Appendix for "Masked Autoencoders for Point Cloud Self-supervised Learning"

Yatian Pang[2] Wenxiao Wang[3] Francis E.H. Tay[2]
Wei Liu[4] Yonghong Tian[1,5] Li Yuan[1,5⋆]

[1] School of Electronic and Computer Engineering, Peking University
[2] National University of Singapore
[3] Zhejiang University
[4] Tencent Data Platform
[5] PengCheng Laboratory
yatian_pang@u.nus.edu; yuanli-ece@pku.edu.cn

1 Transformer

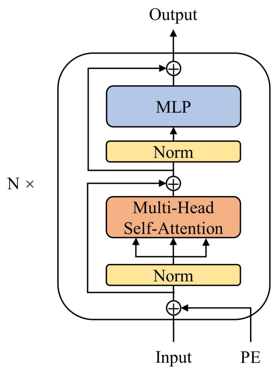

Fig. 1. Transformer architecture. A Transformer block mainly consists of a multi-head self-attention layer and an MLP layer.

As shown in Figure 1, a Transformer [7] block contains a Multi-Head Self-Attention (MSA) layer and an MLP layer. Layernorm (LN) is applied before each layer. Residual connections are applied to both MSA layer and MLP layer. In our work, we add positional embedding (PE) to the input in every Transformer block.

Multi-Head Self-Attention A Self-attention (SA) [7] function maps the input sequence into queries (Q), keys (K), and values (V). Then, the output is

⋆ Corresponding author

computed as a weighted sum over the values, where the weights are computed by the similarity between corresponding queries and keys. Formally, given an input sequence $x \in \mathbb{R}^{N\times C}$, SA is computed as,

$$(Q, K, V) = xW_{qkv} \qquad W_{qkv} \in \mathbb{R}^{C\times 3D}, \tag{1}$$

$$A = softmax(QK^T/\sqrt{D}) \qquad A \in \mathbb{R}^{N\times N}, \tag{2}$$

$$SA(x) = AV. \tag{3}$$

Due to the SA function, Transformer has a quadratic complexity in terms of the length of input sequence.

MSA computes k self-attention functions in parallel and projects their concatenated outputs,

$$MSA(x) = (SA_1(x); SA_2(x); \dots; SA_k(x))W_{msa} \quad W_{msa} \in \mathbb{R}^{kD\times C}. \tag{4}$$

Here, D is typically set to C/k to keep parameter constant.

MLP The MLP consists of two linear layers with GELU activation functions. Typically, both the input and output of the MLP have the same number of dimensions. The hidden dimensions are multiplied input dimensions by a ratio, which is referred as the MLP ratio.

2 Point-MAE Experiments Details

2.1 Model details

In our Point-MAE, for different resolutions of the input point cloud, we divide them into different numbers of patches with a linear scaling. A typical input with $p = 1024$ points is divided into $n = 64$ point patches. For the KNN algorithm, we set $k = 32$ to keep the number of points in each patch constant. In the autoencoder's backbone, the encoder has 12 Transformer blocks while the decoder has 4 Transformer blocks. Each Transformer block has 384 hidden dimensions and 6 heads. MLP ratio in Transformer blocks is set to 4. For downstream tasks, the decoder is discarded.

Pre-training We pre-train our model on ShapeNet [1] training set. ShapeNet consists of about 51,300 clean 3D models, covering 55 common object categories. We split the dataset into a training set and a validation set but only conduct pre-training on the training set. For each instance, we sample 1024 points via FPS as input point cloud. Note that we only apply standard random scaling and random translation for data augmentation during pre-training. For pre-training details, we use an AdamW optimizer [3] and cosine learning rate decay [2]. The initial learning rate is set to 0.001, with a weight decay of 0.05. We pre-train our model for 300 epochs, with a batch size of 128.

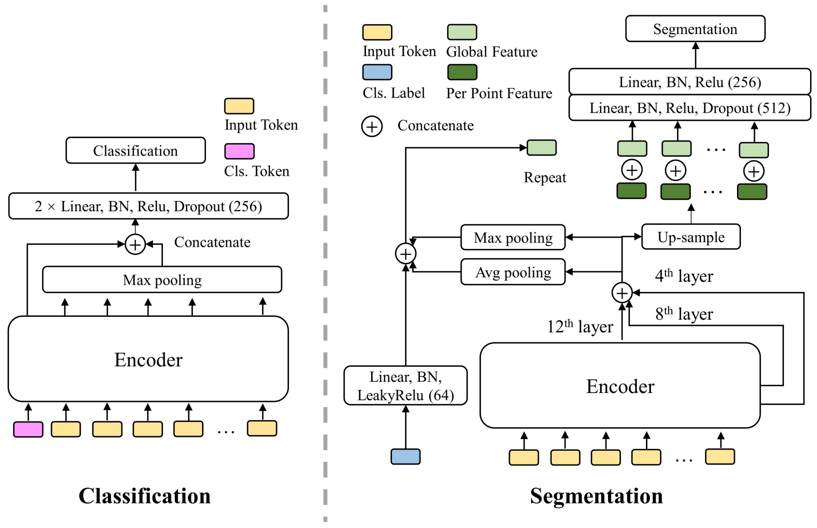

Fig. 2. Fine-tuning models. For classification tasks and few-shot learning, the model is shown on the left. For the part segmentation, the model is shown on the right.

Classification As shown in Figure 2 left, for classification tasks, we add a classification token to the input tokens. Taking the processed input tokens, we adopt a max pooling operation and concatenate the resulted feature with the processed classification token. Then, the concatenated feature is fed to an MLP to complete classification. BatchNorm, RELU activation, and Dropout with a ratio of 0.5 are adopted in each layer of MLP.

As for classification task on ModelNet40 [8], the dataset consists of 12,311 clean 3D CAD models, covering 40 object categories. We follow standard protocols to split ModelNet40 into 9843 instances for the training set and 2468 for the testing set. Standard random scaling and random translation are applied for data augmentation during training.

Few-shot learning In the few-shot learning experiments on ModelNet40 [8], we adopt n-way, m-shot setting, where n is the number of classes that randomly selected from the dataset and m is the number of objects randomly sampled for each class. We use the above-mentioned $n \times m$ objects for training. During testing, we randomly sample 20 unseen objects from each of n classes for evaluation.

Segmentation We evaluate the representation learning capability of our Point-MAE on ShapeNetPart dataset [9], which contains 16,881 objects covering 16 categories. We follow previous works [4, 5, 10] to sample 2048 points as input for each object, which results in 128 point patches.

As shown in Figure 2 right, our segmentation head is simple. We take the features from $4^{th}, 8^{th}$ and 12^{th} Transformer block, then concatenate them together. After doing max pooling and average pooling separately, we concatenate them together with the class feature, resulting in a global feature. The concatenated feature of 3 layers can be regarded as point features for the center points of point patches. Therefore, we up-sample [5] this sparse center points' features to the features of every input point based on point coordinates. Finally, we concatenate global features and per point features, and then feed them to an MLP to predict part label for each point, completing the part segmentation task.

Experiments settings The detailed experiment setting for each task is presented in Table 1.

Table 1. Experiment setting for each task.

Task	Classification	Classification	Few-shot	Segmentation
Dataset	ScanObjectNN[6]	ModelNet40[8]	ModelNet40[8]	ShapeNetPart[9]
Optimizer	AdamW	AdamW	AdamW	AdamW
Learning rate	0.0005	0.0005	0.0005	0.0002
Weight dewcay	0.05	0.05	0.05	0.05
Scheduler	Cosine	Cosine	Cosine	Cosine
Warmup epoch	10	10	10	10
Batch size	32	32	32	16
Epoch	300	300	150	300

3 Additional Study on Model Size

Table 2. Additional study on model size. We report the pre-train loss (10^{-3}) and fine-tune accuracy (%) on ModelNet40 for different size of models.

Model	Trans. dim	Enc. depth	Dec. depth	Loss	Acc.
A(ours)	384	12	4	2.60	**93.19**
B	768	12	4	2.59	92.50
C	384	24	8	2.55	92.75

We also conduct a simple study on model size. We use random masking at a ratio of 60% during pre-training. No voting method is used during testing in fine-tuning on ModelNet40. Other settings remain the same. As shown in Table 2, our base model has 12 Transformer blocks for the encoder and 4 Transformer blocks for the decoder, with a Transformer dimension of 384, achieving the best accuracy 93.19%. When we only increase the dimension of Transformer (Model B), the fine-tune accuracy drops to 92.50%. When we only double the depth of the base model (Model C), the fine-tune accuracy drops to 92.75%. Although a larger model, in terms of depth or width, can achieve better reconstruction, it causes overfitting during fine-tuning, which is also observed in training curves.

References

1. Chang, A.X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., et al.: Shapenet: An information-rich 3d model repository. arXiv preprint arXiv:1512.03012 (2015)
2. Loshchilov, I., Hutter, F.: Sgdr: Stochastic gradient descent with warm restarts. arXiv preprint arXiv:1608.03983 (2016)
3. Loshchilov, I., Hutter, F.: Decoupled weight decay regularization. arXiv preprint arXiv:1711.05101 (2017)
4. Qi, C.R., Su, H., Mo, K., Guibas, L.J.: Pointnet: Deep learning on point sets for 3d classification and segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 652–660 (2017)
5. Qi, C.R., Yi, L., Su, H., Guibas, L.J.: Pointnet++: Deep hierarchical feature learning on point sets in a metric space. Advances in neural information processing systems **30** (2017)
6. Uy, M.A., Pham, Q.H., Hua, B.S., Nguyen, T., Yeung, S.K.: Revisiting point cloud classification: A new benchmark dataset and classification model on real-world data. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 1588–1597 (2019)
7. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. Advances in neural information processing systems **30** (2017)
8. Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., Xiao, J.: 3d shapenets: A deep representation for volumetric shapes. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 1912–1920 (2015)
9. Yi, L., Kim, V.G., Ceylan, D., Shen, I.C., Yan, M., Su, H., Lu, C., Huang, Q., Sheffer, A., Guibas, L.: A scalable active framework for region annotation in 3d shape collections. ACM Transactions on Graphics (ToG) **35**(6), 1–12 (2016)
10. Yu, X., Tang, L., Rao, Y., Huang, T., Zhou, J., Lu, J.: Point-bert: Pre-training 3d point cloud transformers with masked point modeling. arXiv preprint arXiv:2111.14819 (2021)