Unsupervised Point-Cloud Reconstruction and Completion

Alessio Santangelo
alessio.santangelo@studenti.polito.it

Damiano Bonaccorsi
damiano.bonaccorsi@studenti.polito.it

Takla Trad
takla.trad@studenti.polito.it

Dipartimento di Automatica e Informatica (DAUIN)
Politecnico di Torino

Abstract

In this report, we present the results obtained by employing different deep learning models to tackle the problems of point cloud reconstruction and completion. Our work is based on the concept of auto-encoders, a particular kind of design where the final neural network is composed by two main modules, an encoder that outputs a shape-embedding feature vector, followed by a decoder, that outputs the reconstructed/completed shape. Different experiments on the ShapeNet dataset [1] show how our models behave on known and unknown categories, and which kind of solutions can lead to better performances. Nevertheless, the aim of our work is not to achieve state of the art results, but rather propose and discuss the ideas we came up with while working on this project.

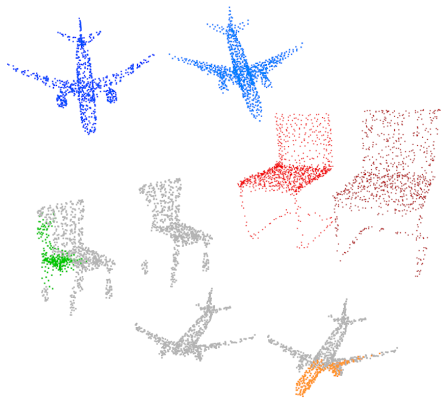

Figure 1. Visualization of the two problems of point cloud reconstruction and completion.

1. Introduction

Efficient methods to process 3D data have become more and more in demand during the past few years, given the rise in popularity and availability of sensors and cameras capable of capturing depth information, either by using stereo vision, or laser scanners like LiDAR. One particular kind of data that these sensors often produce is point clouds, i.e., sets of 3D coordinates, each representing a point lying on the surface of an object.

The point cloud representation has many advantages, as it is memory efficient, without having to really give up too much information regarding the geometry of the object, but it also comes with many drawbacks, one of them being that it is an unordered and unstructured representation, and another one being that most of the point clouds captured in the real world are often incomplete due to occlusion, transparency, or other limitations caused by many different conditions.

To overcome the first problem without having to rely on transformations to a different representation (like voxelization), PointNet [2] was proposed as a neural network architecture whose main idea is to extract per-point features and aggregate them by means of a symmetric function into a permutation-invariant feature representation. DGCNN [4] then builds upon PointNet's idea and improves it by using a graph convolutional network that intuitively matches much better the nature of a point cloud, if we think of each point as a vertex of a graph, respectively connected to its neighbouring vertices.

In our work, we make use of both PointNet [2] and DGCNN [4] to address the secondly mentioned problem of point clouds being often incomplete, employing them as feature extractors in our encoders, and proposing different auto-encoder architectures capable of performing point cloud reconstruction and point cloud completion, whose examples can be seen in figure 1.

In particular, we begin our research by dealing with the reconstruction task, in order to get familiar with working on point clouds and auto-encoders. Here, we also propose our particular training approach, which smartly makes use of data augmentation to increase generalization and thus provide better reconstruction results on novel categories, never seen at training time.

We then move on to the more complex completion task, where we explore different solutions for the decoder portion of the model, proposing our own version of a possible completion pipeline, taking inspiration from DeCo [5] and PF-Net [8]. Specifically, our encoder is obtained by combining two different DGCNNs [4], that capture features respectively at a local and at a global scale, while our decoder is the PPD (Point Pyramid Decoder), proposed in [8]. We call this architecture DGPP as in Double-Graph Point-Pyramid. It is worth mentioning that, as in [5] and [8], we also focus on reconstructing the missing patch only, rather than the whole shape, avoiding the genus-wise distortion [6] that stems from unnecessarily modifying the input partial point cloud. We finally shift our focus to possible pretext tasks to be used to separately pre-train the local and global encoders, as seen in [5], setting up a transfer learning problem instance.

The main contributions of this work are summarized as follows:

- We compare different combinations of encoders and decoders to tackle the problems of reconstruction firstly, and completion secondly.
- We propose a training approach that makes use of data augmentation to improve reconstruction results on novel categories.
- We propose DGPP as a possible completion architecture, which is a middle-ground between DeCo [5] and PF-Net [8], as it makes use of a global-local parallel encoder, and a point pyramid decoder.
- We explore and discuss possible self-supervised pretext tasks for our local and global encoders, in order to improve performance of the downstream completion task.
- We present the results obtained by extensively training many different ablations on a subset of the Shapenet-Part [9] dataset, and subsequently testing these ablations on the Shapenet-Core [1] dataset on both similar and dissimilar unknown categories, for both the reconstruction and completion problems

2. Related Work

Geometric Deep Learning on Point Clouds

PointNet [2] has developed a novel method to process raw point clouds, avoiding previously-common transformations into other representations, such as voxel grid or multiple 2D views of the object, and setting a new standard for deep learning on point clouds. The proposed network accepts a set of x, y, z coordinates as input, processes each point of the cloud individually, and then aggregates the collected information using a symmetric function. As a result, a feature vector embedding the total point cloud is inferred, which can then be used for a subsequent task. By design though, PointNet [2] can not capture local structures induced by the metric space in which points exist, limiting its potential in extracting local features. Several deep learning architectures have been proposed since then, the majority of which are highly influenced by PointNet [2] as they try to make up for its shortcomings. PointNet++ [3] for example, proposes hierarchical feature extraction by stacking two or more PointNets together, while DGCNN [4] introduces the idea of using graph-convolutional layers, named EdgeConv, which extract features between a point and its k nearest neighbours, consequently constructing a graph-like representation of the point cloud.

Point Cloud Completion

Point cloud completion is the problem of reconstructing the missing patch of a point cloud given its partial observation. Different ways to solve the problem have been formulated, starting with the pioneering Point Completion Network PCN [7], which used an encoder-decoder architecture to generate fine-grained total completions. A subsequent work [11] instead splits the process in two stages. First, the network predicts a complete but coarse-grained point cloud, then it merges the coarse-grained prediction with the input point cloud and uses a novel sampling algorithm to obtain the final point cloud. These and other approaches however result in an unnecessarily tough formulation, as they try to reconstruct the entire point cloud rather than simply filling in the missing area. Moreover, as explained in [6], they suffer from the so-called genus-wise distortion [6].

PF-Net [8] on the other hand, proposes an architecture to hierarchically generate the missing patch only, while also implementing an adversarial loss to force the distribution of predicted and real missing regions to be as close as possible. More recently, DeCo [5] builds upon PF-Net [8], and introduces as novelty the usage of a denoising task to collect local cues and a contrastive learning [12] task to learn global features, plus an auxiliary task of reconstructing the region neighbouring the missing patch, called "frame". This way DeCo [5] ensures a smooth blending of the generated points with the partial input.

3. Methodology

3.1. Point Cloud Reconstruction

In this section, we present the different types of auto-encoders we used to address reconstruction, and our reasons for choosing them. To ensure a fair comparison, all the encoders output a 1024-long embedding code, that is downsampled using an MLP into a 256-long feature vector, and then decoded by a simple fully-connected decoder. We chose 256 as code size to ensure downsampling would not become too much of a bottleneck for the encoders. This first section of our research basically aims at comparing the different encoders, while providing a lightweight environment to rapidly test different ablations, training parameters and strategies.
The three networks chosen as encoders for our comparison are PointNet [2], PointNet+1 (not to be confused with PointNet++ [3]) and DGCNN [4]. They are all implemented following their respective original formulation, but slightly modified to output a single feature vector. The only difference between the original PointNet [2] and PointNet+1 is that we added an additional intermediate block in the latter, as we wanted to see what would happen if we had simply added more learnable parameters. The architecture of the adapted DGCNN [4] encoder can be seen in figure 2.

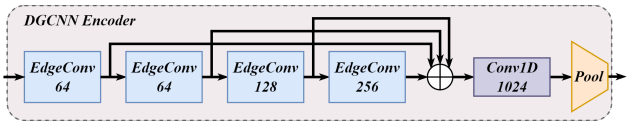

Figure 2. *DGCNN-based encoder.*

The loss function used to compare the output point cloud with the input ground truth is the Chamfer distance, an evaluation metric that takes into account the distance of a pair of points. For each point in each set, it finds the nearest point in the other set, and sums the square of distance up. The Chamfer distance between sets S1 and S2 is defined as:

$$d_{CD}(S_1, S_2) = \sum_{x \in S_1} \min_{y \in S_2} \|x - y\|_2^2 + \sum_{y \in S_2} \min_{x \in S_1} \|x - y\|_2^2$$

3.2. Generalization using Data Augmentation

One key-objective of our research was to find suitable strategies to increase generalization, and make sure our model would perform as best as possible when reconstructing categories never seen at training time. We found that using data augmentation during the first half of the training session, and turning it off during the second half, helped the models in achieving slightly better results on known classes but massively better results on novel categories. Data augmentation in this case consists simply in randomly rotating the point cloud and applying a bit of randomic jitter.

3.3. Point Cloud Completion

In this section, we employ our newly-acquired knowledge to extend the reconstruction framework to now address point cloud completion. Our starting point is a naive approach to completion, then, taking inspiration from DeCo [5] and PF-Net [8], we will move on to a much more reasonable approach and propose DGPP, our definitive completion architecture.

3.3.1 Naive Trials

In the following, we will be referring with X to the original point cloud, with Xp to the partial point cloud, obtained by removing M points from X, and finally with Y to the generated output point cloud
A naive approach to completion consists in:

- taking X and normalizing it to the unitary sphere.
- picking a random viewpoint and dropping the M closest points to obtain Xp.
- feeding Xp to the auto-encoder and inferring Y.
- computing the loss (CD) between X and Y.

This method did not necessarily provide bad results, if anything, most point clouds were being completed correctly from a qualitative point of view. The major flaw of this approach was that, as expected, we incurred in the aforementioned genus-wise distortion [6] phenomenon, i.e., given an input partial point cloud of a certain class, an airplane for example, the output point cloud would be reconstructed to resemble the "ideal" airplane that the model had learned to reconstruct, resulting in a generated point cloud that did not retain the peculiarities of the original one. Figure 3 provides an example visualization of the phenomenon.

We only tested two ablations with this approach, respectively using the PointNet+1 and DGCNN-based encoders [4], but then quickly moved on to the more reasonable approach of reconstructing the missing part only.

Figure 3. *Genus-wise distortion [6].* Despite having correctly reconstructed the missing part of the legs and backrest, the model modified part of the chair to resemble its learned ideal of chair.

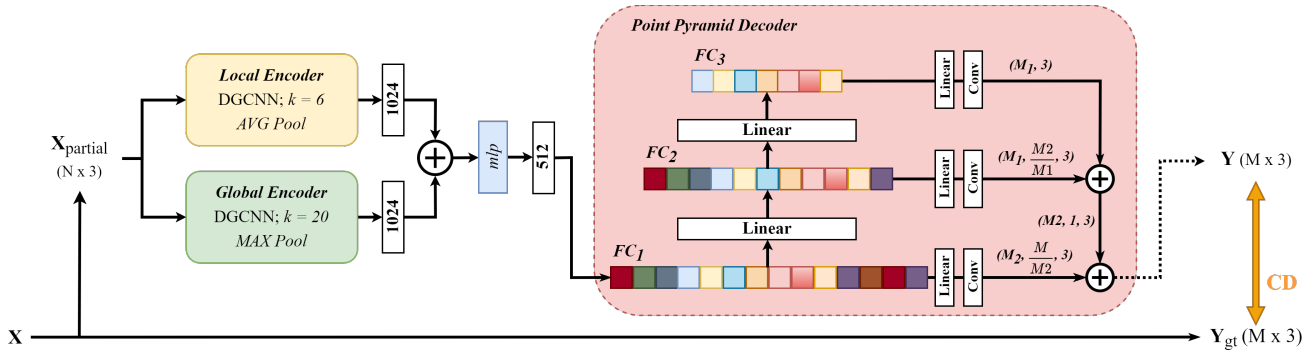

Figure 4. *DGPP completion architecture.* Global and local encoders are both DGCNN-based encoders [4], with the only differences being the chosen k and pooling operation. Their output is summed, downsampled and fed into the PPD decoder [8] that hierarchically reconstructs the missing part at different resolutions and combines by summation the intermediary results into the final output shape.

3.3.2 Definitive Proposal

In the following, we refer with X to the original point cloud, with Xp to the partial point cloud, obtained by removing Ygt from X, and finally with Y to the generated missing patch.
Our architecture takes in input Xp and generates Y, with the loss (CD) being computed between Y and Ygt. We designed DGPP taking inspiration from DeCo [5] and PF-Net [8]. Specifically, as can be seen in figure 4, our architecture has two parallel encoders, meant to capture features respectively at a global and local scale, and they both use the same DGCNN-based encoder [4] design we had previously used for reconstruction, the differences between the two encoders lie exclusively on the two different values of k and the two different pooling operations they adopt, with the global encoder using k=20 and max pooling, and the local encoder using k=6 and avg pooling. We chose avg pooling for the local encoder based on both theoretical assumptions, being it better than max pooling at preserving locality, and on numerical results, as it quantitatively led to a major leap in performances during our ablations research. The two encodings are then summed and fed into the Point Pyramid Decoder described in PF-Net [8].

3.3.3 Local Denoising

Further taking inspiration from DeCo [5], we decided to propose our own formulations of possible self-supervised pretext tasks that could lead to an additional boost in performances. For the local encoder pre-training, we simply adapted the denoising idea used in DeCo [5] by repurposing our reconstruction framework. Specifically, the local encoder used in DGPP is taken by itself and used in a reconstruction auto-encoder, whose decoder part is then dropped. The input point clouds are randomly noised (applying per-point jiggle) but the loss function is computed against the original, clean, point clouds.

3.3.4 Global Jigsaw Puzzle Solving

For the global encoder pre-training instead, we wanted to explore new possibilities and avoid the solutions already proposed in DeCo [5], like contrastive learning [12] and classification. After considering many different options, we settled for the problem of Jigsaw Puzzle Solving, applying it to point clouds as had been done in [10]. Differently from [10] though, we propose it as a mere pretext task, rather than an auxiliary one. Specifically, the global encoder used in DGPP is taken by itself and a part-segmentation problem is set up. Figure 5 offers a step-by-step visualization:

- (a) the point cloud is split into voxels along the axes
- (b) the voxels are randomly shuffled
- (c) the network outputs the predicted per-point segmentation
- (d) the points are moved back to the predicted voxel

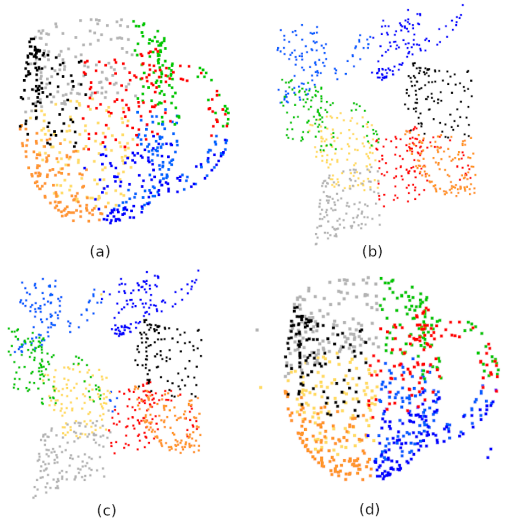

Figure 5. *Jigsaw puzzle solving [10] on point clouds.*

4. Experiments

4.1. Dataset

To train and evaluate our models on known classes, a subset of the Shapenet-Part [9] dataset is selected. In particular, we are only interested in 7 categories, respectively Airplane, Chair, Table, Lamp, Car, Motorbike and Mug. The total number of point clouds is 14550 where 10483 are used for training, 1641 for validation and 2426 for testing
The novel categories instead are extracted from the Shapenet-Core [1] dataset, and they are used to test how our models behave when trying to reconstruct/complete categories never seen during training. We further split this dataset in categories that are similar (basket, bicycle, bowl, helmet, microphone, rifle and watercraft) and dissimilar (bookshelf, bottle, clock, microwave, pianoforte, telephone) to our known categories. All the point clouds are randomly sampled with replacement to 1024 points.

4.2. Reconstruction results

Quantitative analysis In Table 1 we report quantitative results obtained by different versions of our reconstruction pipeline. In particular, we exclude the results obtained with PointNet [2], as we considered it under-performing, in favor of PointNet+1, that nearly matches and sometimes surpasses the DGCNN-based encoder [4].

Qualitative analysis Figure 6 and 7 show respectively some examples of failure and success cases.

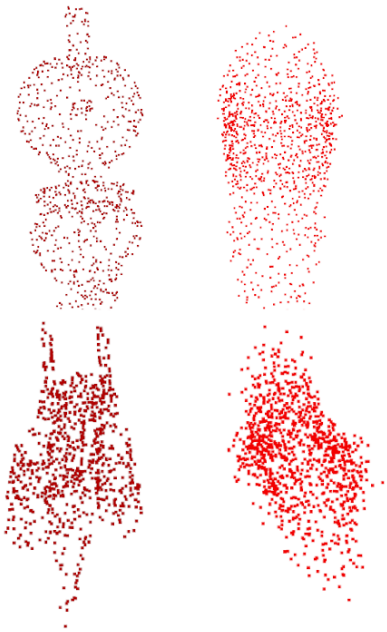

Figure 6. *Reconstruction - Failure cases.*

Encoder	PointNet+1	DGCNN [4]	DGCNN [4]
Augmentation	on - off	on - off	off
	Known categories		
Airplane	**0.911**	0.971	0.913
Chair	1.757	1.766	**1.731**
Table	1.977	**1.953**	1.99
Lamp	**2.332**	2.413	2.423
Car	2.417	2.446	**2.414**
Motorbike	**1.781**	1.831	1.847
Mug	3.737	**3.429**	3.572
	Similar novel categories		
Basket	4.726	**4.432**	4.74
Bicycle	**3.391**	3.618	3.718
Bowl	6.289	**4.529**	5.349
Helmet	**5.975**	6.12	6.07
Microphone	3.243	**3.014**	3.842
Rifle	**1.247**	1.497	1.346
Watercraft	**2.074**	2.113	2.163
	Dissimilar novel categories		
Bookshelf	**4.507**	4.531	4.914
Bottle	**2.322**	2.539	2.405
Clock	4.122	**4.067**	4.79
Microwave	4.495	**4.199**	4.437
Pianoforte	**5.107**	5.41	5.589
Telephone	2.752	**2.726**	3.159

Table 1. *Reconstruction - Quantitative results.* CD between the original and reconstructed point clouds, scaled by 10^3. The lower, the better. We also report the results of training without data augmentation, showing how our mixed augmentation/no-augmentation method improves performance on novel categories.

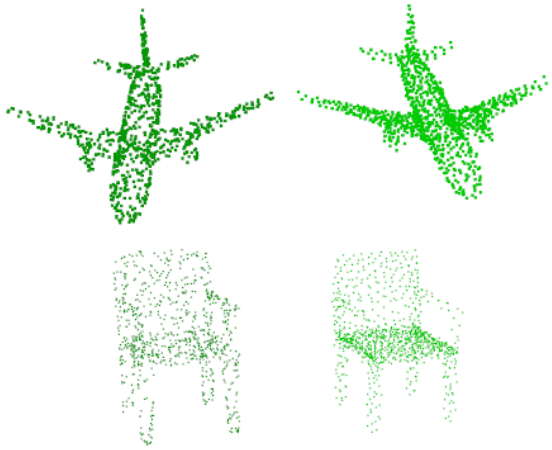

Figure 7. *Reconstruction - Success cases.*

4.3. Completion results

Quantitative analysis In table 2 we report the results obtained by testing the different ablations of our model with and without the various pretext tasks. To ensure a fair comparison, the ablations were tested at the same point in training, specifically after 60 epochs each, with the respective pretext task also previously trained for 60 epochs. While this number of epochs is generally small for such a comparison, we can see that there are still noticeable differences in performances among the various ablations.

The idea of combining denoising and puzzle-solving only worked partially. In fact, when we chose puzzle-solving as our global pretext task, we did not realize that it is not a category-agnostic method, and as a result the model learned to complete very well only the known categories, and some of the similar novel categories, while under-performing in completion of dissimilar novel categories.

In table 3 instead, we report the results obtained on each class after having extensively trained our final model, and we offer a "qualitative quantitative" comparison with PF-Net [8] and DeCo [5], the two models that mostly influenced our work. It is of fundamental importance to mention that the results of PF-Net [8] and DeCo [5] were taken from [5], and they refer to completion of 512-points missing patches, while we only focused on 256-points missing patches.
As such they are not meant in any way to be a basis for a reliable comparison, but simply give the reader an idea of the scale of the loss involved, and where our model might place on this scale. We would have liked to validate those results but could not, mostly because of temporal constraints.

Qualitative analysis In figure 8 we show some examples of completion obtained by our final model on three different categories, Airplane and Lamp from the known classes, and Bowl from the unknown classes.

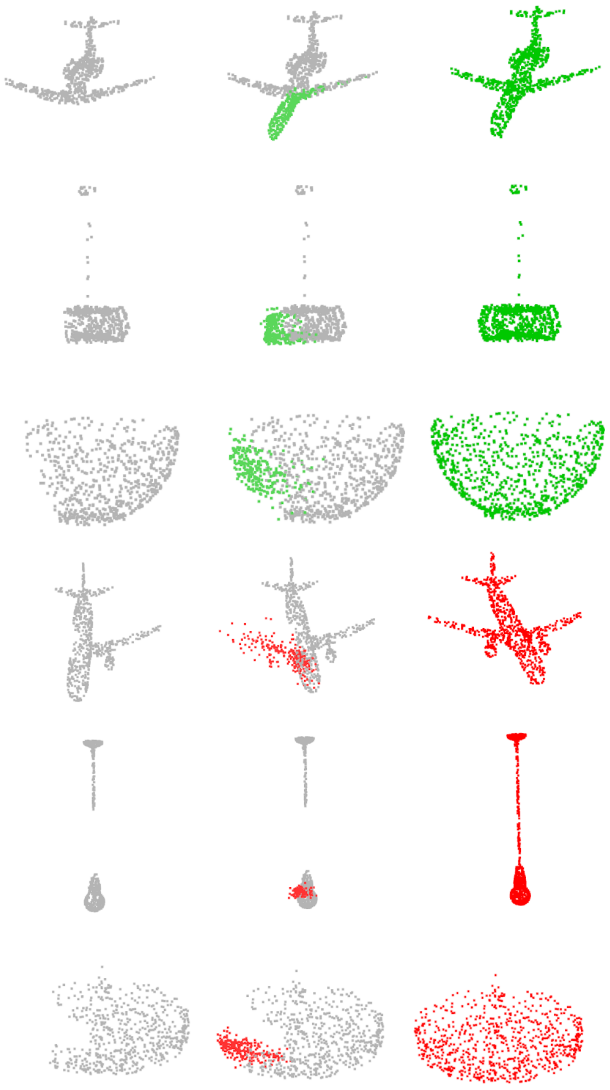

Figure 8. *Completion - Success and failure cases on the Airplane, Lamp and Bowl categories.*

5. Conclusion

In this work we tackled the problems of point cloud reconstruction and completion, by employing deep neural networks modelled around the auto-encoder architectural pattern. We focused particularly on the completion problem, and took inspiration from recent works to design our version of a possible completion architecture, which we called DGPP as in Double-Graph Point-Pyramid. We then explored possible self-supervised solutions to be used as pretext tasks to pre-train part of the model, further confirming the findings in [5] that this approach can undoubtedly lead to better performances in the downstream task.

Local denoise	✗	✓	✗	✓
Global puzzle	✗	✗	✓	✓
Known categories				
Airplane	1.768	1.4668	1.477	**1.4594**
Chair	2.766	2.7758	2.7801	**2.6833**
Table	3.0914	2.985	3.0791	**2.9313**
Lamp	4.125	4.0212	4.1476	**3.697**
Car	3.2072	**2.9465**	3.0398	2.9572
Motorbike	2.8883	**2.5793**	2.7455	2.7691
Mug	4.5248	4.6073	4.7339	**4.5159**
Similar novel categories				
Basket	6.0796	6.0449	6.5735	**5.5998**
Bicycle	5.0822	**4.87**	4.8962	5.2342
Bowl	6.828	6.6762	6.9175	**6.2171**
Helmet	**7.7392**	8.1916	8.0857	8.0513
Microphone	**5.847**	6.8263	6.0154	6.9745
Rifle	3.2138	2.8823	3.3449	**2.8738**
Watercraft	3.3253	3.1292	3.236	**3.1141**
Dissimilar novel categories				
Bookshelf	6.2904	**6.0075**	6.3222	6.0571
Bottle	3.7276	3.8067	3.8981	**3.6368**
Clock	**6.2864**	6.3552	7.071	6.5291
Microwave	5.9927	5.8782	**5.73**	5.9273
Pianoforte	**7.6422**	7.8413	7.7479	8.0778
Telephone	**4.3733**	4.6652	5.2738	4.747

Table 2. *Completion - Quantitative results between ablations.* CD between the originally removed patch and the reconstructed missing patch, scaled by 10^3. The lower, the better. The ablations were tested at the same point in training.

Categories	PF-Net [8]	DeCo [5]	DGPP
Known categories			
Airplane	1.0805	1.0003	1.2506
Chair	1.9490	1.6428	2.2191
Table	2.0845	1.7120	2.4399
Lamp	4.2910	2.4150	3.0785
Car	2.1640	2.2963	2.668
Motorbike	1.9905	1.9136	2.3041
Mug	3.1880	3.4239	4.0199
Similar novel categories			
Basket	5.7066	3.4613	5.4119
Bicycle	4.7186	3.9684	4.6776
Bowl	7.8793	3.5209	5.5302
Helmet	6.9849	4.7412	7.2699
Rifle	2.8684	1.2004	3.0659
Dissimilar novel categories			
Bookshelf	5.5123	3.4681	5.562
Bottle	2.4578	2.0002	3.2382
Clock	4.8373	3.2826	6.2171
Microwave	5.6152	4.1877	4.9353
Pianoforte	6.2994	4.9429	7.7373
Telephone	3.2063	2.0106	4.0084

Table 3. *Completion - Quantitative results between our model, and its reference models.* DeCo [5] and PF-Net [8] results taken from [5]. CD between the originally removed patch and the reconstructed missing patch, scaled by 10^3. The lower, the better. These results are only meant to give the reader an idea of the scale of the loss involved, but should not in any way be considered a reliable basis for comparison as we could not validate them ourselves.

References

[1] 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. 1, 2, 5

[2] C. R. Qi, H. Su, K. Mo and L. J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation, 2016. 1, 2, 3, 5

[3] Charles R. Qi, Li Yi, Hao Su and Leonidas J. Guibas. 2017c. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. In Proc. Adv. Neural Inf. Process. Syst., 2017. 2, 3

[4] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein and Justin M. Solomon. Dynamic graph cnn for learning on point clouds. ACM Transactions on Graphics (TOG), 2019. 1, 2, 3, 4, 5

[5] Antonio Alliegro, Diego Valsesia, Giulia Fracastoro, Enrico Magli and Tatiana Tommasi. Denoise and Contrast for Category Agnostic Shape Completion, 2021 2, 3, 4, 6, 7

[6] Yaoqing Yang, Chen Feng, Yiru Shen and Dong Tian. Foldingnet: Point cloud auto-encoder via deep grid deformation. CVPR, 2018. 2, 3

[7] Wentao Yuan, Tejas Khot, David Held, Christoph Mertz and Martial Hebert. Pcn: Point completion network. In 3DV, 2018. 2

[8] Zitian Huang, Yikuan Yu, Jiawen Xu, Feng Ni and Xinyi Le. Pf-net: Point fractal network for 3d point cloud completion. In CVPR, 2020. 2, 3, 4, 6, 7

[9] Li Yi, Vladimir G. Kim, Duygu Ceylan, I-Chao Shen, Mengyan Yan, Hao Su, Cewu Lu, Qixing Huang, Alla Sheffer, and Leonidas Guibas. A scalable active framework for region annotation in 3d shape collections. ACM Trans. Graph., 35(6), Nov. 2016. 2, 5

[10] Antonio Alliegro, Davide Boscaini and Tatiana Tommasi. Joint Supervised and Self-Supervised Learning for 3D Real-World Challenges, 2020. 4

[11] Minghua Liu, Lu Sheng, Sheng Yang, Jing Shao and Shi-Min Hu. Morphing and Sampling Network for Dense Point Cloud Completion, AAAI 2020. 2

[12] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In ICML, 2020. 2, 4

[13] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, ICLR, 2015. 8

6. Supplementary material

The Pytorch code produced all throughout this project can be found at https://github.com/ozerodb/aml-pointcloud-recon-pletion, together with a spreadsheet containing all the results of each and every thing we tried, and some utility scripts to ease the training and testing processes.

6.1. Training Details

All our models were trained using the Adam optimizer [13] with a learning rate of 0.001 and no scheduler. The batch size was set to 32 during the reconstruction trainings, and lowered to 16 when dealing with completion, because of the additional memory required by the larger models.