2710, 2017. 2

[15] Omid Poursaeed, Tianxing Jiang, Quintessa Qiao, Nayun Xu, and Vladimir G Kim. Self-supervised learning of point clouds via orientation estimation. *arXiv preprint arXiv:2008.00305*, 2020. 2

[16] 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*, pages 652–660, 2017. 2

[17] Charles R Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++ deep hierarchical feature learning on point sets in a metric space. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 5105–5114, 2017. 3

[18] Jonathan Sauder and Bjarne Sievers. Self-supervised deep learning on point clouds by reconstructing space. *arXiv preprint arXiv:1901.08396*, 2019. 2

[19] 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*, pages 945–953, 2015. 2

[20] Maxim Tatarchenko, Alexey Dosovitskiy, and Thomas Brox. Octree generating networks: Efficient convolutional architectures for high-resolution 3d outputs. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2088–2096, 2017. 2

[21] Hong Wang, Xin Wang, Wenhan Xiong, Mo Yu, Xiaoxiao Guo, Shiyu Chang, and William Yang Wang. Self-supervised learning for contextualized extractive summarization. *arXiv preprint arXiv:1906.04466*, 2019. 2

[22] Jiawei Wu, Xin Wang, and William Yang Wang. Self-supervised dialogue learning. *arXiv preprint arXiv:1907.00448*, 2019. 2

[23] 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*, pages 1912–1920, 2015. 2, 4

[24] Qiangeng Xu, Weiyue Wang, Duygu Ceylan, Radomir Mech, and Ulrich Neumann. Disn: Deep implicit surface network for high-quality single-view 3d reconstruction. *arXiv preprint arXiv:1905.10711*, 2019. 2