SELF-SUPERVISED DEEP LEARNING ON POINT CLOUDS BY RECONSTRUCTING SPACE

Tasneem Bawendi
tb9g16@soton.ac.uk
28281799

Antoine Poullet
ap6u16@soton.ac.uk
28821947

Ahmad Sarvmeily
as20g16@soton.ac.uk
28464842

1 INTRODUCTION

In their paper submitted to NIPS 2019, Sauder & Sievers propose a self-supervised learning task on 3D point cloud data by learning to reconstruct voxel-shuffled point clouds. The method claims to improve performance in downstream object classification tasks. It also learns both per-point representations and per-object representations, and avoids computationally expensive reconstruction losses on point clouds.

The aim of this report is to re-implement the proposed pre-training method, then use the pre-trained models to reproduce a subset of the three experiments outlined in the original paper. The three experiments used a DGCNN using the same setup proposed by Wang et al. (2018). The source code for all three reproduced experiments is available on GitHub[1].

2 EXPERIMENTAL SETUP

2.1 PRE-TRAINING

The self-supervised labels are generated from a dataset of point clouds using the method introduced by Sauder & Sievers. Each point cloud is mean-centred and scaled to unit cube volume. Then, each dimension is split into 3, such that the points are separated into 27 distinct voxels in the unit cube space. Each point is then assigned a voxel ID, $v \in \{0, 1, ..., 26\}$, before all voxels are swapped randomly. These shuffled point clouds, along with the *original* voxel IDs form the self-supervised dataset. The pre-training task is then to correctly identify the original voxel ID of each point by reconstructing the voxel-shuffled shape.

Sauder & Sievers also note better generalisation can be achieved on this task by augmenting the dataset, however the details of this are vague and it is not stated as a mandatory part of the algorithm. Thus, our experiment does not include it.

2.2 EXPERIMENT 1: OBJECT CLASSIFICATION

This experiment analyses the effects of self-supervised pre-training in the context of training a linear SVM on the embeddings of a neural network.

It can be separated into 3 different parts. First, the authors evaluate the embeddings learned by the pre-trained network through a classification task performed on the ModelNet40 dataset (Wu et al., 2015) with a Linear SVM classifier. They then show that the performance of the SVM, and therefore the quality of the embeddings, increases as the self-supervised training loss decreases.

The second part of the experiment involves solving the same classification task, but on varying training set sizes. In doing so, the effect of pre-training on limited dataset sizes is assessed.

The last part of the experiment compares the classification accuracy of a neural network pre-trained using the authors' technique and one that was not. Again, this is done on the ModelNet40 dataset.

[1] `https://github.com/COMP6248-Reproducability-Challenge/Self-supervised-deep-learning-on-point-clouds-by-reconstructing-space`

2.2.1 PRE-TRAINING

The first two parts of this experiment use neural networks pre-trained on point clouds consisting of 2,048 points whilst the last part uses point clouds made out of 1,024 points.

Although they did not explicitly state it, Figure 2a from the authors' paper shows they trained the network in the first part for 100 epochs. As they did not specify a number of epochs for the second part the assumption was made that the training lasted 100 epochs as well. Finally, the authors say the network in the last part was also trained for 100 epochs.

2.3 EXPERIMENT 2: PART SEGMENTATION

The pre-training method is evaluated on a part segmentation problem. Given a 3D point cloud and a provided label of the shape, the task is to identify which part of the shape each point in the cloud belongs to. As with experiment 1, the pre-training method is applied to the ShapeNet V1 dataset (Chang et al., 2015), using the part segmentation variant of the DGCNN (Wang et al., 2018).

2.3.1 PRE-TRAINING & TRAINING

Pre-training is performed over 200 epochs on the generated self-supervised data. The self-supervised voxelisation method only requires point clouds as input, however the architecture of the DGCNN model requires that shape labels are still passed as input to the network. Following Sauder & Sievers, random shape labels are passed to the DGCNN during pre-training.

After pre-training for 200 epochs on the self-supervised dataset generated from ShapeNet v1, the DGCNN is trained for 200 epochs on the ShapeNet Part dataset (Yi et al., 2016). In order to correct the number of outputs from 27 to 50, the last convolutional layer of the pre-trained network is re-instantiated with 50 output channels, one for each part label, before training on ShapeNet Part.

2.4 EXPERIMENT 3: SEMANTIC SEGMENTATION

In this experiment, the pre-training method is applied to a semantic segmentation task using the Stanford Large-Scale 3D Indoor Spaces (S3DIS) dataset (Armeni et al., 2016). This dataset contains point clouds for six indoor areas, containing 272 rooms (e.g. pantry1, office3) and 13 semantic classes for items in the rooms (e.g. floor, table, clutter). Rooms are split into 1m x 1m blocks, with points given as 6D vectors of XYZ coordinates and RGB colour codes. Although we attempted to, due to memory constraints, we were unable to reproduce the semantic segmentation experiment; it required 16GB-24GB of VRAM which we did not have access to.

3 IMPLEMENTATION DETAILS

All experiments were implemented using PyTorch instead of TensorFlow (which was used by the authors). An NVIDIA RTX 2070 was used for pre-training, and an NVIDIA Tesla P100 for training. The authors did not specify the hardware used, but investigation of the code revealed that they used two 12GB unspecified GPUs.

The PyTorch implementation of the DGCNN network architecture was taken from `https://github.com/AnTao97/dgcnn.pytorch`, which is referenced from the official GitHub repository for the paper introducing the DGCNN (Wang et al., 2018) at `https://github.com/WangYueFt/dgcnn`.

Many implementation details of the original experiments are not present in the paper. These include the optimiser, learning rate scheduler, batch size, number of epochs and the loss function. Instead these were inferred from the code provided by the authors, or from the DGCNN repositories mentioned above. The code for the paper was not referenced in the paper itself, but the NIPS2019 submission contained the code used by the authors. For all experiments and pre-training, we used the Adam optimiser, a batch size of 32 and cross-entropy loss with label smoothing ($\epsilon = 0.2$). For pre-training, an initial LR of 0.0001 is used, with a Multiplicative scheduler ($\gamma = 0.5$, $lr_{min} = 0.00001$). For experiments, an initial LR of 0.001 is used, with a Cosine Annealing scheduler ($\eta_{min} = 0.001$).

	Mean	Aero	Bag	Cap	Car	Chair	Earphone	Guitar	Knife	Lamp	Laptop	Motor	Mug	Pistol	Rocket	Skateboard	Table
# Shapes		2690	76	55	898	3758	69	787	392	1547	451	202	184	283	66	152	5271
DGCNN	85.1	**84.2**	83.7	84.4	**77.1**	**90.9**	78.5	**91.5**	87.3	82.9	**96.0**	67.8	93.3	**82.6**	59.7	75.5	82.0
DGCNN+P	**85.3**	84.1	**84.0**	**85.8**	77.0	**90.9**	**80.0**	**91.5**	87.0	**83.2**	95.8	**71.6**	**94.0**	82.6	**60.0**	**77.9**	81.8
Ours (M)	67.7	80.5	51.6	67.7	74.0	82.2	53.0	90.5	**87.9**	78.7	**96.0**	21.9	92.6	76.9	28.2	67.8	**83.1**
Ours (CA)	72.1	81.1	54.0	75.3	74.2	82.7	55.6	88.3	86.4	82.2	95.3	42.9	92.0	73.0	59.6	67.0	77.5

Table 1: Part segmentation accuracy, calculated using mean Intersection over Union (mIoU%). mIoU for a shape is calculated by averaging the IoUs for each part in that shape. The mean value in the first column is calculated by averaging the IoUs across *all 50 parts*. DGCNN+P refers to the pre-trained model as reported by Sauder & Sievers. We report two variants of our pre-trained implementation, using different learning rate schedulers: M (Multiplicative LR), CA (Cosine Annealing).

4 Results

4.1 Experiment 1: Object Classification

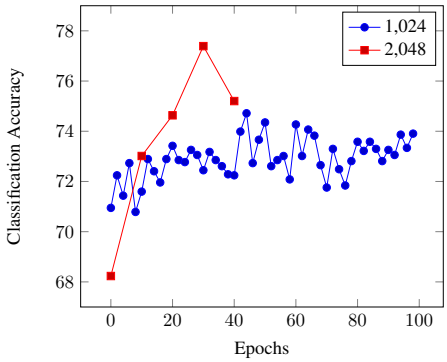

Figure 1: Part 1 results.

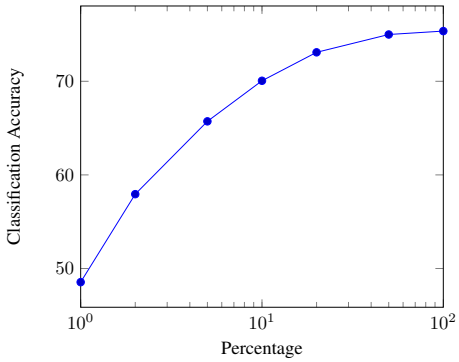

Figure 2: Part 2 results.

4.2 Experiment 2: Part Segmentation

Many of the reproduced IoUs are lower than those reported by Sauder & Sievers. The most significant differences can be observed for the shapes with the fewest labelled data points. Our cosine annealing variant performs significantly better than its multiplicative LR counterpart. Full results are shown in Table 1.

Computational Requirements - We found that the training process for the part segmentation problem required 3.1GB of RAM and 12.7GB of VRAM. When tested on an NVIDIA Tesla P100 16GB GPU, the time taken to perform 200 epochs was $\sim$10.8 hours.

5 Analysis & Evaluation

5.1 Experiment 1

The results obtained for experiment 1 were worse than presented in the paper. Due to confusion, we first pre-trained the network for the first part of the experiment on point clouds that contained 1,024 points instead of 2,048. Due to time constraints we were not able to gather enough data points with the correct version of the pre-training. The results of this first part can be seen in figure 1. The performance of the SVM using the embeddings of the network that was pre-trained on 1,024 points point clouds is noticeably worse, but even when using a network that was correctly pre-trained the performance is not comparable with the results presented in the paper. This reason as to why are not clear. As mentioned earlier, a lot of assumptions had to be made due to the lack of specificity in both the paper and the code. In fact, the code does not have a single example of how the authors used an SVM in their experiments.

The second part of experiment 1 was conducted using the network that was pre-trained on 2,048 points. As mentioned before, time constraints kept us from completing the training and therefore

the performance may have suffered as can be seen in figure 2. However, we can notice that even when using 10% of the data the performance resembles the performance we get using 100% of it. While the absolute performance differs from the original paper, the accuracy improvement is similar.

The last part of the experiment was not attempted as the authors did not provide clear guidance on how to modify the DGCNN network that was pre-trained on the self-supervised task to perform classification.

The first part of experiment 1 took $\sim$ 17 hours to execute and the second part took $\sim$ 30 minutes.

5.2 Experiment 2

The results obtained in Table 1 are significantly worse than those reported in the original paper. There may be several reasons for this. Firstly, the learning rate scheduler used for our pre-training was found to be incorrect. Instead of a Multiplicative LR scheduler, a Step LR scheduler should have been used with a step size of 20. This would have had the effect of reducing the learning rate over a larger range of epochs, which may have led to improved loss minimisation. Another factor may have been the lack of self-supervised augmentation, which was omitted from our algorithm because it was assumed to be optional. However, this step may have been crucial to the effectiveness of the method.

However, neither of these factors explain why the performance was also significantly worse than a randomly initialised DGCNN, unless the incorrect pre-training was detrimental to the supervised learning process. To the best of our knowledge, the rest of the experiment was implemented as described by the authors.

6 Conclusion

Implementing the pre-training method as described in Sauder & Sievers (2019) was not as straightforward as the authors proposed. The augmentation step lacked details, with only point shifting being suggested, and looking at the code we found they were also performing other operations such as random shape rotations. Furthermore, a lack of clarity as to which versions of the datasets were used, along with missing implementation details made it extremely difficult to reproduce.

We did not achieve the same results as the original paper, in either of the two experiments that were carried out. Part of this was due to misinterpretations of vague implementation details and a general lack of clarity in the original report. Thus, we conclude that the paper is not reproducible.

References

Iro Armeni, Ozan Sener, Amir R. Zamir, Helen Jiang, Ioannis Brilakis, Martin Fischer, and Silvio Savarese. 3d semantic parsing of large-scale indoor spaces. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*, 2016.

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.

Jonathan Sauder and Bjarne Sievers. Self-supervised deep learning on point clouds by reconstructing space, 2019.

Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon. Dynamic graph cnn for learning on point clouds, 2018.

Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1912–1920, 2015.

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. *SIGGRAPH Asia*, 2016.