Point2Vec for Self-Supervised Representation Learning on Point Clouds

Karim Abou Zeid*, Jonas Schult*, Alexander Hermans, Bastian Leibe
Computer Vision Group, Visual Computing Institute
RWTH Aachen University
karim.abou.zeid@rwth-aachen.de {schult,hermans,leibe}@vision.rwth-aachen.de

Abstract

Recently, self-supervised pre-training of Transformer architectures has made remarkable progress in 2D computer vision. In particular, data2vec has shown impressive results using a masked student–teacher approach. However, it remains open whether such a framework generalizes to the unique challenges posed by 3D point clouds. To this end, we extend data2vec to the point cloud domain and show promising results on several downstream tasks. However, our analysis reveals that disclosing positional information can expose the object's overall shape to the student, which hinders data2vec from learning strong representations. To address this 3D-specific shortcoming, we propose point2vec, which unleashes the full potential of data2vec-like pre-training on point clouds. Our experiments show that point2vec outperforms other self-supervised Transformer-based methods on shape classification on ModelNet40 and ScanObjectNN. Our results suggest that the learned representations are both transferable and strong, highlighting the potential of point2vec as a promising direction for self-supervised learning of point cloud representations.

1. Introduction

In this work, we address the task of self-supervised representation learning of Transformer-based models on 3D point clouds. Self-supervised training has shown impressive results in natural language processing [7, 26], speech [2, 13], and 2D vision [1, 3, 6, 10, 11], enabling learning of meaningful representations from massive unlabeled datasets without any human annotations. Only recently, we have seen self-supervised methods being successfully applied to Transformer architectures for 2D vision [1, 3, 11] and 3D point clouds [18, 28, 30]. Baevski *et al.* propose data2vec [1], a modality-agnostic self-supervised learning framework showing competitive performance in speech

*Equal contribution.

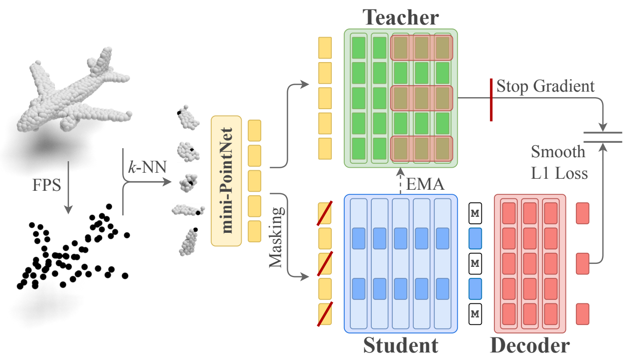
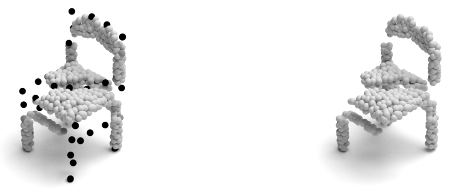


Figure 1. Illustration of our point2vec pre-training method.

recognition, image classification, and natural language understanding. Data2vec uses a joint-embedding architecture [1, 3, 10] with a *student* Transformer encoder and a *teacher* network parameterized as the exponential moving average of the student weights. Specifically, the teacher first predicts latent representations using an uncorrupted view of the input, which the student network then predicts from a masked view of the same input.

In this work, our aim is to apply data2vec-like pre-training to point clouds. The key difference to top-performing Transformer-based approaches for point cloud representation learning such as Point-MAE [18], Point-M2AE [30] and Point-BERT [28] is the target representation. The self-attention in the student Transformer encoder of data2vec generates *contextualized* feature targets that contain *global* information of the entire input. In contrast, Point-MAE [18] and Point-M2AE [30] explicitly reconstruct only *local* point cloud patches, and Point-BERT [28] is restricted to a fixed-sized vocabulary of token representations. To apply data2vec [1] on point clouds, we use the same underlying 3D-specific Transformer model as Point-BERT [28] and Point-MAE [18]. In experiments, we show that these modality-specific adaptations to data2vec already enable competitive performance compared to highly 3D-specific self-supervised approaches [15, 18, 24, 28, 30]. Encouraged by these promising results, we perform a subsequent analysis that reveals a crucial and point cloud specific



(a) Disclosed Positions (**data2vec–pc**) **(b)** Concealed Positions (**point2vec**)

Figure 2. Leakage Of Positional Information. The center points ● of masked point patches are associated with the masked embeddings (Fig. 1, $\boxed{\text{M}}$). **(a)** data2vec–pc discloses the positions of masked patches to the student, revealing the chair's overall shape. **(b)** point2vec excludes masked embeddings from the student and therefore conceals the positions of the masked patches.

shortcoming that restricts data2vec's representation learning capabilities: data2vec uses masked embeddings in the student network which carry positional information. Unlike images, text, and speech, the positional information in point clouds contains semantic meaning, namely 3D point locations (Fig. 2). Feeding masked embeddings with positional information into the student network therefore reveals the overall object shape to the student which makes the masking operation far less effective, as also reported by Pang *et al.* [18] in the context of masked autoencoders for point clouds. Based on this analysis, we propose point2vec that effectively addresses the leakage of positional information to the student and thus unleashes the full potential of data2vec-like pre-training for point clouds. To this end, we exclude masked embeddings from the student network. This prevents the overall object shape from being revealed, while also decreasing the computational cost. Instead, we introduce a shallow decoder which processes masked embeddings together with the student's outputs and which is trained to regress the representations of the teacher (Fig. 1). As a result, point2vec learns transferable features in a self-supervised manner, outperforming top-performing self-supervised approaches on several downstream tasks.

2. Method

The aim of this work is to unlock the full potential of data2vec-like [1] pre-training on point clouds by addressing point cloud specific challenges. To achieve this, we first summarize the technical concepts of data2vec (Sec. 2.1) and show how to learn rich representations on point clouds using data2vec pre-training (Sec. 2.2). Finally, we propose point2vec, which accounts for the point cloud specific limitations of data2vec (Sec. 2.3).

2.1. Data2vec

Data2vec [1] is designed to pre-train Transformer-based models, which involve a feature encoder that maps the input data to a sequence of embeddings. These embeddings are subsequently passed to a standard Transformer encoder to generate the final latent representations. During pre-training, two versions of the Transformer encoder are kept: a *student* and a *teacher*. The teacher is a momentum encoder, *i.e.* its parameters Δ track the student's parameters θ by being updated after each training step according to an exponential moving average (EMA) rule [1, 3, 10, 12]: $\Delta \leftarrow \tau\Delta + (1-\tau)\theta$, where $\tau \in [0,1]$ is the EMA decay rate. The teacher provides the training targets, which the student predicts given a corrupted version of the same input.

In a first step, the teacher encodes the uncorrupted input sequence. The training targets are then constructed by averaging the outputs of the last K blocks of the teacher, which are normalized beforehand to prevent a single block from dominating the sum. Due to the self-attention layers, these targets are *contextualized*, *i.e.* they incorporate global information from the whole input sequence. This is an important difference to other masked-prediction methods such as BERT [7] and MAE [11], where the targets only comprise local information, *e.g.* a word or an image patch.

The student is given a masked version of the same input, where some of the embeddings in the input sequence are substituted by a special learned *mask embedding*. The student's task is to predict the targets corresponding to the masked parts of the input. The model is trained by optimizing a Smooth L1 loss on the regressed targets.

2.2. Data2vec for Point Clouds

To apply data2vec to point clouds, we utilize the same underlying model as Point-BERT [28] and Point-MAE [18]. This model is well suited for data2vec pre-training: it extracts a sequence of patch embeddings from the input point cloud and feeds it to a standard Transformer encoder. For downstream tasks, we append a task-specific head to the Transformer encoder (Sec. 3). Next, we describe the point cloud embedding and the Transformer in detail and conclude with a summary of data2vec for point clouds.

Point Cloud Embedding. First, we sample n center points from the input point cloud using farthest point sampling (FPS) [20]. Grouping the center points' k-nearest neighbors (k-NN) in the point cloud yields n contiguous *point patches*, *i.e.* sub-clouds of k elements. Next, we normalize the point patches by subtracting the corresponding center point from the patch's points. This untangles the positional and the structural information. To account for the permutation-invariant property of point clouds, we employ a mini-PointNet [19], that maps each normalized point patch to a *patch embedding*. The mini-PointNet involves the following steps: First, we map each point of a patch to a feature vector using a shared MLP. Then, we concatenate max-pooled features to each feature vector. The resulting feature vectors are passed through a second shared MLP and a final max-pooling layer to obtain the patch embedding.

Transformer Encoder. The central component of the model is a standard Transformer encoder. The patch embeddings form the input sequence to the Transformer encoder. Since the point patches are normalized, the patch embeddings carry no positional information; therefore, a two-layer MLP maps each center point to a position embedding, which is then added to the corresponding patch embedding. Due to the special importance of positional information in point clouds, the position embeddings are added again before each subsequent Transformer block to ensure that the positional information is incorporated at every step of the encoding process.

Data2vec–pc. To establish a baseline, we apply the unmodified data2vec approach to the previously described underlying model of Point-BERT and Point-MAE. Going forward, we will refer to this approach as data2vec–pc.

2.3. Point2vec

In Fig. 1, we present the complete pipeline of our point2vec model. Directly applying data2vec to point cloud data without modifications is not optimal, as the position embeddings are also added to the mask embeddings, revealing the overall shape of the point cloud to the student. As positions are the only features for point clouds, this makes the masking far less effective, as noted by Pang *et al.* [18] in the context of masked autoencoders.

To solve this issue, we adopt an approach inspired by MAE [11], where we only feed the non-masked embeddings to the student 🟦. A separate decoder 🟥, implemented as a shallow Transformer encoder, takes the output of the student and the previously held-back masked embeddings $\boxed{\text{M}}$ as input and predicts the training targets. In contrast to data2vec–pc, this approach does not suffer from leaking positional information from the masked-out point patches to the student. Moreover, utilizing an MAE-inspired setup provides additional benefits: First, the student is more computationally efficient, as it only needs to process the non-masked embeddings. Second, the model's inputs during fine-tuning are more similar to those during pre-training because the inputs during pre-training are no longer dominated by masked embeddings which are absent during fine-tuning. This likely makes the learned representations more transferable to downstream tasks.

3. Experiments

In this section, we describe the self-supervised pre-training of point2vec on ShapeNet [4] (Sec. 3.1). Next, we compare point2vec with top-performing self-supervised approaches and our baseline method data2vec–pc for shape classification on two well-established datasets (Sec. 3.2). In the supp. mat., we further provide results on few-shot classification and part segmentation. Finally, we put the spotlight on the architectural changes from our data2vec adaptation for point clouds to our proposed model point2vec which address the unique challenges of 3D point clouds (Sec. 3.3).

3.1. Self-Supervised Pre-training

Following the pre-training protocol propagated by Point-BERT [28], Point-MAE [18] and Point-M2AE [30], we pre-train point2vec on the training split of ShapeNet [4] consisting of 41 952 synthetic 3D meshes of 55 categories, *e.g.* '*chair*', '*guitar*', '*airplane*'. We set the number of Transformer blocks to 12 with an internal dimension of 384. To pre-train our point-based approach, we uniformly sample 8192 points from the surfaces of the objects and then resample 1024 points using farthest point sampling [20]. During the point cloud embedding step, we sample n=64 center points and k=32 nearest neighbors. We train point2vec with a batch size of 512 for 800 epochs using the AdamW [17] optimizer and a cosine learning rate decay [16] with a maximum learning rate of 10^{-3} after 80 epochs of linear warm-up. For data2vec–pc, we increase the batch size and learning rate to 2048 and $2{\times}10^{-3}$, respectively, as this empirically led to better results. Following data2vec [1], we set β=2 for the Smooth L1 loss and average the last K=6 blocks of the teacher. We use minimal augmentations during pre-training: we randomly scale the input between $[0.8, 1.2]$ and rotate around the gravity axis.

3.2. Main Results on Downstream Tasks

In order to evaluate the effectiveness of point2vec's self-supervised learning capabilities, we test point2vec against top-performing self-supervised Transformer-based methods on well-established benchmarks. To that end, we discard the teacher network as well as the decoder and append a task-specific head to the student network. We then fine-tune the full network end-to-end for the specific task.

Synthetic Shape Classification. After pre-training on ShapeNet, we finetune our model for shape classification on ModelNet40 [25] consisting of 12 311 *synthetic* 3D models of 40 semantic categories. We obtain the semantic class label by passing the concatenated mean- and max-pooled output of the Transformer encoder into a 3-layer MLP and finetune the whole network end-to-end. During the point cloud embedding step, we sample n=64 center points and k=32 nearest neighbors. In Tab. 1, we report a new state-of-the-art for shape classification on ModelNet40 [25] among self-supervised methods by a large margin of +1.3% without voting [18, 28, 30]. Interestingly, pre-training with data2vec–pc results only in marginal improvements (+0.3% without voting) over the same model trained *from scratch* on ModelNet40. Unlike data2vec–pc, we observe that point2vec unleashes the full potential of data2vec-like pre-training on ModelNet40 by achieving substantial performance gains of +1.7% over the baseline trained from scratch.

Table 1. Shape Classification. We report the overall accuracy on ModelNet40 [25] and ScanObjectNN [23].

	Overall Accuracy		
	ModelNet40		**ScanObjNN**
	+Voting	−Voting	`PB-T50-RS`
Point-BERT [28]	93.2	92.7	83.1
MaskPoint [15]	93.8	–	84.6
Point-MAE [18]	93.8	93.2	85.2
Point-M2AE [30]	94.0	93.4	86.4
from scratch	93.3	93.0	84.3
data2vec–pc	93.6 ≀ +0.3	93.3 ≀ +0.3	85.5 ≀ +1.2
point2vec (Ours)	**94.8** ≀ +1.2	**94.7** ≀ +1.4	**87.5** ≀ +2.0

Table 2. Ablation. We find that a deferred shallow decoder (**D**) (Fig. 1 □) predicting the teacher's representations for masked patches shows consistent improvements but we identify that concealing positional information (**no** $\boxed{\text{M}}$) from the student is key.

			Overall Accuracy		
			ModelNet40		**ScanObjNN**
	no $\boxed{\text{M}}$	D	+Voting	−Voting	`PB-T50-RS`
data2vec–pc	✗	✗	93.6	93.3	85.5
	✗	✓	94.0	93.6	86.8
point2vec	✓	✓	**94.8**	**94.7**	**87.5**

Real-World Shape Classification. Next, we fine-tune point2vec on ScanObjectNN [23] containing 2902 *real-world* object scans of 15 semantic classes. In contrast to shape classification on ModelNet40, we use 2048 points and sample n=128 center points for the point cloud embedding step. Although pre-trained on synthetic data, Tab. 1 shows that point2vec generalizes well to cluttered real-world data and achieves state-of-the-art performance among self-supervised methods by a significant margin of +1.1% on `PB-T50-RS`, the most difficult variant of the dataset. Furthermore, we observe that pre-training point2vec on ShapeNet plays a crucial role to its strong performance. Compared to the baseline trained from scratch on ScanObjectNN, pre-training with point2vec achieves an impressive performance gain of +3.2%.

3.3. Leakage of Positional Information

The main limitation of data2vec–pc is that it directly feeds masked embeddings, along with their positional information, to the student network, which undermines the effectiveness of masking. To visualize this problem, we show a representative example in Fig. 2(a). Disclosing the positions of masked patches inadvertently weakens the learning objective because it provides the student with a coarse view of the entire object. To mitigate this issue, point2vec excludes masked embeddings from the student and only subsequently feeds them to the decoder. As a result, several sections of the chair in Fig. 2(b) are effectively concealed from the student network, leading to a more resilient learning framework. In Tab. 2, we report that point2vec outperforms our baseline data2vec–pc by a significant margin of up to +2.0%. In particular, we observe that the decoder itself provides consistent improvements, but the key contribution of point2vec is to conceal positional information from the student. Complementary to our findings, He *et al.* [11] show that moving masked embeddings to a deferred shallow decoder reduces memory requirements and training time significantly. Our findings align with those of Pang *et al.* [18], who found similar benefits for masked autoencoders on point clouds. While we seek a challenging pretext task to learn rich representations, ambiguity should not be the primary source of difficulty.

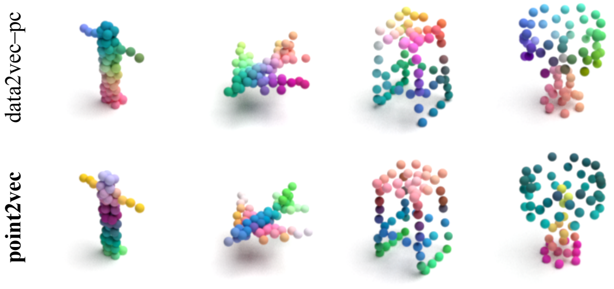


Figure 3. Visualization of Learned Representations. We use PCA to project the representations into RGB space. data2vec–pc pre-training shows a fairly strong positional bias, whereas point2vec exhibits a stronger semantic grouping without being trained on downstream dense prediction tasks.

Visualization of representations learned by point2vec. In Fig. 3, we show qualitative examples of representations of ModelNet40 instances after pre-training on ShapeNet. Data2vec–pc pre-training shows a strong positional bias, whereas point2vec exhibits a stronger semantic grouping without being trained on downstream dense prediction tasks. Unlike data2vec–pc, point2vec conceals positional information from the student, forcing it to learn more about the semantics of the data.

4. Conclusion

In this work, we have extended data2vec to the point cloud domain. Through an in-depth analysis, we have discovered that the disclosure of positional information to the student network hampers data2vec's ability to learn strong representations on point clouds. To overcome this limitation, we have introduced point2vec, a self-supervised representation learning approach which unleashes the full potential of data2vec-like pre-training on point clouds. Point2vec achieves remarkable results on various downstream tasks, surpassing other self-supervised learning approaches in few-shot learning as well as shape classification on well-established benchmarks. Future work might include extending point2vec for scene-level representation learning.

References

[1] Alexei Baevski, Wei-Ning Hsu, Qiantong Xu, Arun Babu, Jiatao Gu, and Michael Auli. Data2vec: A general framework for self-supervised learning in speech, vision and language. In *International Conference on Machine Learning*, 2022. 1, 2, 3, 7, 8

[2] Alexei Baevski, Yuhao Zhou, Abdelrahman Mohamed, and Michael Auli. wav2vec 2.0: A framework for self-supervised learning of speech representations. In *Neural Information Processing Systems*, 2020. 1

[3] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *International Conference on Computer Vision*, 2021. 1, 2

[4] 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. *arXiv preprint arXiv:1512.03012*, 2015. 3

[5] Chun-Fu (Richard) Chen, Quanfu Fan, and Rameswar Panda. CrossViT: Cross-Attention Multi-Scale Vision Transformer for Image Classification. In *International Conference on Computer Vision*, 2021. 7

[6] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A Simple Framework for Contrastive Learning of Visual Representations. In *International Conference on Machine Learning*, 2020. 1

[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2018. 1, 2

[8] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2020. 7

[9] Haoqi Fan, Bo Xiong, Karttikeya Mangalam, Yanghao Li, Zhicheng Yan, Jitendra Malik, and Christoph Feichtenhofer. Multiscale Vision Transformers. In *International Conference on Computer Vision*, 2021. 7

[10] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent-a new approach to self-supervised learning. In *Neural Information Processing Systems*, 2020. 1, 2

[11] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2022. 1, 2, 3, 4

[12] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2020. 2

[13] Wei-Ning Hsu, Yao-Hung Hubert Tsai, Benjamin Bolte, Ruslan Salakhutdinov, and Abdelrahman Mohamed. Hubert: How Much Can a Bad Teacher Benefit ASR Pre-Training? In *IEEE Conference on Acoustics, Speech and Signal Processing*, 2021. 1

[14] Yanghao Li, Chao-Yuan Wu, Haoqi Fan, Karttikeya Mangalam, Bo Xiong, Jitendra Malik, and Christoph Feichtenhofer. MViTv2: Improved multiscale vision transformers for classification and detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2022. 7

[15] Haotian Liu, Mu Cai, and Yong Jae Lee. Masked Discrimination for Self-Supervised Learning on Point Clouds. In *European Conference on Computer Vision*, 2022. 1, 4, 7, 8

[16] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. In *International Conference on Learning Representations*, 2017. 3

[17] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. 3

[18] Yatian Pang, Wenxiao Wang, Francis EH Tay, Wei Liu, Yonghong Tian, and Li Yuan. Masked autoencoders for point cloud self-supervised learning. In *European Conference on Computer Vision*, 2022. 1, 2, 3, 4, 7, 8, 9

[19] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep Learning on Point Sets for 3D Classification and Segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2017. 2

[20] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. In *Advances in Neural Information Processing Systems*, 2017. 2, 3, 7

[21] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, 2015. 7

[22] Charu Sharma and Manohar Kaul. Self-Supervised Few-Shot Learning on Point Clouds. In *Neural Information Processing Systems*, 2020. 7

[23] Mikaela Angelina Uy, Quang-Hieu Pham, Binh-Son Hua, Duc Thanh Nguyen, and Sai-Kit Yeung. Revisiting Point Cloud Classification: A New Benchmark Dataset and Classification Model on Real-World Data. In *International Conference on Computer Vision*, 2019. 4, 9, 11

[24] Hanchen Wang, Qi Liu, Xiangyu Yue, Joan Lasenby, and Matthew J. Kusner. Unsupervised Point Cloud Pre-Training via Occlusion Completion. In *International Conference on Computer Vision*, 2021. 1, 7

[25] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d ShapeNets: A Deep Representation for Volumetric Shapes. In *International Conference on Computer Vision*, 2015. 3, 4, 7, 9, 11

[26] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. XLNet: Generalized Autoregressive Pretraining for Language Understanding. In *Neural Information Processing Systems*, 2019. 1

[27] 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. 7, 8, 11

[28] Xumin Yu, Lulu Tang, Yongming Rao, Tiejun Huang, Jie Zhou, and Jiwen Lu. Point-bert: Pre-training 3d point cloud transformers with masked point modeling. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2022. 1, 2, 3, 4, 7, 8

[29] Pengchuan Zhang, Xiyang Dai, Jianwei Yang, Bin Xiao, Lu Yuan, Lei Zhang, and Jianfeng Gao. Multi-Scale Vision Longformer: A New Vision Transformer for High-Resolution Image Encoding. In *International Conference on Computer Vision*, 2021. 7

[30] Renrui Zhang, Ziyu Guo, Peng Gao, Rongyao Fang, Bin Zhao, Dong Wang, Yu Qiao, and Hongsheng Li. Point-M2AE: Multi-scale masked autoencoders for hierarchical point cloud pre-training. In *Advances in Neural Information Processing Systems*, 2022. 1, 3, 4, 7, 8

Point2Vec for Self-Supervised Representation Learning on Point Clouds
Supplementary Material

Abstract

This supplementary material contains further results and ablation studies on the efficiency of pre-training data and the selection of hyperparameters during pre-training and fine-tuning on downstream tasks. Our code and model will be made publicly available for research purposes. We present point2vec in a supplementary video.

A. Further Results

Few-Shot Classification. Following the standard evaluation protocol proposed by Sharma *et al.* [22], we test the few-shot capabilities of point2vec in a m-way, n-shot setting. To this end, we randomly sample m classes and select n instances for training at random for each of these classes. For testing, we randomly pick 20 unseen instances from each of the m support classes. We provide the standard deviation over 10 independent runs. In Tab. 3, we report a new state-of-the-art by improvements up to +1.3% in the most difficult 10-way 10-shot setting. Point2vec clearly outperforms the data2vec–pc baseline in all settings. We conclude that point2vec learns rich feature representations which are also well suited for transfer learning in a low-data regime.

Part Segmentation. We also address the task of part segmentation, which assigns a semantic part label to each point in a 3D point cloud of a single object. For this purpose, we employ a simple segmentation head that is similar to the segmentation head in Point-MAE [18]. First, we average the outputs of the 4th, 8th, and 12th Transformer blocks to incorporate features from multiple levels of abstraction. We then concatenate the mean- and max-pooling of the n averaged token outputs, along with the one-hot encoded class label of the object, to obtain a global feature vector. At the same time, we up-sample the n averaged outputs from the corresponding center points to all points using a PointNet++ [20] *feature propagation layer*, which uses inverse distance weighting and a shared MLP to produce a local feature vector for each point. Finally, we concatenate the global feature vector with each local feature vector and a shared MLP predicts a part label for each point. In Tab. 4, we report competitive results on ShapeNetPart [27] which consists of 16 881 3D models of 16 semantic categories. Apart from Point-M2AE [30], point2vec outperforms all other self-supervised methods. We hypothesize that Point-M2AE's multi-scale U-Net like architecture [21] enables to learn more expressive spatially localized features which results in slightly better scores (+0.2 mIoU_I). Since point2vec relies on a standard single-scale Transformer backbone, we see multi-scale Transformers for 3D point clouds as an interesting orthogonal improvement, similar to the advances in 2D vision [5, 9, 14, 29] extending vision Transformers [8] with multi-scale capabilities.

Table 3. Few-Shot Classification on ModelNet40 [25]. We report mean and standard deviation over 10 runs.

	5-way		10-way	
Method	10-shot	20-shot	10-shot	20-shot
OcCo [24]	91.9±3.6	93.9±3.1	86.4±5.4	91.3±4.6
Transf.-OcCo [28]	94.0±3.6	95.9±2.3	89.4±5.1	92.4±4.6
Point-BERT [28]	94.6±3.1	96.3±2.7	91.0±5.4	92.7±5.1
MaskPoint [15]	95.0±3.7	97.2±1.7	91.4±4.0	93.4±3.5
Point-MAE [18]	96.3±2.5	97.8±1.8	92.6±4.1	95.0±3.0
Point-M2AE [30]	96.8±1.8	98.3±1.4	92.3±4.5	95.0±3.0
from scratch	93.8±3.2	97.1±1.9	90.1±4.6	93.6±3.9
data2vec–pc	96.2±2.6	97.8±2.2	92.6±4.9	95.0±3.2
point2vec (Ours)	**97.0**±2.8	**98.7**±1.2	**93.9**±4.1	**95.8**±3.1

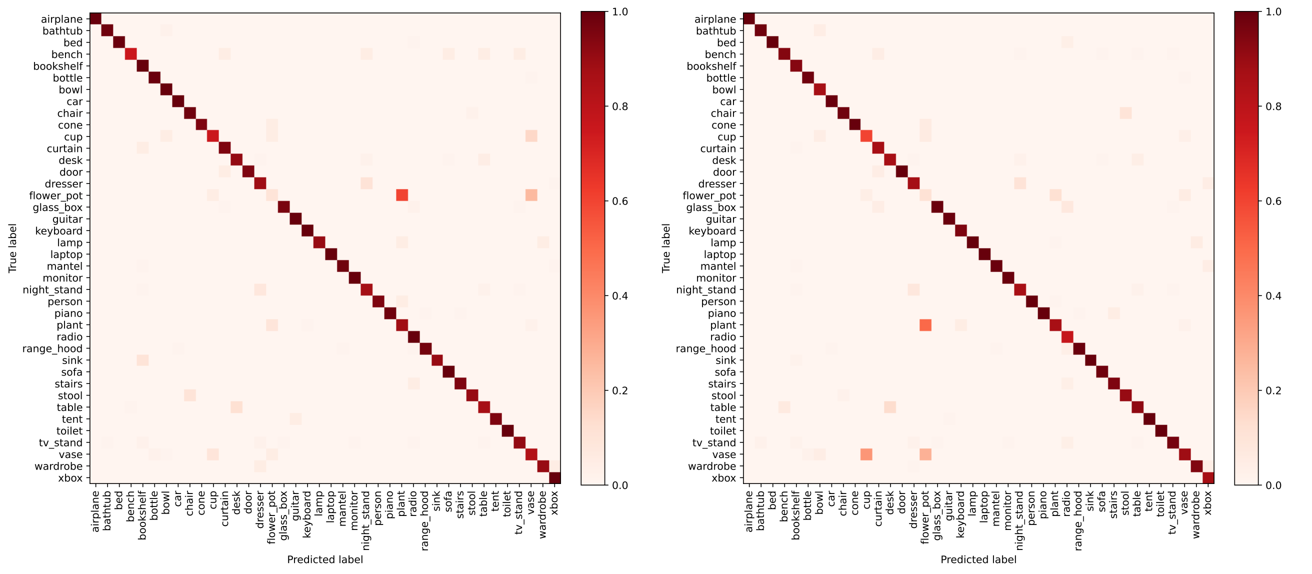
B. Further Ablation Studies

Pretext Task. In the main paper, we have only explored self-supervised pre-training on the ShapeNet dataset. However, ShapeNet also contains class labels which could instead be used for a fully supervised classification-based pre-training. As can be seen in Tab. 5, this yields a significantly worse performance than using point2vec or even than directly training *from scratch*. We can also pre-train using point2vec directly on ModelNet40, which constitutes roughly a quarter of ShapeNet's size. Still, we see improved downstream performances, indicating that the point2vec pretext task is meaningful for pre-training.

Warm-Up EMA Rate. During pre-training, we linearly warm-up the EMA rate τ over the first τ_n epochs from its initial value τ_0 to its final value τ_e [1]. This approach is based on the idea that we should update the teacher network more frequently at the start of training since the feature representations are not yet well-established. In Tab. 6, we show overall accuracy scores on ModelNet40 and the `PB-T50-RS` variant of ScanObjectNN using various values for τ_n. Our findings suggest that τ_n is a crucial hyperparameter for achieving effective pre-training with point2vec. In Tab. 8, we provide the EMA rates employed by our baseline data2vec–pc and point2vec, respectively.

Table 4. Part Segmentation Results on ShapeNetPart [27]. We report the mean IoU across all part categories mIoU$_C$ and the mean IoU across all instance mIoU$_I$, as well as the IoU for each categories.

Methods	mIoU$_C$	mIoU$_I$	aero	bag	cap	car	chair	earph	guitar	knife	lamp	laptop	motor	mug	pistol	rocket	skateb	table
Transf.-OcCo [28]	83.4	85.1	83.3	85.2	88.3	79.9	90.7	74.1	91.9	87.6	84.7	95.4	75.5	94.4	84.1	63.1	75.7	80.8
Point-BERT [28]	84.1	85.6	84.3	84.8	88.0	79.8	91.0	81.7	91.6	87.9	85.2	95.6	75.6	94.7	84.3	63.4	76.3	81.5
MaskPoint [15]	84.4	86.0	84.2	85.6	88.1	80.3	91.2	79.5	91.9	87.8	86.2	95.3	76.9	95.0	85.3	64.4	76.9	81.8
Point-MAE [18]	84.1	86.1	84.3	85.0	88.3	80.5	91.3	78.5	92.1	87.4	86.1	96.1	75.2	94.6	84.7	63.5	77.1	82.4
Point-M2AE [30]	**84.9**	**86.5**	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
from scratch	84.1	85.7	84.2	84.8	87.9	80.2	91.2	79.3	91.6	87.8	85.4	96.1	75.8	94.7	85.0	64.7	75.4	81.1
data2vec–pc	84.1	85.9	84.7	85.8	87.4	80.7	91.4	79.3	91.7	87.9	85.2	95.8	76.2	94.9	83.6	61.6	78.5	80.7
point2vec (Ours)	84.6	86.3	85.2	85.7	88.5	80.3	91.5	77.3	91.8	88.5	85.7	96.1	77.3	95.5	84.9	66.2	77.0	82.0



(a) Row-normalized confusion matrix

(b) Column-normalized confusion matrix

Figure 4. Confusion matrix of point2vec on the ModelNet40 test split. We present the confusion matrix, both row-normalized (a) and column-normalized (b). The diagonals of these show the recall and precision respectively. As expected, the matrix reveals that the majority of misclassifications occur between a small number of closely related classes. The most frequent cases of misclassifications are `night_stands` that are classified as `dressers`, `flower_pots` that are classified as `plants` and `tables` that are classified as `desks`.

Target Layer Aggregation. During training of data2vec–pc, as well as point2vec, we need to specify which layers of the teacher should be defined as the target. The target is constructed by averaging the last K layers, where Baevski *et al.* [1] recommend to use half the number of total layers in absence of additional experiments. We ablate this hyperparameter and report results in Tab. 7. Although all tested values lead to usable results, indeed $K = 6$ overall leads to the best performance.

Pre-Training Data Efficiency. We evaluate the efficiency of self-supervised pre-training with point2vec. To this end, we partition the ShapeNet training dataset into subsets containing 25%, 50%, 75% and 100% of the data. We then fine-tuned our model for shape classification on ModelNet40 and ScanObjectNN, respectively. As shown in Fig. 5, point2vec achieves consistent improvements on both datasets. Notably, pre-training point2vec with only 25% of the training data yields superior results compared to Point-MAE pre-trained with 100% of the training data.

Table 5. Pretext Tasks. After pre-training with a classification objective on ShapeNet, fine-tuning on ModelNet40 leads to no performance gains over directly training *from scratch* and significantly worse performance on the most difficult test split of ScanObjectNN. However, point2vec already brings performance gains when pre-trained with the much smaller ModelNet40 dataset and significant improvements when pre-trained with the large ShapeNet dataset.

	Overall Accuracy		
	ModelNet40		**ScanObjNN**
Pretext Task	+Voting	−Voting	`PB-T50-RS`
none / from scratch	93.3	93.0	84.3
classification (ShapeNet)	93.2	93.0	82.9
point2vec (ModelNet40)	93.9	93.6	84.4
point2vec (ShapeNet)	**94.8**	**94.7**	**87.5**

Table 6. Warm-Up EMA Rate. We linearly warm-up the EMA rate during the first τ_n epochs.

	Overall Accuracy		
	ModelNet40		**ScanObjNN**
τ_n	+Voting	−Voting	`PB-T50-RS`
80	94.5	94.1	86.7
160	94.6	94.2	87.4
200	**94.8**	**94.7**	**87.5**
300	94.1	94.0	87.3
400	94.0	94.0	87.3

Table 7. Target Layer Aggregation. We construct training targets by averaging the outputs of the last K Transformer blocks of the teacher. We observe that $K = 6$ is optimal for ModelNet40 and close to optimal for the `PB-T50-RS` variant of ScanObjectNN.

	Overall Accuracy		
	ModelNet40		**ScanObjNN**
K	+Voting	−Voting	`PB-T50-RS`
1	94.4	94.1	87.0
3	94.7	94.3	87.1
6	**94.8**	**94.7**	87.5
9	94.3	94.0	**87.6**
12	94.5	94.3	87.3

Class Confusions on ModelNet40. Given the very high overall accuracies on the ModelNet40 dataset, we further analyze the remaining errors. Fig. 4 shows the confusion matrix on the ModelNet40 test split, clearly showing that most remaining mistakes are made for a few classes with very similar appearances, which might also be difficult for humans to distinguish.

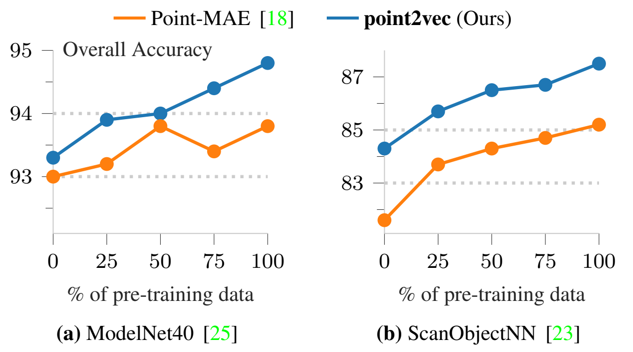


Figure 5. Pre-training Data Efficiency. Irrespective of the quantity of pre-training data used from ShapeNet, point2vec consistently achieves better results than Point-MAE [18] on ModelNet40 (with voting) and the most difficult test split of ScanObjNN.

C. Architecture Details

In Tab. 8, we provide detailed hyperparameters for pre-training data2vec–pc and point2vec on ShapeNet. We, furthermore, report the hyperparameters for fine-tuning point2vec for the shape classification (Tab. 9) and part segmentation task (Tab. 10). In Listing 1, we provide the PyTorch-inspired pseudocode for point2vec pre-training.

D. Qualitative Results for Part Segmentation

In Fig. 6, we show qualitative results for part segmentation on the ShapeNetPart dataset. Point2vec achieves remarkable results, as the boundaries between parts are accurately localized with minimal semantic errors. In the majority of instances, there is no perceivable difference between the results produced by point2vec and the ground truth.

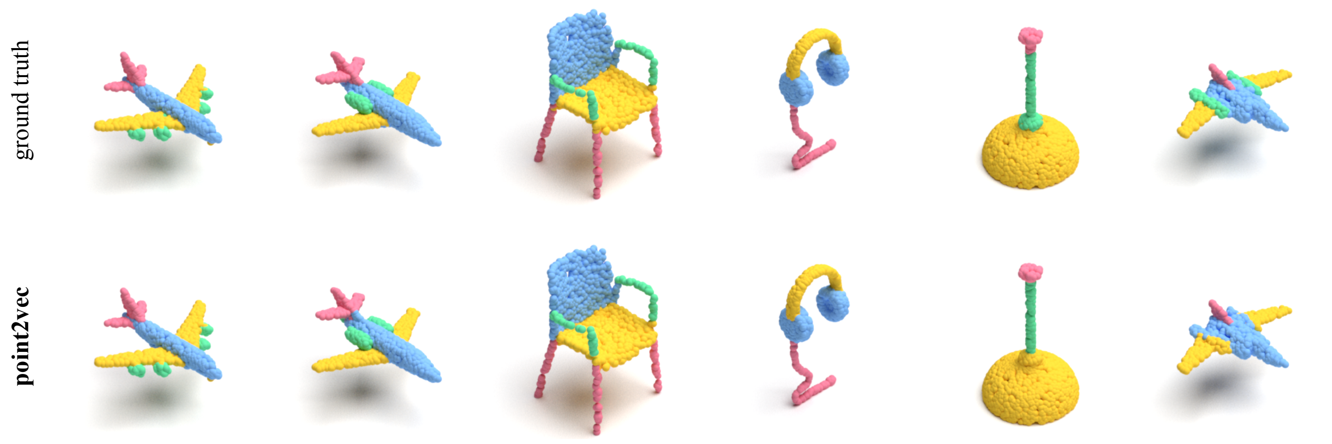


Figure 6. Qualitative Results on ShapeNetPart. point2vec produces well localized boundaries between parts with minimal semantic errors. In most cases, the differences between the results of point2vec and the ground truth are imperceptible to the human eye. However, the last example shows a failure case where the jet engine is not properly segmented.

```
# N: batch size (512)
# S: number of groups/embeddings (64)
# E: embedding feature dimension (384)

for point_cloud in data_loader:
    # point cloud embedding
    center_points = self.FPS(point_cloud)  # (N, S, 3)
    point_patches = self.KNN(point_cloud, center_points)  # (N, S, 32, 3)
    patch_embeddings = self.mini_pointnet(point_patches)  # (N, S, E) (Fig. 1, 🟨)
    pos_embeddings = self.pos_encoder(center_points)  # (N, S, E)

    # masking
    mask_embeddings = self.mask_embedding.expand(N, S, E)  # (N, S, E)
    mask = generate_mask(center_points)  # (N, S) boolean

    # targets
    with torch.no_grad():
        teacher_states = self.teacher(patch_embeddings, pos_embeddings)  # (12, N, S, E) (Fig. 1, 🟩)
        target_layers = [F.layer_norm(x, (E,)) for x in teacher_states[6:]]  # [(N, S, E)]
        targets = torch.stack(target_layers).mean(0)  # (N, S, E)
        targets = F.layer_norm(targets, (E,))  # (N, S, E)

    # predictions
    last_student_state = self.student(  # (N, S, E) (Fig. 1, 🟦)
        patch_embeddings[~mask].reshape(N, -1, E),
        pos_embeddings[~mask].reshape(N, -1, E)
    )[-1]
    predictions = self.decoder(  # (N, S, E) (Fig. 1, 🟥)
        mask_embeddings.index_put([~mask], last_student_state.reshape(-1, E)),
        pos_embeddings
    )[-1]

    # optimization
    loss = F.smooth_l1_loss(predictions[mask], targets[mask])
    loss.backward()
    optimizer.step()

    # update teacher weights
    ema_update(student, teacher)
```

Listing 1. PyTorch-inspired pseudocode for point2vec pre-training.

Table 8. Hyperparameters for data2vec–pc and point2vec. Data2vec–pc denotes our adaptation of data2vec to the point cloud modality. We report the best performing hyperparameters for both data2vec–pc and point2vec. LN: layer normalization. AVG: average pooling over layers.

	data2vec–pc	**point2vec**
Steps	800 epochs	800 epochs
Optimizer	AdamW	AdamW
Learning rate	2×10^{-3}	1×10^{-3}
Weight decay	0.05	0.05
LR Schedule	cosine	cosine
LR Warm-Up	80 epochs	80 epochs
Batch size	2048	512
Encoder layers	12	12
Encoder dimension	384	384
Decoder layers	–	4
Masking strategy	random	random
Masking ratio	65%	65%
τ_0 (EMA start)	0.9998	0.9998
τ_e (EMA end)	0.99999	0.99999
τ_n (EMA warm-up)	200 epochs	200 epochs
K (layers to average)	6	6
Target normalization	LN→AVG→LN	LN→AVG→LN

Table 9. Hyperparameters for Classification. We use the same hyperparameters when fine-tuning point2vec and data2vec–pc on ModelNet40 [25] and ScanObjectNN [23]. When training from scratch, we increase the learning rate to 1×10^{-3} and do not freeze the Transformer encoder.

Epochs	150
Batch size	32
Optimizer	AdamW
Learning rate	3×10^{-4}
Weight decay	0.05
Learning rate schedule	cosine
Learning rate warm-up	10 epochs
points	1024 (2048 for ScanObjNN)
n (center points)	64 (128 for ScanObjNN)
k (k-NN grouping)	32
mini-PointNet 1st MLP dim	128, 256
mini-PointNet 2nd MLP dim	512, 384
Encoder layers	12
Encoder dimension	384
Encoder heads	6
Encoder drop path	$0\%, \ldots, 20\%$
Encoder frozen	100 epochs
Feature aggregation	mean- & max-pooling
Classification head dim	256, 256, #classes
Classification head dropout	50%
Label smoothing	0.2

Table 10. Hyperparameters for Part Segmentation. We use the same hyperparameters when fine-tuning point2vec and data2vec–pc on ShapeNetPart [27].

Epochs	300
Batch size	16
Optimizer	AdamW
Learning rate	2×10^{-4}
Weight decay	0.05
Learning rate schedule	cosine
Learning rate warm-up	10 epochs
points	2048
n (center points)	128
k (k-NN grouping)	32
mini-PointNet 1st MLP dim	128, 256
mini-PointNet 2nd MLP dim	512, 384
Encoder layers	12
Encoder dimension	384
Encoder heads	6
Encoder drop path	$0\%, \ldots, 20\%$
Feature propagation	described in main paper
Segmentation head dim	512, 256, #classes
Segmentation head dropout	50%