point2vec for Self-Supervised Representation Learning on Point Clouds

Karim Knaebel* Jonas Schult* Alexander Hermans Bastian Leibe

RWTH Aachen University

https://vision.rwth-aachen.de/point2vec

Abstract

We present point2vec, a self-supervised method for learning point cloud representations through masked prediction of a teacher network's latent features. Building on data2vec, we investigate a limitation of applying this approach to point clouds: positional information associated with masked patches reveals the overall object shape to the student, weakening the learning task even under heavy masking. point2vec addresses this issue by passing only visible patches to the student encoder and introducing a shallow decoder to predict representations for masked patches. The teacher processes the complete point cloud to provide contextualized training targets. These targets capture global context through the teacher's self-attention, providing supervision beyond the local geometry of individual patches. This design conceals masked-patch positions from the student and reduces its computational cost during pre-training. Our ablations show that the decoder alone provides only part of the improvement; concealing positional information is key. After pre-training on ShapeNet, point2vec outperforms the evaluated self-supervised baselines on shape classification on ModelNet40 and ScanObjectNN and few-shot classification on ModelNet40, while achieving competitive part segmentation results on ShapeNetPart.

1. Introduction

In this work, we address the task of self-supervised representation learning on 3D point clouds. With the ever-increasing availability of affordable consumer-grade 3D sensors, point clouds are becoming a widely adopted data representation for capturing real-world objects and environments [1, 6–8, 15]. They provide accurate 3D geometric information, making them a valuable input for applications in robotics, autonomous driving [7, 8], and AR/VR. The 3D computer vision community has made impressive progress by developing 3D-centric approaches which directly process 3D point clouds to semantically understand 3D objects and environments [14, 32, 33, 36]. However, these approaches typically rely on fully supervised training from scratch [45], requiring time-consuming and labor-intensive human annotations. For example, semantically annotating a single room-scale scene of the ScanNet dataset takes about 22 min on average [15]. This results in a lack of large-scale annotated point cloud datasets, making it challenging to learn strong representations from limited data.

At the same time, self-supervised training has shown impressive results in natural language processing [16, 47], speech [2, 25], and 2D vision [3, 9, 12, 21, 23], enabling the learning of meaningful representations from massive unlabeled datasets without any human annotations.

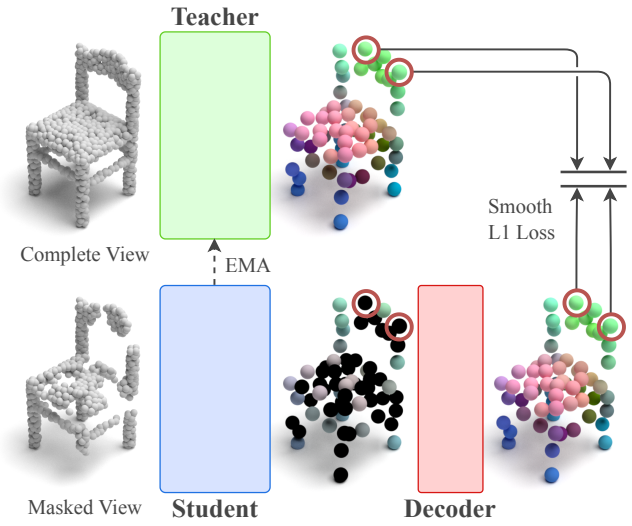


Figure 1. **Overview of point2vec.** During training, a teacher network ▢ predicts latent representations using a complete view of the point cloud. The student network ▢ encodes a partial view of the same point cloud. A shallow decoder ▢ predicts the teacher's representations for masked regions ●, providing the training signal for the student and decoder. The teacher weights are updated using an exponential moving average of the student weights.

*Equal contribution.

Only recently, we have seen self-supervised methods being successfully applied to Transformer architectures for 2D vision [3, 9, 23] and 3D point clouds [31, 49, 51]. Baevski et al. [3] propose data2vec, a modality-agnostic self-supervised learning framework showing competitive performance in speech recognition, image classification, and natural language understanding. data2vec uses a joint-embedding architecture [3, 9, 21] 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 paper, our aim is to apply data2vec-like pre-training to point clouds. The key difference to top-performing approaches for point cloud representation learning such as Point-MAE [31], Point-M2AE [51], and Point-BERT [49] is the target representation. The self-attention in the teacher Transformer encoder of data2vec generates contextualized feature targets that contain *global* information about the entire input. In contrast, Point-MAE and Point-M2AE explicitly reconstruct only *local* point cloud patches, and Point-BERT is restricted to a fixed-size vocabulary of token representations. To apply data2vec to point clouds, we use the same underlying 3D-specific Transformer model as Point-BERT and Point-MAE. In experiments, we show that these modality-specific adaptations to data2vec already enable competitive performance compared to highly 3D-specific self-supervised approaches [28, 31, 43, 49, 51]. Encouraged by these promising results, we perform a subsequent analysis that reveals a crucial shortcoming specific to point clouds 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. 4). 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. [31] in the context of masked autoencoders for point clouds. Based on this analysis, we propose point2vec, which 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).

Evaluating the quality of the learned representations on downstream tasks is a crucial step for analyzing self-supervised methods. Our experiments demonstrate that, after pre-training on the ShapeNet dataset [10], point2vec outperforms other self-supervised methods on both the ModelNet40 [44] and ScanObjectNN [40] shape classification benchmarks. Additionally, point2vec achieves state-of-the-art performance on few-shot classification on ModelNet40 and competitive results on part segmentation on ShapeNetPart [48]. These findings suggest that the learned representations are strong and transferable, indicating that point2vec is a promising approach for self-supervised point cloud representation learning.

To summarize, our contributions are:

1. We extend the seminal work data2vec to the point cloud domain.
2. In our experiments, we discover a crucial shortcoming of data2vec that limits its representation learning capabilities for point clouds: masked embeddings leak positional information to the student, revealing the overall object shapes even under heavy masking.
3. We propose point2vec, which unleashes the full potential of data2vec-like pre-training for self-supervised representation learning by addressing the aforementioned shortcoming. point2vec learns strong and transferable features in a self-supervised manner, outperforming self-supervised approaches on several downstream tasks.

2. Related Work

Self-supervised learning. Recently, self-supervised learning has gained much attention due to its promise to learn meaningful data representations without any human annotations. At the heart of self-supervised learning is the pretext task, offering a vast range of diverse options. One such line of work investigates contrastive learning objectives [12, 22, 38, 39, 41], *i.e.* maximizing feature similarity across multiple views of the same training sample while simultaneously minimizing the similarity to other training samples. Contrastive learning approaches typically rely on a careful choice of data augmentations, negative sample mining, or large batch sizes [21]. Addressing these limitations, non-contrastive approaches [3, 5, 9, 13, 21] follow a joint-embedding architecture, *i.e.* two copies of the same network are trained to produce similar latent representations for two views of the identical input. Among them and most important to our work is data2vec [3], which relies on a teacher first generating targets by predicting latent representations using the complete view of the input and a student which predicts these targets using only a masked view of the same input. Inspired by data2vec's flexibility across a wide range of modalities, in this paper, we seek to unlock the full potential of data2vec-like pre-training for point clouds by specifically taking the unique characteristics of point clouds into account.

Self-supervised learning on point clouds. The success of self-supervised learning in 2D vision [3–5, 9, 12, 21, 23, 41], natural language processing [3, 16], and speech [2, 3] has inspired a number of recent works proposing self-supervised learning frameworks for point cloud understanding tasks. Among them, contrastive self-supervised frameworks are typically deployed for room-scale pre-training. The pioneering work of Xie et al. [45] contrasts corresponding 3D points from multiple partial views of a reconstructed static scene, showing impressive improvements when fine-tuned on several scene-level downstream tasks. Building on this, Hou et al. [24] propose to leverage both point-level correspondences and spatial contexts of 3D scenes. In contrast to room-scale pre-training, we see a line of work developing self-supervised methods tailored towards single-object understanding tasks [19, 26, 28, 31, 34, 37, 43, 46, 49, 51]. They typically use the inherent structure and geometry of 3D point clouds to learn meaningful representations, *e.g.* by explicitly reconstructing point cloud patches using the Chamfer distance [31, 51], discriminating masked points from noise [28], or performing point cloud completion for occluded regions [43]. Another line of work additionally leverages multi-modal information to improve the latent representation of 3D point clouds, *i.e.* incorporating knowledge from models on 2D images [17, 46, 52, 53] or text descriptions [46, 52]. The advances of the above methods are orthogonal to our approach point2vec, as it operates on point clouds only. Most relevant to our work are Transformer-based self-supervised learning approaches on point clouds. Due to the success of pre-trained Transformer architectures in various domains [3, 9, 16, 23], we have recently seen a shift towards pre-training Transformer-based approaches for point clouds [28, 49, 51]. Among them, Point-BERT [49] introduces a standard ViT-like [18] backbone to point clouds and extends BERT [16] pre-training to point clouds. Point-MAE [31] and Point-M2AE [51] follow the masked autoencoder (MAE) approach proposed by He et al. [23]. In contrast to these methods, we do not explicitly reconstruct masked point cloud patches but predict contextualized targets in the latent feature space, circumventing the need to define sophisticated distance metrics to compare point cloud patches.

3. Method

The aim of this work is to unlock the full potential of data2vec-like [3] pre-training on point clouds by addressing challenges specific to point clouds. To achieve this, we first summarize the technical concepts of data2vec (Sec. 3.1) and show how to learn rich representations on point clouds using data2vec pre-training (Sec. 3.2). Finally, we propose point2vec, which accounts for the limitations of data2vec specific to point clouds (Sec. 3.3).

3.1. data2vec

data2vec 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 [42] 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 [3, 9, 21, 22]:

$$\Delta \leftarrow \tau\Delta + (1-\tau)\theta,$$

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 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 from other masked-prediction methods such as BERT [16] and MAE [23], 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 replaced 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.

3.2. data2vec for Point Clouds

To apply data2vec to point clouds, we utilize the same underlying model as Point-BERT [49] and Point-MAE [31]. 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. 4). 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) [33]. 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 ensure invariance to the

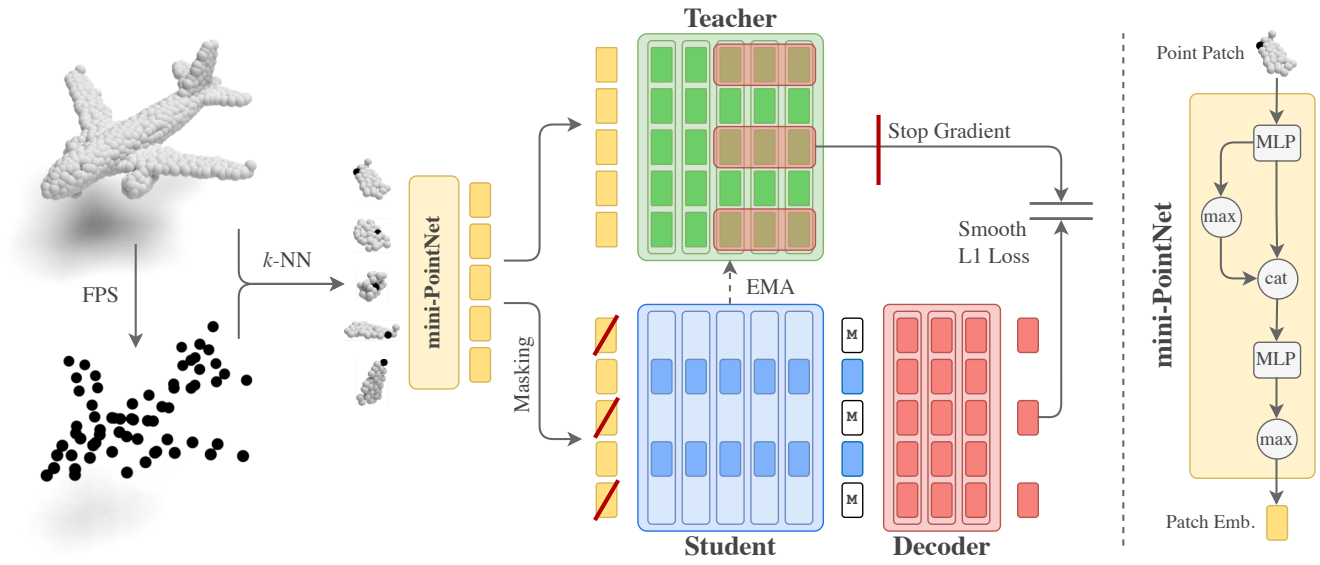


Figure 2. **point2vec pre-training.** We divide the point cloud into patches using farthest point sampling (FPS) and k-NN aggregation. We obtain patch embeddings by applying a mini-PointNet ▢ to each point patch (right). The teacher Transformer encoder ▢ infers contextualized representations for all patch embeddings, which, after normalization and averaging over the last K Transformer layers, serve as training targets. The student's input is a masked view of the input data, *i.e.* we randomly mask out a fraction of patch embeddings and only pass the remaining embeddings into the student Transformer encoder ▢. After applying a shallow decoder ▢ to the outputs of the student, padded with learned mask embeddings Ⓜ, we train the student and decoder to predict the teacher's latent patch representations.

order of points, we use a mini-PointNet [32] (Fig. 2, right) that maps each normalized point patch to a *patch embedding*.

The mini-PointNet involves the following steps:

1. Map each point of a patch to a feature vector using a shared MLP.
2. Concatenate max-pooled features to each feature vector.
3. Pass the resulting feature vectors 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. Our baseline, data2vec-PC, applies unmodified data2vec to the Point-BERT and Point-MAE backbone described above.

3.3. point2vec

In Fig. 2, 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, this makes the masking far less effective, as noted by Pang et al. [31] in the context of masked autoencoders (MAE).

To solve this issue, we adopt an approach inspired by MAE, 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 Ⓜ 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 they are no longer dominated by masked embeddings, which are absent during fine-tuning. This likely makes the learned representations more transferable to downstream tasks.

4. Experiments

In this section, we describe the self-supervised pre-training of point2vec on ShapeNet [10] (Sec. 4.1). Next, we compare point2vec with top-performing self-supervised approaches and our baseline method data2vec-PC on three well-established datasets and four downstream tasks (Sec. 4.2). 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. 4.3). In the supplementary material, we provide detailed hyperparameters of our model. Our code and checkpoints are publicly available.

4.1. Self-Supervised Pre-Training

Following the pre-training protocol used by Point-BERT [49], Point-MAE [31], and Point-M2AE [51], we pre-train point2vec on the training split of ShapeNet 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 [33]. 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 [30] optimizer and a cosine learning rate decay [29] 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 2×10^{-3}, respectively, as this empirically led to better results. Following data2vec [3], we set $\beta = 2$ for the Smooth L1 loss and average the last $K = 6$ blocks of the teacher. We use minimal data augmentations during pre-training: we randomly scale the input by a factor in $[0.8, 1.2]$ and rotate it around the gravity axis. Pre-training takes roughly 18 h on a single V100 GPU.

4.2. 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 methods on four different downstream tasks 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. We provide detailed hyperparameters for all downstream tasks in the supplementary material.

Synthetic shape classification. After pre-training on ShapeNet, we fine-tune our model for shape classification on ModelNet40 [44], which consists 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 fine-tune the whole network end-to-end. We use minimal data augmentations consisting only of resampling 1024 points with farthest point sampling, applying random anisotropic scaling of up to 40%, centering at the origin, and rescaling to the unit sphere. Other commonly used augmentations did not improve performance; *e.g.* random rotations around the axis of gravity and random translations are detrimental as ModelNet40 instances are canonically oriented.

Table 1. **Shape classification on ModelNet40 [44].** We report the overall accuracy with and without voting, as well as the mean per-class accuracy (mAcc).

Method	Overall accuracy		mAcc
	Voting	No voting	
Transf.-OcCo [49]	92.1	—	—
ParAE [19]	—	92.9	—
STRL [26]	93.1	—	—
Point-BERT [49]	93.2	92.7	—
PointGLR [34]	—	93.0	—
OcCo [43]	93.0	—	—
MaskPoint [28]	93.8	—	—
Point-MAE [31]	93.8	93.2	—
Point-M2AE [51]	94.0	93.4	—
from scratch	93.3	93.0	89.8
data2vec-PC	93.6	93.3	90.5
point2vec (ours)	**94.8**	**94.7**	**92.0**

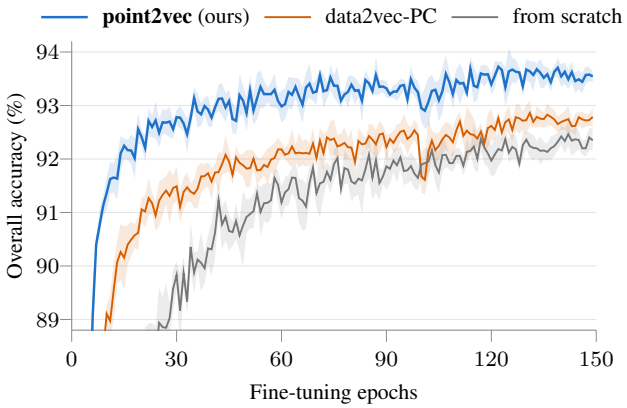


Figure 3. **Learning curves for ModelNet40 [44].** We show the mean (solid line) and the standard deviation (shaded background) over 6 independent runs of point2vec, data2vec-PC as well as the model trained from scratch on ModelNet40. point2vec consistently outperforms the baselines by a large margin.

Table 2. **Part segmentation results on ShapeNetPart [48].** We report category-averaged mIoU$_C$ and instance-averaged mIoU$_I$, as well as the mean IoU for each object category.

Method	mIoU$_C$	mIoU$_I$	Aero	Bag	Cap	Car	Chair	Earph	Guitar	Knife	Lamp	Laptop	Motor	Mug	Pistol	Rocket	Skateb	Table
Transf.-OcCo [49]	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 [49]	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 [28]	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 [31]	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 [51]	**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

Table 3. **Shape classification on ScanObjectNN [40].** We report the overall accuracy over the three subsets OBJ-BG, OBJ-ONLY, and the most challenging variant PB-T50-RS.

Method	Overall accuracy		
	OBJ-BG	OBJ-ONLY	PB-T50-RS
Transf.-OcCo [49]	84.9	85.5	78.8
Point-BERT [49]	87.4	88.1	83.1
MaskPoint [28]	89.3	89.7	84.6
Point-MAE [31]	90.0	88.3	85.2
Point-M2AE [51]	**91.2**	88.8	86.4
from scratch	88.1	88.8	84.3
data2vec-PC	89.7	88.1	85.5
point2vec (ours)	**91.2**	**90.4**	**87.5**

Table 4. **Few-shot classification on ModelNet40 [44].** We report the mean and standard deviation over 10 runs.

Method	5-way		10-way	
	10-shot	20-shot	10-shot	20-shot
OcCo [43]	91.9 ± 3.6	93.9 ± 3.1	86.4 ± 5.4	91.3 ± 4.6
Transf.-OcCo [49]	94.0 ± 3.6	95.9 ± 2.3	89.4 ± 5.1	92.4 ± 4.6
Point-BERT [49]	94.6 ± 3.1	96.3 ± 2.7	91.0 ± 5.4	92.7 ± 5.1
MaskPoint [28]	95.0 ± 3.7	97.2 ± 1.7	91.4 ± 4.0	93.4 ± 3.5
Point-MAE [31]	96.3 ± 2.5	97.8 ± 1.8	92.6 ± 4.1	95.0 ± 3.0
Point-M2AE [51]	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**

During the point cloud embedding step, we sample $n = 64$ center points and $k = 32$ nearest neighbors. In Tab. 1, we report state-of-the-art results for shape classification on ModelNet40 among self-supervised methods by a margin of 1.3 percentage points without voting [31, 49, 51]. Interestingly, pre-training with data2vec-PC results in only marginal improvements (0.3 percentage points 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 percentage points over the baseline trained from scratch. In Fig. 3, we plot the accuracy per training epoch of point2vec, data2vec-PC, as well as our baseline trained from scratch on ModelNet40. We observe that point2vec outperforms our strong baselines by a consistent margin throughout the entire training. point2vec effectively learns strong feature representations on ShapeNet, resulting in a significantly accelerated adaptation to the fine-tuning task (Fig. 3).

Real-world shape classification. Next, we fine-tune point2vec on ScanObjectNN [40], which contains 2902 real-world object scans of 15 semantic classes. In contrast to shape classification on ModelNet40, we do not resample points but use all 2048 points and sample $n = 128$ center points for the point cloud embedding step. We found more aggressive scaling to be detrimental and use random anisotropic scaling of up to 10%. Table 3 shows that, although pre-trained on synthetic data, point2vec generalizes well to cluttered real-world data and achieves state-of-the-art performance among self-supervised methods by a margin of 1.1 percentage points on PB-T50-RS, the most difficult variant of the dataset. We observe that pre-training point2vec on ShapeNet plays a crucial role in its strong performance. Compared to the baseline trained from scratch on ScanObjectNN, pre-training with point2vec achieves a performance gain of 3.2 percentage points. We again report improvements of point2vec over data2vec-PC of up to 2.3 percentage points.

Few-shot classification. Following the few-shot evaluation protocol of Sharma and Kaul [37], we test the few-shot

capabilities of point2vec in an 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. Following prior work [31], we report the mean and standard deviation over 10 independent runs. In Tab. 4, we report state-of-the-art results, with an improvement of 1.3 percentage points 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.

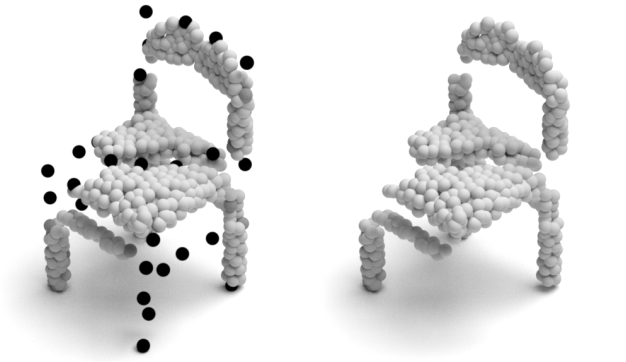
Part segmentation. Finally, we address the task of part segmentation, which assigns a semantic part label to each point in a 3D point cloud of a single object. For this purpose, we employ a simple segmentation head that is similar to the segmentation head in Point-MAE. First, we average the outputs of the 4th, 8th, and 12th Transformer blocks to incorporate features from multiple levels of abstraction. We then concatenate the mean- and max-pooled features 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 upsample the n averaged outputs from the corresponding center points to all points using a PointNet++ [33] 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. 2, we report competitive results on ShapeNetPart [48], which consists of 16 881 3D models of 16 semantic categories. Apart from Point-M2AE, point2vec outperforms all other self-supervised methods. We hypothesize that Point-M2AE's multi-scale U-Net-like architecture [35] enables it to learn more expressive spatially localized features, resulting 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 [11, 20, 27, 50] extending vision Transformers [18] with multi-scale capabilities.

4.3. Analysis

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. 4a. Revealing the positions of masked patches of the chair inadvertently weakens the learning objective because it allows the student to rely on the positional information instead of truly learning to predict the teacher's representations of the corresponding masked-out patches. 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. 4b are effectively concealed from the student network, leading to a more resilient learning framework. In Tab. 5, we report that point2vec outperforms our baseline data2vec-PC by a margin of up to 2.0 percentage points. 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, *i.e.* shifting mask tokens from the encoder's input to the decoder (No Ⓜ). Complementary to our findings, He et al. [23] 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. [31], who found similar benefits for masked autoencoders on point clouds.

Table 5. **Ablation.** We find that a deferred shallow decoder (D) (Fig. 2, ▢) predicting the teacher's representations for masked patches shows consistent improvements but we identify that concealing positional information (No Ⓜ) from the student is key.

Method	No Ⓜ	D	Overall accuracy		
			ModelNet40		ScanObjNN
			Voting	No voting	PB-T50-RS
data2vec-PC	✗	✗	93.6	93.3	85.5
	✗	✓	94.0	93.6	86.8
point2vec (ours)	✓	✓	**94.8**	**94.7**	**87.5**



(a) Disclosed positions: data2vec-PC (b) Concealed positions: **point2vec** (ours)

Figure 4. **Leakage of positional information.** The center points ● of masked point patches are associated with the masked embeddings (Fig. 2, Ⓜ). (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.

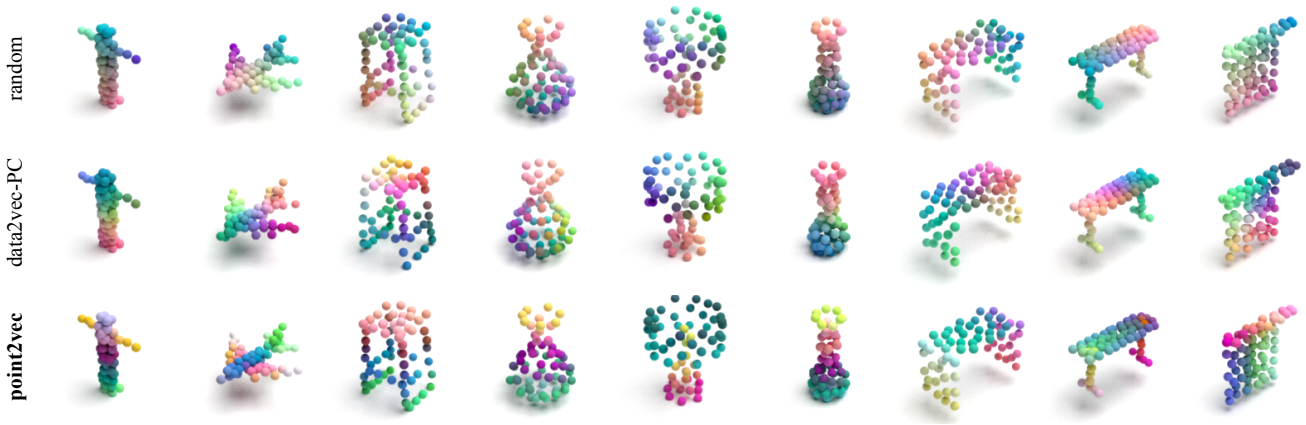
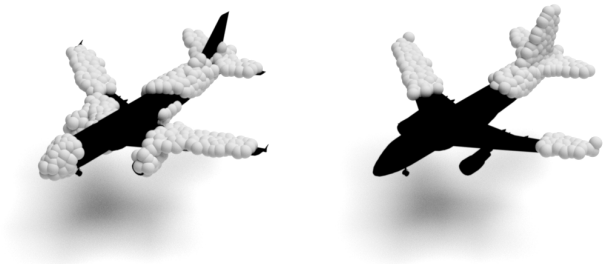


Figure 5. **Visualization of learned representations.** We use PCA to project the learned representations into RGB space. Both a random initialization and data2vec-PC pre-training show a fairly strong positional bias, whereas point2vec exhibits stronger semantic grouping without being trained on downstream dense prediction tasks.

Table 6. **Masking strategy.** We explore two variants for masking the input of the student. For (a) random masking, we uniformly mask out a given ratio of all embeddings. For (b) block masking, we mask out a random embedding and its nearest neighbors. We report the overall accuracy on ModelNet40 and ScanObjectNN.

(a) 65% random masking (b) 65% block masking



Strategy	Ratio (%)	Overall accuracy		
		ModelNet40		ScanObjNN
		Voting	No voting	PB-T50-RS
random	45	94.5	94.3	86.8
random	65	**94.8**	**94.7**	**87.5**
random	85	94.5	93.8	86.7
block	25	93.9	93.7	86.3
block	45	94.5	93.8	87.4
block	65	94.0	93.9	86.1

Masking strategy. The masking strategy defines which of the student's input embeddings are masked (Fig. 2, Ⓜ). In this study, we investigate two variants of masking strategies with different masking ratios: *random* masking and *block* masking. For random masking, we mask out a specified ratio of embeddings for the student. In contrast, block masking masks out a random embedding and its nearest neighbors such that the specified masking ratio is achieved. This strategy focuses on masking out spatially contiguous regions of the point cloud, whereas random masking is independent of position. Our findings, summarized in Tab. 6, reveal that random masking with a 65% masking ratio performs best for both ModelNet40 and ScanObjectNN, while block masking lags behind. We attribute this to the high level of ambiguity that arises when masking a spatially contiguous region, resulting in several potential point clouds that could have given rise to the masked input. While we seek a challenging pretext task to learn rich representations, ambiguity should not be the primary source of difficulty.

Recall that our masking strategy is applied to patch embeddings rather than individual points. Consequently, points may belong to both masked and unmasked patches. While certain masked patches may be easy to predict, our masking ratio of 65% ensures that there are still plenty of regions entirely masked (Fig. 4 and Tab. 6). As a result, we conclude that the masking ratio is a sensitive hyperparameter that requires some careful tuning to strike the right balance of difficulty for the pretext task.

Visualization of representations learned by point2vec. In Fig. 5, we show qualitative examples of representations of ModelNet40 instances after pre-training on ShapeNet. Both a random initialization and data2vec-PC pre-training show a strong positional bias, whereas point2vec exhibits 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, resulting in more semantically meaningful representations.

5. 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 limits 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 to scene-level representation learning.

Acknowledgments. This work has been funded, in part, by the ERC CoG grant DeeViSe (ERC-2017-CoG-773161) and by the BMBF project 6GEM (16KISK036K). We gratefully acknowledge computing resources granted by RWTH Aachen University (thes1313, supp0003) and thank István Sárándi for helpful discussions and feedback.

References

[1] 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 *CVPR*, 2016.

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

[3] 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 *ICML*, 2022.

[4] Hangbo Bao, Li Dong, Songhao Piao, and Furu Wei. BEiT: BERT pre-training of image transformers. In *ICLR*, 2022.

[5] Adrien Bardes, Jean Ponce, and Yann LeCun. VICReg: Variance-invariance-covariance regularization for self-supervised learning. In *ICLR*, 2022.

[6] Gilad Baruch, Zhuoyuan Chen, Afshin Dehghan, Tal Dimry, Yuri Feigin, Peter Fu, Thomas Gebauer, Brandon Joffe, Daniel Kurz, Arik Schwartz, and Elad Shulman. ARKitScenes: A diverse real-world dataset for 3D indoor scene understanding using mobile RGB-D data. In *NeurIPS Datasets and Benchmarks Track*, 2021.

[7] Jens Behley, Martin Garbade, Andres Milioto, Jan Quenzel, Sven Behnke, Cyrill Stachniss, and Jürgen Gall. SemanticKITTI: A dataset for semantic scene understanding of LiDAR sequences. In *ICCV*, 2019.

[8] Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuScenes: A multimodal dataset for autonomous driving. In *CVPR*, 2020.

[9] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *ICCV*, 2021.

[10] 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.

[11] Chun-Fu (Richard) Chen, Quanfu Fan, and Rameswar Panda. CrossViT: Cross-attention multi-scale vision transformer for image classification. In *ICCV*, 2021.

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

[13] Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. In *CVPR*, 2021.

[14] Christopher Choy, JunYoung Gwak, and Silvio Savarese. 4D spatio-temporal ConvNets: Minkowski convolutional neural networks. In *CVPR*, 2019.

[15] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. ScanNet: Richly-annotated 3D reconstructions of indoor scenes. In *CVPR*, 2017.

[16] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*, 2019.

[17] Runpei Dong, Zekun Qi, Linfeng Zhang, Junbo Zhang, Jianjian Sun, Zheng Ge, Li Yi, and Kaisheng Ma. Autoencoders as cross-modal teachers: Can pretrained 2D image transformers help 3D representation learning? In *ICLR*, 2023.

[18] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021.

[19] Benjamin Eckart, Wentao Yuan, Chao Liu, and Jan Kautz. Self-supervised learning on 3D point clouds by learning discrete generative models. In *CVPR*, 2021.

[20] Haoqi Fan, Bo Xiong, Karttikeya Mangalam, Yanghao Li, Zhicheng Yan, Jitendra Malik, and Christoph Feichtenhofer. Multiscale vision transformers. In *ICCV*, 2021.

[21] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H. Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, Bilal Piot, Koray Kavukcuoglu, Rémi Munos, and Michal Valko. Bootstrap your own latent: A new approach to self-supervised learning. In *NeurIPS*, 2020.

[22] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *CVPR*, 2020.

[23] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *CVPR*, 2022.

[24] Ji Hou, Benjamin Graham, Matthias Nießner, and Saining Xie. Exploring data-efficient 3D scene understanding with contrastive scene contexts. In *CVPR*, 2021.

[25] 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 *ICASSP*, 2021.

[26] Siyuan Huang, Yichen Xie, Song-Chun Zhu, and Yixin Zhu. Spatio-temporal self-supervised representation learning for 3D point clouds. In *ICCV*, 2021.

[27] 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 *CVPR*, 2022.

[28] Haotian Liu, Mu Cai, and Yong Jae Lee. Masked discrimination for self-supervised learning on point clouds. In *ECCV*, 2022.

[29] Ilya Loshchilov and Frank Hutter. SGDR: Stochastic gradient descent with warm restarts. In *ICLR*, 2017.

[30] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2019.

[31] Yatian Pang, Wenxiao Wang, Francis E. H. Tay, Wei Liu, Yonghong Tian, and Li Yuan. Masked autoencoders for point cloud self-supervised learning. In *ECCV*, 2022.

[32] Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. PointNet: Deep learning on point sets for 3D classification and segmentation. In *CVPR*, 2017.

[33] Charles R. Qi, Li Yi, Hao Su, and Leonidas J. Guibas. PointNet++: Deep hierarchical feature learning on point sets in a metric space. In *NeurIPS*, 2017.

[34] Yongming Rao, Jiwen Lu, and Jie Zhou. Global-local bidirectional reasoning for unsupervised representation learning of 3D point clouds. In *CVPR*, 2020.

[35] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional networks for biomedical image segmentation. In *MICCAI*, 2015.

[36] Jonas Schult, Francis Engelmann, Alexander Hermans, Or Litany, Siyu Tang, and Bastian Leibe. Mask3D: Mask transformer for 3D semantic instance segmentation. In *ICRA*, 2023.

[37] Charu Sharma and Manohar Kaul. Self-supervised few-shot learning on point clouds. In *NeurIPS*, 2020.

[38] Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive multiview coding. In *ECCV*, 2020.

[39] Yonglong Tian, Chen Sun, Ben Poole, Dilip Krishnan, Cordelia Schmid, and Phillip Isola. What makes for good views for contrastive learning? In *NeurIPS*, 2020.

[40] 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 *ICCV*, 2019.

[41] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

[42] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017.

[43] Hanchen Wang, Qi Liu, Xiangyu Yue, Joan Lasenby, and Matthew J. Kusner. Unsupervised point cloud pre-training via occlusion completion. In *ICCV*, 2021.

[44] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3D ShapeNets: A deep representation for volumetric shapes. In *CVPR*, 2015.

[45] Saining Xie, Jiatao Gu, Demi Guo, Charles R. Qi, Leonidas Guibas, and Or Litany. PointContrast: Unsupervised pre-training for 3D point cloud understanding. In *ECCV*, 2020.

[46] Le Xue, Mingfei Gao, Chen Xing, Roberto Martín-Martín, Jiajun Wu, Caiming Xiong, Ran Xu, Juan Carlos Niebles, and Silvio Savarese. ULIP: Learning a unified representation of language, images, and point clouds for 3D understanding. In *CVPR*, 2023.

[47] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R. Salakhutdinov, and Quoc V. Le. XLNet: Generalized autoregressive pretraining for language understanding. In *NeurIPS*, 2019.

[48] 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 TOG*, 2016.

[49] 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 *CVPR*, 2022.

[50] 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 *ICCV*, 2021.

[51] 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 *NeurIPS*, 2022.

[52] Renrui Zhang, Ziyu Guo, Wei Zhang, Kunchang Li, Xupeng Miao, Bin Cui, Yu Qiao, Peng Gao, and Hongsheng Li. PointCLIP: Point cloud understanding by CLIP. In *CVPR*, 2022.

[53] Renrui Zhang, Liuhui Wang, Yu Qiao, Peng Gao, and Hongsheng Li. Learning 3D representations from 2D pre-trained models via image-to-point masked autoencoders. In *CVPR*, 2023.

point2vec for Self-Supervised Representation Learning on Point Clouds

Supplementary Material

We provide additional ablation studies, architecture and training details, and qualitative part-segmentation results. Our code and model are publicly available for research purposes.

A. Further Ablation Studies

Pretext task. In the main paper, we have only explored self-supervised pre-training on the ShapeNet dataset [10]. However, ShapeNet also contains class labels which could instead be used for fully supervised classification-based pre-training. As can be seen in Tab. 7, this performs substantially worse than point2vec and offers no improvement over training from scratch. We can also pre-train using point2vec directly on ModelNet40 [44], which constitutes roughly a quarter of ShapeNet's size. Still, we see improved downstream performance, indicating that the point2vec pretext task is meaningful for pre-training.

EMA rate warm-up. 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 [3]. This approach is based on the idea that we should update the teacher network more rapidly at the start of training since the feature representations are not yet well established. In Tab. 8, we show overall accuracy scores on ModelNet40 and the PB-T50-RS variant of ScanObjectNN [40] 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. 11, we provide the EMA rates employed by our baseline data2vec-PC and point2vec, respectively.

Target layer aggregation. During training of data2vec-PC, as well as point2vec, we need to specify which layers of the teacher to use for the target. The target is constructed by averaging the last K layers of the teacher; Baevski et al. [3] use $K = 6$ for their vision models. We ablate this hyperparameter and report results in Tab. 9. Although all tested values lead to usable results, $K = 6$ gives the best overall 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-tune our model for shape classification on ModelNet40 and ScanObjectNN. As shown in Fig. 7, 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 [31] pre-trained with 100% of the training data.

Table 7. **Pretext tasks.** Pre-training with a classification objective on ShapeNet offers no gains over training from scratch on ModelNet40 and yields worse performance on the PB-T50-RS variant 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.

Pretext task	Overall accuracy		
	ModelNet40		ScanObjNN
	Voting	No 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 8. **EMA rate warm-up.** We linearly warm up the EMA rate during the first τ_n epochs.

τ_n	Overall accuracy		
	ModelNet40		ScanObjNN
	Voting	No 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

B. Architecture Details

In Tab. 11, we provide detailed hyperparameters for pre-training data2vec-PC and point2vec on ShapeNet [10]. We also report the hyperparameters for fine-tuning point2vec on shape classification (Tab. 12) and part segmentation (Tab. 10). In Pseudocode 1, we provide the PyTorch-inspired pseudocode for point2vec pre-training. The decoder outputs are mapped to the target space by a learned linear projection.

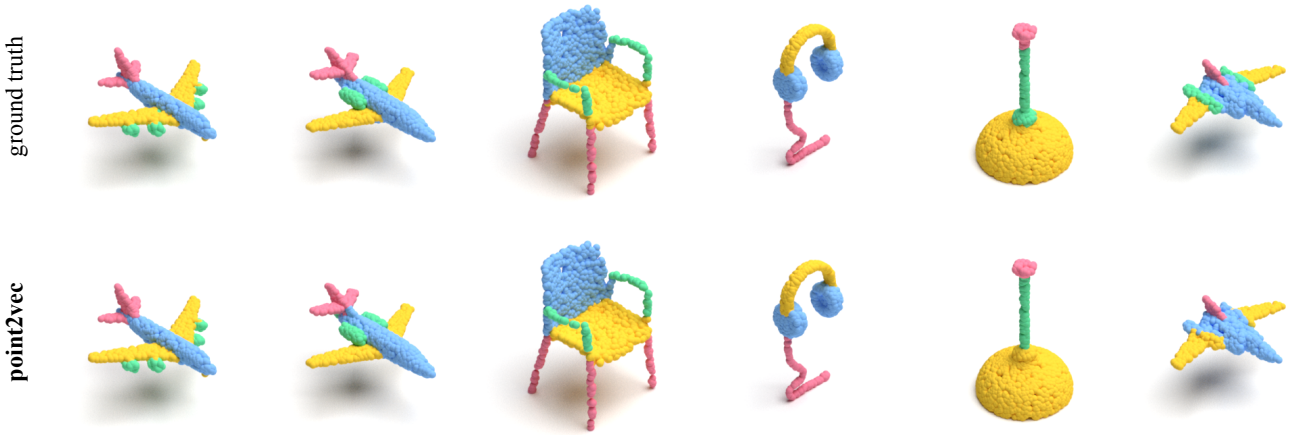
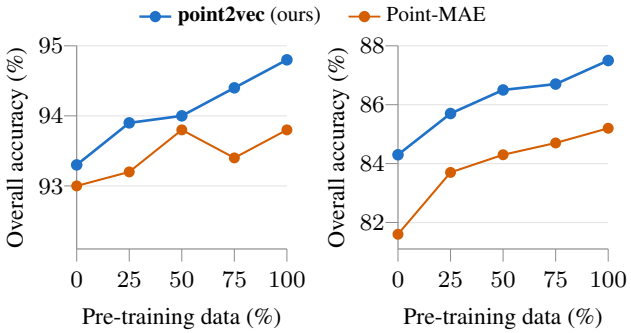


Figure 6. **Qualitative results on ShapeNetPart [48].** point2vec produces well-localized boundaries between parts with minimal semantic errors. In most cases, the differences between the results of point2vec and the ground truth are imperceptible to the human eye. However, the last example shows a failure case where the jet engine is not properly segmented.

Table 9. **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.

K	Overall accuracy		
	ModelNet40		ScanObjNN
	Voting	No 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



(a) ModelNet40 [44] (b) ScanObjectNN [40]

Figure 7. **Pre-training data efficiency.** Irrespective of the quantity of pre-training data used from ShapeNet, point2vec consistently achieves better results than Point-MAE [31] on ModelNet40 (with voting) and the PB-T50-RS variant of ScanObjectNN.

Table 10. **Hyperparameters for part segmentation.** We use the same hyperparameters when fine-tuning point2vec and data2vec-PC on ShapeNetPart [48].

Parameter	Value
Epochs	300
Batch size	16
Optimizer	AdamW
Learning rate	2×10^{-4}
Weight decay	0.05
LR schedule	cosine
LR warm-up	10 epochs
Points	2048
n (center points)	128
k (k-NN grouping)	32
mini-PointNet MLP 1	128, 256
mini-PointNet MLP 2	512, 384
Encoder layers	12
Encoder dimension	384
Encoder heads	6
Encoder drop path	$0\%, \ldots, 20\%$
Feature propagation	See Sec. 4.2
Head dimensions	512, 256, #classes
Head dropout	50%

C. Qualitative Results for Part Segmentation

Figure 6 shows qualitative results on ShapeNetPart [48]: point2vec accurately localizes part boundaries, and most predictions visually match the ground truth.

Table 11. **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.

Parameter	data2vec-PC	point2vec (ours)
Epochs	800	800
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)	0.999 99	0.999 99
τ_n (EMA warm-up)	200 epochs	200 epochs
K (layers to average)	6	6
Target normalization	LN→AVG→LN	LN→AVG→LN

Table 12. **Hyperparameters for classification.** Shared settings for point2vec and data2vec-PC on ModelNet40 [44] and ScanObjectNN [40]. Paired values list ModelNet40 first and ScanObjectNN second. Training from scratch uses a learning rate of 1×10^{-3} and no encoder freezing.

Parameter	Value
Epochs	150
Batch size	32
Optimizer	AdamW
Learning rate	3×10^{-4}
Weight decay	0.05
LR schedule	cosine
LR warm-up	10 epochs
Points	1024 / 2048
n (center points)	64 / 128
k (k-NN grouping)	32
mini-PointNet MLP 1	128, 256
mini-PointNet MLP 2	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
Head dimensions	256, 256, #classes
Head dropout	50%
Label smoothing	0.2

Pseudocode 1: **PyTorch-inspired pre-training for point2vec.**

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

for point_cloud in data_loader:
    # point cloud embedding
    center_points = fps(point_cloud)  # (N, S, 3)
    point_patches = knn(point_cloud, center_points)  # (N, S, 32, 3)
    point_patches = point_patches - center_points.unsqueeze(dim=2)
    patch_embeddings = mini_pointnet(point_patches)  # (N, S, E) (Fig. 2, )
    pos_embeddings = pos_encoder(center_points)  # (N, S, E)

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

    # targets
    with torch.no_grad():
        teacher_states = teacher(patch_embeddings, pos_embeddings)  # (12, N, S, E) (Fig. 2, )
        target_layers = [F.layer_norm(x, (E,)) for x in teacher_states[6:]]  # [(N, S, E)]
        targets = torch.stack(target_layers).mean(dim=0)  # (N, S, E)
        targets = F.layer_norm(targets, (E,))  # (N, S, E)

    # predictions
    last_student_state = student(  # (N, V, E) (Fig. 2, )
        patch_embeddings[~mask].reshape(N, -1, E),
        pos_embeddings[~mask].reshape(N, -1, E),
    )[-1]
    predictions = decoder(  # (N, S, E) (Fig. 2, )
        mask_embeddings.index_put([~mask], last_student_state.reshape(-1, E)),
        pos_embeddings,
    )[-1]
    predictions = regressor(predictions)  # (N, S, E)

    # optimization
    optimizer.zero_grad()
    loss = F.smooth_l1_loss(predictions[mask], targets[mask], beta=2)
    loss.backward()
    optimizer.step()

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