3D Object Part Segmentation using Self-supervised Learning

Kiray, Mert
mert.kiray@tum.de

Karali, Merve
merve.karali@tum.de

Sen, Umur
umur.sen@tum.de

Calik, Nazmican
nazmican.calik@tum.de

Abstract

In this work, we show that self-supervised learning can achieve competitive results compared to supervised training on 3D object part segmentation task with a smaller amount of labels. We show this by adapting the contrastive learning framework SimCLR. SimCLR enables us to form a descriptive embedding space without using the ground truths. Furthermore, the augmentations used in the SimCLR framework improves the rotation invariance of the model. We evaluate our methods on ShapeNet with class average IoU.

1. Introduction

3D object part segmentation is a challenging yet widely performed task. Point clouds and meshes are the most common data types used for part segmentation. PointNet [7] is the pioneer model that uses point clouds for 3D object part segmentation. Supervised deep learning models require a vast amount of labeled data to generalize well. Finding labeled data is hard and expensive, especially for 3D objects. On the other hand, unlabeled data is easy to access and cheap. Nowadays one can even scrape the web to find a vast amount of unlabeled data. The recent developments on self-supervised learning shows great promise to achieve good results with limited amount of labels [4]. In this work, we adapt the state-of-the-art self-supervised learning framework SimCLR [4, 5] to learn useful 3D visual representations with contrastive learning and achieve reliable results with a limited amount of annotations. To boost the performance of contrastive learning, we take advantage of various data augmentation; reflection, rotation, scaling, flipping, cutout and additive Gaussian noise. With these augmentations we create a model that is invariant to various transformations such as rotation.

2. Related Work

Self-supervised Learning: Self supervised learning is a learning scheme where the intrinsic features of data is learned using only the data itself. No labeling or supervision is needed. Language models have been exploiting this phenomena when training models such as BERT [6] on corpus to understand languages better. However, for computer vision it is comparatively new and we used SimCLR [4, 5] as our framework because it provides a suitable environment for applying self-supervised learning on other types of data.

Deep Learning on 3D Point Clouds: Point clouds differ greatly from other type of representations of 3D data because they do not carry a structure. This very reason makes them hard to work with arbitrary and simpel neural architectures. Pointnet [7] solves this with a permutation invariant approach. We chose PointNet for its simplicity and used it to apply self supervised learning on 3D point cloud data.

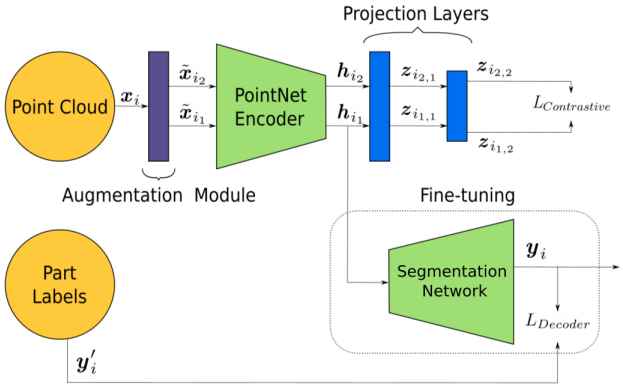


Figure 1: Self-supervised learning framework architecture for 3D object part segmentation. It consists of two stages: (1) contrastive learning of the encoder and the projection layers to extract object representation vector $\boldsymbol{h}$ from the input point cloud $\boldsymbol{x}$, and (2) a supervised fine-tuning of the segmentation network to predict segmentation class $\boldsymbol{y}$.

3. Method

The framework consists of three major components as presented in Figure 1; (1) a PointNet encoder to extract visual representations from the 3D point cloud objects, (2) projection layers for representations vector mapping in self-supervised training and (3) a PointNet segmentation net-

work for 3D object part segmentation. While the encoder and projection layers are trained with self-supervised learning using unlabeled data, the segmentation network is fine-tuned using the labeled data.

3.1. Self-supervised Encoder Training

SimCLR learns object representations by maximizing the agreement between two augmented versions of the same visual data. The contrastive learning of SimCLR consist of four major steps as presented in Figure 1:

1. A stochastic data augmentation module transforms the 3D visual input $\boldsymbol{x}$ into two different visual representations $\tilde{\boldsymbol{x}}_i$ and $\tilde{\boldsymbol{x}}_j$. Two different augmentation sets are sampled from the same stochastic augmentation tree and applied on the visual input $\boldsymbol{x}$. In this project, we select random cuboid, Gaussian noise and rotation augmentations for the module. As presented in Section 3.2, the selected augmentations are essential to extract good representations and to be invariant for different 3D object representations such as rotated inputs.

2. An encoder extracts the representations from the augmented 3D visuals. The framework does not contain any restrictions for the base encoder. This design choice of SimCLR allows us to focus on the augmentation techniques without designing a specialized architecture [1] or a memory bank [9]. We select the encoding part of PointNet as the encoder because it presents promising results for the part segmentation task [7].

3. The representation vectors are mapped into a smaller latent space $\boldsymbol{z}$ by a projection head to perform contrastive learning. The projection head is a multilayer perceptron (MLP) which consist of two fully connected layers, a nonlinear activation function (ReLU) and batch normalization. Projections are necessary only for the contrastive learning because contrastive learning benefits from smaller latent space [4]. These layers are removed in the fine-tuning part and $\boldsymbol{h}$ is collected as the feature vector.

4. The networks are trained with NT-Xent (the normalized temperature-scaled cross entropy) loss [4]. On each iteration over the projected representations, the contrastive task is to identify the positive pairs which are the two representations that belongs to the same visual input

$$\mathcal{L}_{i,j} = -\log \frac{\exp(\text{sim}(\boldsymbol{z}_i, \boldsymbol{z}_j)/\tau)}{\sum_{k=1}^{2n} \mathbf{1}_{[k\neq i]} \exp(\text{sim}(\boldsymbol{z}_i, \boldsymbol{z}_k)/\tau)}, \quad (1)$$

where τ is the temperature hyperparameter, $\mathbf{1}_{[k\neq i]}$ is the indicator function and $\text{sim}(.,.)$ is the cosine similarity function

$$\text{sim}(\boldsymbol{z}_i, \boldsymbol{z}_j) = \frac{\boldsymbol{z}_i^\top \boldsymbol{z}_j}{\|\boldsymbol{z}_i\| \|\boldsymbol{z}_j\|}. \quad (2)$$

The normalized representations are mapped into a unit sphere. The aim is to map the positive pairs to the similar direction, while mapping the negative samples in a further direction.

3.2. Fine-tuning the Segmentation Network

The goal of fine-tuning is to train the segmentation network to predict 3D object part classes from the extracted visual representations. Since this process is a supervised training, it includes object part labels as well as the point cloud data.

Similar to the self-supervised stage, we transform the input data with stochastic augmentation module, and then extract the representation vector by using the encoding part of PointNet as encoder. In this training, we remove the projection layers because they are only necessary for the latent space mapping in self-supervised learning. The extracted representation vector $\boldsymbol{h}$ enters segmentation network to predict the per point features $\boldsymbol{y}$. During this process, we freeze the PointNet encoder model in order to train only the PointNet segmentation network.

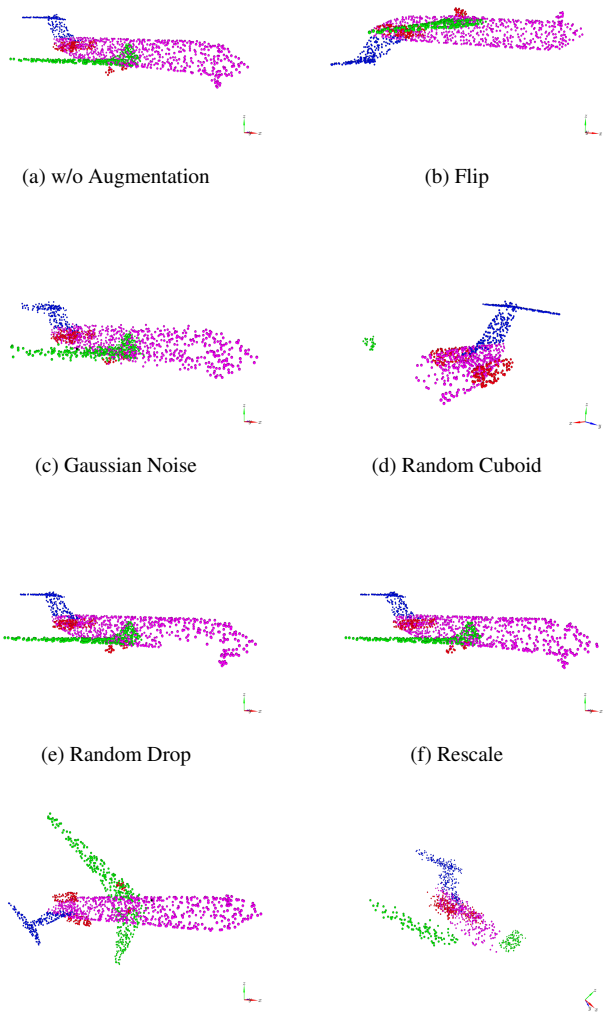
4. 3D Data Augmentations for Contrastive Representation Learning

Combining multiple data augmentation is crucial for contrastive learning to yield effective representations [4]. For this reason, we introduce several data augmentation techniques; flip, Gaussian noise, rescale, and rotation. On top of these augmentation techniques, we follow [10] to use random cuboid and random drop. Visualizations of these augmentation techniques can be seen in Figure 2. Input point cloud for these augmentations can be seen from Figure 2a. In this section we explain our data augmentation techniques:

- ***Flip***. For a given point cloud we flip the points along the XZ or YZ plane randomly. An example can be seen from Figure 2b.

- ***Gaussian noise***. For each point cloud we jitter the position of each point by a Gaussian noise with 0 mean and 0.0075 standard deviation. An example can be seen from Figure 2c.

- ***Random cuboid***. We follow [10] to extract random cuboids from the input point cloud. An example can be seen from Figure 2d.



(a) w/o Augmentation (b) Flip

(c) Gaussian Noise (d) Random Cuboid

(e) Random Drop (f) Rescale

(g) Rotation (h) Combined Augmentation

Figure 2: Augmentations

- ***Random drop***. We follow [10] to drop random cuboids from the input point cloud. An example can be seen from Figure 2e.
- ***Rescale***. We scale each point in point cloud by uniform distribution. An example can be seen from Figure 2f.
- ***Rotation***. We rotate the point cloud randomly in X, Y or Z plane. An example can be seen from Figure 2g.

Figure 3 presents the fine-tuning test results under individual and composition of augmentations. The presented augmentations are applied in both self-supervised training and fine-tuning stages. Random cuboid is defined as a preliminary augmentation in all self-supervised trainings

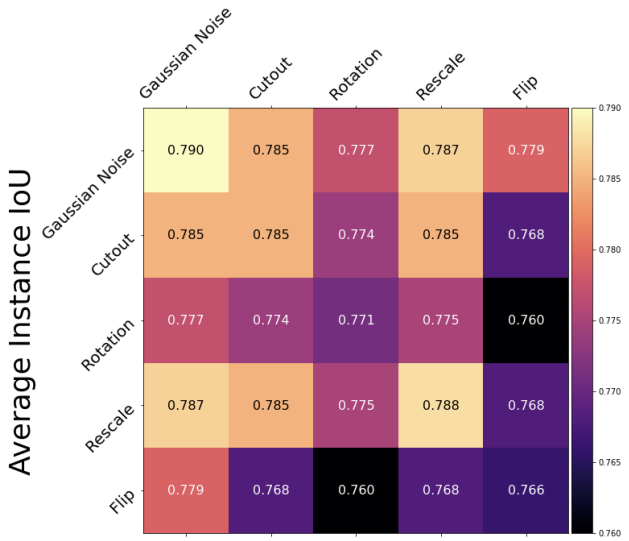


Figure 3: Fine-tuning average instance IoU test results of single and composition of augmentations.

so it is not separately evaluated in the experiments as the other augmentations. There are two reasons for this design choice: (1) SimCLR presents the importance of localization (crop and resize) in contrastive learning, and (2) PointNet++ presents dramatically improved results by introducing the local context information. In the results, we can observe the Gaussian noise achieves the highest scores in both individual and pairwise experiments. Therefore, Gaussian noise is included in the final set of stochastic augmentation module. Additionally, we include the rotation augmentation to make the model rotation invariant.

5. Experiments

5.1. Datasets

Following [7], we used Shapenet [3] to train our architecture. It has in total 16 classes and only three classes have more than 2k samples (5k chair) while several classes has around 50 samples. Our encoder that makes use of SimCLR framework trained without ground-truth labels. The decoder part is then fine-tuned with different fractions of Shapenet respectively: %1, %10, %100.

5.2. Metrics

We utilize various metrics to measure how the separate modules of our network performs. We follow [7] to calculate the class average and instance average IoU. To compare model performances we use class average IoU.

5.3. Experiment Results

Self-supervised learning allows us to achieve good scores where the annotated data is scarce by learning useful representation features from the unlabeled data [4]. In order to experiment the label efficiency of the SimCLR framework, and the performance of the supervised model under insufficient annotations, we limit our dataset labels to one and ten percent for the trainings. The experiment results are presented in Table 1. According to the results, self-supervised learning performs the best when the labels are insufficient. On the other hand, the supervised model without any augmentations included in the training performs better in the case of sufficient data. The results prove that self-supervised learning is label efficient and supervised learning requires a vast amount of labels to achieve reliable scores.

Also as stated in [4], when the supervised model is trained with the same set of augmentations that are used in the self-supervised learning, the performance of the model degrades. This situation is observable from the supervised model results of augmented and classical training in Table 1.

To test the robustness of our method we apply different augmentations to the test set separately, and evaluate three different approaches; namely self-supervised, supervised and supervised with data augmentations. The prediction results of augmented test set is presented in Table 2. The results show that training with augmentations is to achieve a rotation and flip invariant model. Figure 4 presents the supervised and self-supervised inference on a plane class. While the self-supervised model present robust results against rotated inputs, the supervised model fails to generalize. On the other hand, the model is already robust against Gaussian noise and scale operations and requires no augmentation.

6. Ablation Study

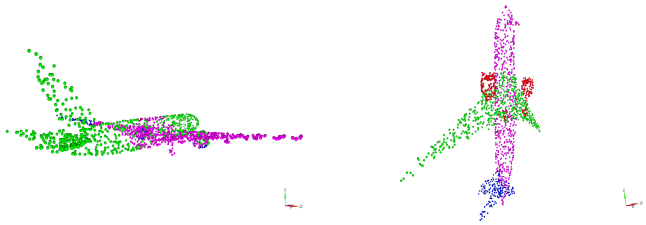
SimCLR benefits from larger batch size [4]. Larger batch size holds more negative samples, thereby increasing the number of compared samples for contrastive loss function. This eventually increases the performance. We experimented on various batch sizes and as the results can be seen in Table 3, the instance and class average IoU values tend to increase even though it is non-monotonously. The results show that bigger batch sizes will lead to even better scores.

Label fraction	Self-supervised	Supervised with augmentations	Supervised w.o augmentations
%1	**69.4**	57.6	61.3
%10	71.5	67.4	**72.4**
%100	78.0	71.1	**80.4**

Table 1: Self-supervised vs supervised performance in terms of class average IoU

Model	Rotation	Flip	Gaussian Noise	Rescale
Self-supervised	**68.8**	49.0	77.3	78.1
Supervised	63.6	41.4	**78.9**	**80.4**
Supervised w/ Augmentations	67.5	**56.0**	70.3	71.2

Table 2: Average class IoU of augmented test set according to augmentation type and model.



(a) Supervised (b) Self-supervised

Figure 4: Inference results on rotated inputs

Batch size	Class Average IoU
16	71.9
32	71.2
64	71.1
128	72.1
256	**72.5**

Table 3: Effect of batch size on SimCLR framework

7. Conclusion

In this work, we show that training a model with self-supervised learning on 3D point cloud for object part segmentation yields reliable results with a limited amount of labels. Furthermore, the self-supervised model presents invariance against 3D rotations. Due to memory limitations, we are unable to reach the optimal batch size (e.g 2048 or 4096) of the SimCLR framework. Our results show that the performance can be further improved with a larger batch size.

References

[1] Philip Bachman, R. Devon Hjelm, and William Buchwalter. Learning representations by maximizing mutual information across views. *CoRR*, abs/1906.00910, 2019. 2

[2] Lukas Biewald. Experiment tracking with weights and biases, 2020. Software available from wandb.com. 4

[3] 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, 2015. 3

[4] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations, 2020. 1, 2, 4

[5] Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey Hinton. Big self-supervised models are strong semi-supervised learners, 2020. 1

[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019. 1

[7] Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation, 2017. 1, 2, 3

[8] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605, 2008. 6

[9] Zhirong Wu, Yuanjun Xiong, Stella X. Yu, and Dahua Lin. Unsupervised feature learning via non-parametric instance-level discrimination. *CoRR*, abs/1805.01978, 2018. 2

[10] Zaiwei Zhang, Rohit Girdhar, Armand Joulin, and Ishan Misra. Self-supervised pretraining of 3d features on any point-cloud. 2021. 2, 3

Appendix A. TSNE

Figure 5: t-SNE applied to embedding space of self-supervised encoder. Objects are clustered according to their classes.

We use the self-supervised encoder to get the embedding space description of our test set. To further investigate the learned representations we reduce the number of dimensions to two with t-SNE [8]. As seen on Figure 5, each object class is assigned to a specific color, and instances are represented as a point. Since the same class objects are clustered together we deduce that our encoder successfully separates different classes of objects according to their intrinsic descriptions.

Furthermore, as can be seen in Figure 6, our encoder clusters a specific class of objects inside itself as well. For example, armchairs and standard chairs are clustered separately inside the general chair group.

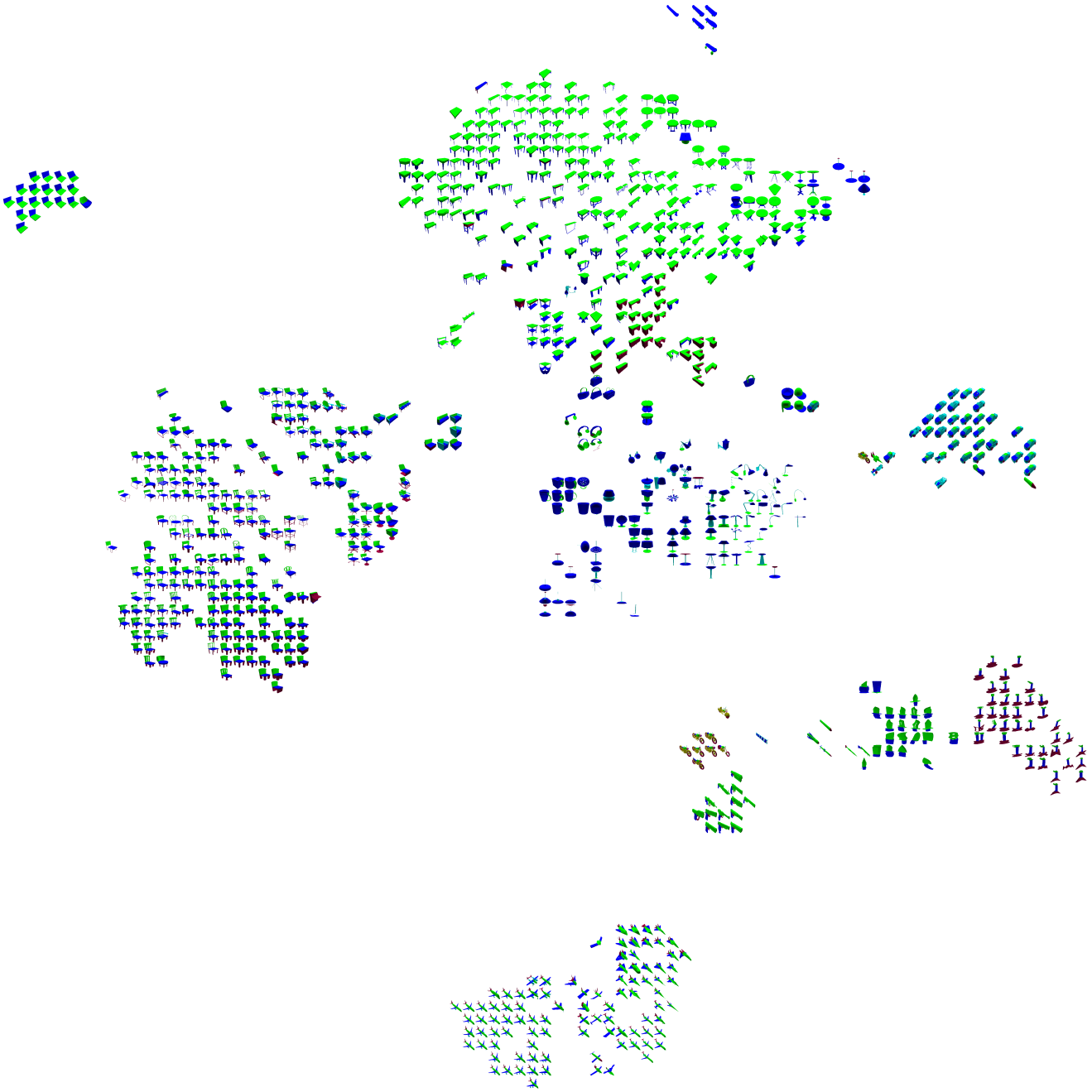


Figure 6: t-SNE applied to the embedding space of the self-supervised encoder on test set. Objects are clustered succesfully according to their classes. Objects are also clustered inside their own class. For example war planes are clustered on the right side of commercial planes as they differ in shape.