Asymmetric Dual Self-Distillation for 3D Self-Supervised Representation Learning

Remco F. Leijenaar* **Hamidreza Kasaei**
Department of AI, University of Groningen, The Netherlands

Abstract

Learning semantically meaningful representations from unstructured 3D point clouds remains a central challenge in computer vision, especially in the absence of large-scale labeled datasets. While masked point modeling (MPM) is widely used in self-supervised 3D learning, its reconstruction-based objective can limit its ability to capture high-level semantics. We propose AsymDSD, an Asymmetric Dual Self-Distillation framework that unifies masked modeling and invariance learning through prediction in the latent space rather than the input space. AsymDSD builds on a joint embedding architecture and introduces several key design choices: an efficient asymmetric setup, disabling attention between masked queries to prevent shape leakage, multi-mask sampling, and a point cloud adaptation of multi-crop. AsymDSD achieves state-of-the-art results on ScanObjectNN (90.53%) and further improves to 93.72% when pretrained on 930k shapes, surpassing prior methods.

1 Introduction

As domains such as robotics, autonomous driving, AR/VR, and remote sensing continue to grow, the importance of three-dimensional (3D) data becomes increasingly pronounced. A central challenge in 3D vision lies in learning semantically meaningful representations from unstructured 3D point clouds. In contrast to 2D computer vision—where large labeled datasets like ImageNet [1] have played a prominent role in driving progress—3D datasets remain limited in both scale and diversity (see Fig. 1). This scarcity, which has been referred to as the *data desert* problem [2], is exacerbated by the difficulty of annotating 3D data, particularly at the point level [3]. Although recent efforts such as Objaverse [4, 5] mark progress toward web-scale 3D data collection, they still lag behind the scale achieved in 2D vision and natural language processing (NLP). The lack of labeled 3D data has fueled a growing interest in self-supervised representation learning (SSL/SSRL) for 3D understanding [6]. SSL has proven highly effective in both NLP [7, 8] and 2D vision [9–12], offering strong scalability [13–16], and robust down-stream capabilities [17, 18, 16], even with minimal labeled data [19]. These successes have inspired SSRL techniques to be adapted to point cloud data. Particularly, masked point modeling (MPM) approaches have gained traction [20–26], given their effectiveness and conceptual alignment with masked modeling frameworks for 2D and NLP.

Despite their popularity, we argue that MPM-based approaches have fundamental limitations. Reconstruction objectives tend to emphasize short-range dependencies and high-frequency details [27, 28], which are often dominated by noise rather than semantically meaningful structure. This issue is exacerbated in complex 3D geometries, where undersampling introduces high target variance. More broadly, MPM may not lead to the semantic abstraction crucial for robust downstream performance. This is backed by empirical findings that demonstrate that these models underperform compared to invariance-based alternatives in low-shot and linear probing regimes [19]. To overcome these limitations, several works have attempted to fuse generative reconstruction with invariance-based objectives [21, 26]. However, the pattern differences of these objectives can cause them to interfere

*Code available at `https://github.com/RFLeijenaar/AsymDSD`

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

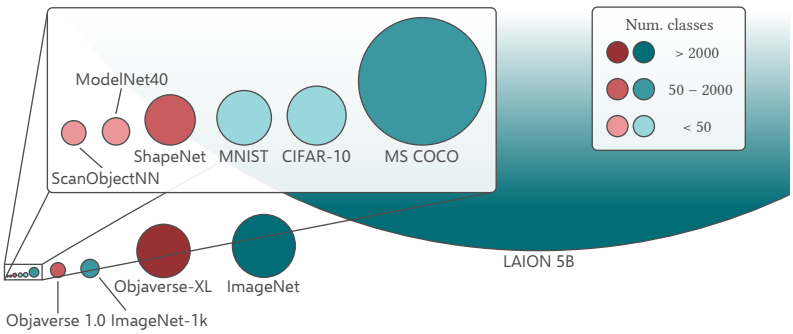


Figure 1: A depiction of the size of some well-known predominantly object-centered datasets in the computer vision domain. The blue color corresponds to (2D) image datasets, and red to 3D datasets.

with one another, resulting in worse performance than using reconstruction alone. Notably, ReCon [26] solves this through a two-tower architecture that separates the objectives, but this comes with significantly increased computational overhead.

In light of these issues, we seek a more elegant integration of local masked modeling and global invariance learning. Particularly, we posit a reframing of the MPM paradigm: rather than predicting the input from the latent, what if we instead predict the latent representations themselves? This shift aligns with the philosophy behind works such as CPC [29], data2vec [30, 31], and I-JEPA [32], that argue for learning semantics through prediction in latent space. The intuition stems from the notion that semantically meaningful features are those that are predictive across spatial contexts, even when fine details are occluded. For example, given only a visible wing of an airplane, one cannot reasonably predict the exact geometry of the tail, but one can predict the presence of a tail—a semantic category.

Building on this insight, we introduce **AsymDSD**: an **Asym**metric **D**ual **S**elf-**D**istillation framework for 3D point clouds (Fig. 2). AsymDSD effectively combines predictive masked modeling with invariance learning, using a joint embedding architecture (JEA) trained through self-distillation from a momentum teacher. The framework is designed with efficiency in mind, and addresses issues such as representation collapse and shape leakage, taking inspiration from a variety of recent works in SSRL on images. We summarize the core components and main contributions as follows:

- **Dual Self-Distillation Objectives**: The model jointly optimizes (1) a patch-level latent masked point modeling (MPM) objective through same-view self-distillation on masked tokens, and (2) a global invariance learning objective using cross-view self-distillation [33, 18]. The representations are projected to a distribution over a discrete latent variable, thereby explicitly modeling a posterior. This gives more direct control to overcome representation collapse and stabilize training.
- **Asymmetric Architecture**: The student model, unlike the teacher, hosts an encoder-predictor design. Given effective high mask ratios [22], the relatively heavy encoder processes only a small number of visible patches, while a lightweight predictor builds the representations of masked patches. Unlike previous methods [34, 22], we disable self-attention over the mask queries, which avoids leaking the global shape through the positional queries, and further enhances efficiency.
- **Multi-Mask**: To amortize the cost of the additional teacher, we introduce *multi-mask* [31], which samples multiple independent masks per point cloud. This allows targets and non-contextualized student embeddings to be reused across masks, effectively increasing batch size at minimal cost.
- **Multi-Crop**: While many point cloud-specific augmentations fall short in enforcing a challenging invariance objective, the modality-agnostic cropping augmentations proves effective. In particular, we adapt *multi-crop* [35] to point clouds, encouraging the model to efficiently learn robust local-to-global mapping capabilities.

To evaluate the effectiveness of AsymDSD, we adopt the common ShapeNet [36] pretraining protocol and transformer backbone [21], ensuring a fair comparison. Under this setup, AsymDSD achieves state-of-the-art performance on ScanObjectNN [37], reaching 90.53% accuracy, which is a 5.35% absolute improvement over the Point-MAE baseline [22]. The method also exhibits strong generalization in few-shot settings, as demonstrated on ModelNet40 [38]. Furthermore, we run ablations on our training framework to show that the proposed components contribute substantial improvements.

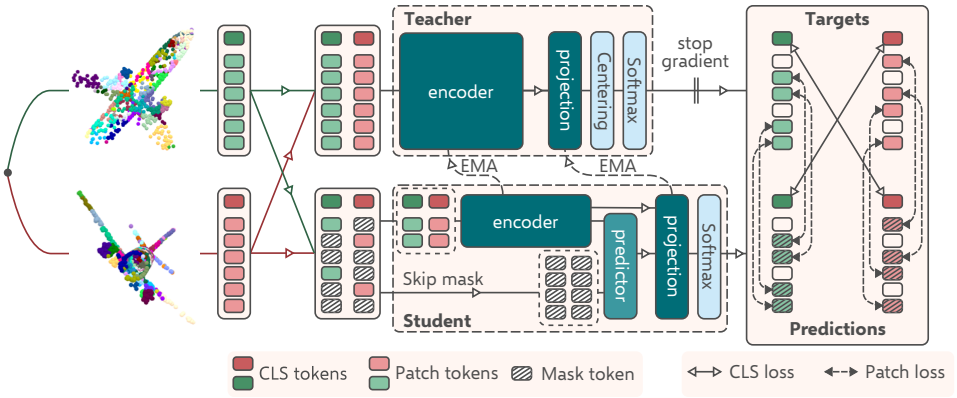


Figure 2: High level overview of **AsymDSD**. The diagram highlights the asymmetry between the teacher and student networks, and shows the distillation of knowledge from the momentum encoded (EMA) teacher on both a cross-view global (CLS) and same-view patch level. The block widths indicate the number of patches processed, while their lengths represent network depth. The student's efficient design is reflected in its deep but narrow encoder and wide but shallow predictor.

To assess scalability, we pretrain AsymDSD on a large composite dataset incorporating synthetic and scanned point clouds, including Objaverse [4], totaling over 930,000 shapes. Pre-training on this large dataset attains 93.72% accuracy on ScanObjectNN, a new single-modal SOTA with a standard transformer, exceeding PointGPT-L [24] by 2.6%.

2 Related Work

SSRL in 2D Vision has evolved along two main paradigms: *discriminative* (or invariance-based) and *generative* approaches. Discriminative methods primarily focus on overcoming *representation collapse*: a mode wherein representations become constant, and thus independent of the input. Approaches such as SimCLR [10] and MoCo [39] employed contrastive learning, which relies on comparing positive pairs (augmentations of the same image) against a large set of negatives. However, these methods require large batch sizes or memory banks to be effective. Subsequent works, such as BYOL [9] and SimSiam [40], demonstrated that contrastive negatives are dispensable. They utilize a student-teacher setup with architectural asymmetry to stabilize training and avoid collapse. Follow-up studies further explored this space by integrating regularization techniques like mean entropy maximization [41, 19], clustering constraints [35], and representation decorrelation [42, 43].

Parallel to this, denoising autoencoders [44], have evolved into masked image modeling (MIM) approaches [45, 12]. MAE, in particular, employs an efficient encoder-decoder setup that leverages the transformer's capacity to process sparse inputs. BEiT [46] introduced discrete token prediction using a pre-trained dVAE [47], however its targets lack high-level semantics [33]. Hybrid methods have since emerged that blend MIM with joint embedding architectures. Notably, data2vec [30, 31] and I-JEPA [32] reframe MIM as latent representation prediction, regressing contextualized embeddings produced by a momentum teacher instead of raw pixels. Differently, iBOT [33] and DINOv2 [18] include a global invariance objective to ensure semantically rich targets for the MIM objective.

SSRL for Point Clouds. The success of SSRL in 2D vision has inspired analogous developments in 3D point cloud learning. Early invariance-based methods adopted contrastive frameworks [48–50] but also BYOL-style [51] or DINO-style [52] training. With the adoption of the standard transformer, point cloud-specific adaptations of MIM emerged. Point-BERT [21] and Point-MAE [22] employ masked token prediction strategies analogous to their image counterparts. To avoid shape leakage through the mask queries, PointGPT [24] sequentializes the point patches to enable autoregressive modeling. In contrast, we demonstrate that a simpler and more efficient alternative—disabling attention on mask tokens—is equally effective. point2vec [34] builds on data2vec [30] with an added predictor module, yet it does not address shape leakage at the predictor stage. ReCon [26] successfully combines masked point modeling (MPM) with multi-modal invariance learning but introduces added complexity through its dual-tower architecture. In comparison, our method seamlessly incorporates invariance learning without any changes to the underlying model architecture.

3 AsymDSD: Asymmetric Dual Self-Distillation

AsymDSD unfies two complementary self-supervised objectives, which we initially introduce as independent SSRL approaches. While not tied to a specific model architecture, we present it under a ViT-style [53] backbone with a lightweight PointNet-based patch embedding module, following Point-BERT [21]. A more detailed description of this architecture can be found in Appendix A.

3.1 Global Objective: Invariance learning

The global objective of AsymDSD innvolves the learning of representations with an abstraction of semantics that is discriminative to individual inputs, up to a set of data augmentations. For this purpose, it integrates knowledge distillation [54] and invariance learning in an end-to-end framework. Specifically, given an input sample $\mathrm{x} \sim p(\mathrm{x})$, latent variable $z \in \mathcal{Z}$, and two augmentations $\mathrm{t}_1, \mathrm{t}_2 \sim \mathcal{T}$, the invariance-based distillation objective is:

$$\theta^* = \underset{\theta \in \Theta}{\operatorname{argmin}} \, \mathbb{E}_{\mathrm{x} \sim p(\mathrm{x}),\, \mathrm{t}_1, \mathrm{t}_2 \sim \mathcal{T}} \left[D_{\mathrm{KL}} \left(p^t_{\theta'}(\mathrm{z} \mid \mathrm{t}_1(\mathrm{x})) \parallel p^s_\theta(\mathrm{z} \mid \mathrm{t}_2(\mathrm{x})) \right) \right] \tag{1}$$

where $p_\theta(z \mid x) = \mathrm{softmax}(f_\theta(x)/\tau)_z$, with τ a temperature to control the sharpness. In SSRL, the teacher posterior $p^t_{\theta'}(\mathrm{z}|\mathrm{x})$ is not known ahead of training. Instead, past iterations of the student serve as a proxy for the teacher. Particularly, we use an exponential moving average (EMA) of the student's parameter: $\theta' \leftarrow \eta\theta' + (1-\eta)\theta$, where θ' are the teacher and θ the student parameters, and η a decay rate.

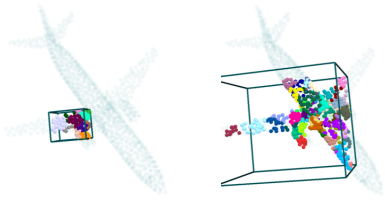
Iteratively optimizing Equation 1, may lead to pathological solutions in absence of additional constraints. A 'shortcut' in minimizing any such objective is for the latent z to become independent of the input x, i.e. $p_\theta(\mathrm{z}) = p_\theta(\mathrm{z} \mid \mathrm{x})$. This is particularly implied by the scenarios:

1. $H\left(p_\theta(\mathrm{z})\right) = H(\mathbb{E}_{\mathrm{x} \sim p(\mathrm{x})}[p_\theta(\mathrm{z} \mid \mathrm{x})]) = 0$, i.e., the marginal entropy becomes zero. The model collapses to a single latent representation z that is always assigned a probability of 1.
2. $H(p_\theta(\mathrm{z} \mid \mathrm{x})) = \log |\mathcal{Z}|$, i.e., the posterior entropy becomes maximal. The model always outputs a uniform distribution over the latent space $\mathcal{Z}$.

To overcome these modes of posterior collapse, we follow DINO [11], by applying *centering* and *sharpening*. Centering prevents scenario (1) by subtracting a running mean from the teacher logits, thereby reducing the logits of the latent z that are frequently assigned high probability, thus helping to avoid low-entropy collapse. Sharpening prevents scenario (2) by setting a lower teacher softmax temperature compared to the student: $\tau_t < \tau_s$. This makes the posterior distribution more peaked, pushing it away from a uniform distribution and thus avoiding high-entropy collapse.

Cropping Augmentation. In principle, one can omit view-invariance by using identical inputs—reducing the objective to some form of mutual information maximization with centering and sharpening; this does not necessarily lead to useful representations [55, 56]. Instead, strong performance in self-supervised learning is often tied to carefully designed view augmentations. However, rather than relying on hand-crafted occlusions or corruptions [20, 57], we follow recent image-based SSRL trends that favor modality-agnostic augmentations such as masking or cropping [30, 32]. Specifically, AsymDSD generates crops by sampling randomly rotated bounding boxes with variable aspect ratios, rescaled to retain a random fraction c of the original points (see Fig. 3).

Multi-Crop. Furthermore, we adopt *multi-crop* [35] to point clouds by generating two global crops x^g_1 and x^g_2 that cover a large fraction of the point cloud ($c > 0.4$), along with several local crops x^l_i that may include as little as 5% of the original points ($0.05 < c < 0.4$). To simplify the implementation, all crops are subsampled to a fixed number of points, with local crops containing one-quarter the points of global crops. This sets up a challenging objective that encourages learning globally consistent representations from highly localized views.



(a) Local crop, $c = 0.1$ (b) Global crop, $c = 0.6$

Figure 3: Local and global crops.

Particularly, we define the *multi-crop* loss over the full set of crops $\mathcal{V} = \left\{x_1^{\mathrm{g}}, x_2^{\mathrm{g}}, x_1^{\mathrm{l}}, \ldots, x_{N_{\mathrm{L}}}^{\mathrm{l}}\right\}$ as:

$$\mathcal{L}^{\mathrm{MC}}(\mathcal{V}) = \frac{1}{2|\mathcal{V}|} \sum_{u \in \left\{x_1^g, x_2^g\right\}} \sum_{v \in \mathcal{V} \setminus \{u\}} \mathcal{L}^{\mathrm{CLS}}(u, v), \tag{2}$$

$$\mathcal{L}^{\mathrm{CLS}}(u, v) = -P_{\theta'}^t(u)^{\intercal} \log P_\theta^s(v), \tag{3}$$

where $P_{\theta'}^t(u)$, and $P_\theta^s(v)$ are the probability mass vectors. These are obtained by attaching a dedicated CLS-token to each view to aggregate global shape information. The logits over the latent are computed via a separate projection module [11] to process this global representation: $f_\theta(u) = h_\omega^{\mathrm{proj}}(f_\phi^{\mathrm{enc}}(u)_{\mathrm{CLS}})$. Additionally, a KoLeo loss [58, 18] is added to further encourage diversity in representations within a batch.

3.2 Patch-Level Objective: Masked Point Modeling (MPM)

While the global objective encourages inter-instance invariance and discrimination, the local objective promotes intra-instance predictability. Specifically, we adopt a masked point modeling (MPM) objective to learn spatially contextualized representations.

Given a *patchified* input consisting of N_c patches, denoted by $\boldsymbol{x} = (\boldsymbol{X}^P, \boldsymbol{c})$, where $\boldsymbol{X}^P$ represents a collection of local point groups and $\boldsymbol{c}$ denotes the corresponding center points for these groups, we define a binary mask $m \in \{0,1\}^{N_c}$ with a masking ratio M_r. The set of masked patch indices is given by $\mathcal{M} = \{i \mid m_i = 1\}$, and the set of visible patch indices by $\tilde{\mathcal{M}} = \{i \mid m_i = 0\}$. With latent MPM, the model trained on maximizing the log-posterior where the conditional comprises the visible context $\tilde{\boldsymbol{x}} = \left(\boldsymbol{X}_{\tilde{\mathcal{M}}}^{\boldsymbol{P}}, \boldsymbol{c}_{\tilde{\mathcal{M}}}\right)$ accompanied by a position query $\mathrm{c_i}$ such that $\mathrm{i} \in \mathcal{M}$, i.e.

$$\mathbb{E}_{(\mathbf{x}, \mathrm{z_i}) \sim q_\phi(\mathbf{x}, \mathrm{z_i}), \mathcal{M}} \left[\sum_{i \in \mathcal{M}} \log p_\theta(\mathrm{z_i} \mid \tilde{\mathbf{x}}, \mathrm{c_i}) \right]. \tag{4}$$

In simple terms, this expectation measures how well the model can predict the latent variable z_i corresponding to the center $\boldsymbol{c}_i$ given a visible context $\tilde{\boldsymbol{x}}$. However, we cannot jointly sample $\boldsymbol{x}$ and z_i as we do not have a model $q_\phi(\mathbf{x}, \mathrm{z_i})$. While variational methods such as BEiT [46] and Point-BERT [21] are enticing due to their mathematical interpretation [59], it remains guided by reconstruction-based learning, compromising semantic abstraction. Instead, we adopt a dynamic momentum teacher $p_{\theta'}^t$ similar to the global objective, allowing us to reformulate MPM as:

$$R^{\mathrm{MPM}}(\theta, \theta') = \mathbb{E}_{\mathrm{x} \sim p(\mathrm{x}),\, \mathcal{M}} \left[- \sum_{i \in \mathcal{M}} \mathbb{E}_{\mathrm{z}_i \sim p_{\theta'}^t(\mathrm{z}_i \mid \mathbf{x})} \left[\log p_\theta^s(\mathrm{z}_i \mid \tilde{\mathbf{x}}, \mathbf{c}_i) \right] \right] \tag{5}$$

In practice, the targets $\mathrm{z}_i \sim p_{\theta'}^t(\mathrm{z}_i \mid \mathbf{x})$ are not sampled from the teacher posterior as we use a latent with finite support. This enables the enumeration of all latent realizations for the exact computation of the cross-entropy as well as centering and sharpening to avoid collapse. Still, the design of the parameterized student f_θ^s and teacher $f_{\theta'}^t$ model underlying their respective posterior density function is crucial in ensuring a latent that represents semantically abstract information under the objective R^{MPM}. While symmetric architectures (e.g., denoising encoders [33, 30]) are plausible, we adopt an asymmetric design with the student hosting an encoder-predictor setup:

$$f_\theta^s(\tilde{\boldsymbol{x}}, \boldsymbol{c}_{\mathcal{M}}) \triangleq \left(h_\omega^{\mathrm{proj}} \circ g_\psi^{\mathrm{pred}}\right) \left(f_\phi^{\mathrm{enc}}(\tilde{\boldsymbol{x}}), \boldsymbol{c}_{\mathcal{M}}\right), \tag{6}$$

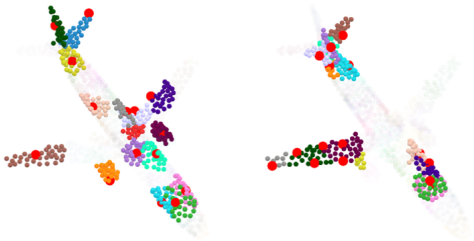
$$f_{\theta'}^t(\boldsymbol{x}) \triangleq \left(h_{\omega'}^{\mathrm{proj}} \circ f_{\phi'}^{\mathrm{enc}}\right)(\boldsymbol{x}), \tag{7}$$

where f^{enc} is a contextualizing encoder processing the visible context $\tilde{\boldsymbol{x}}$; g^{pred} a prediction module that predicts the contextualized embeddings for each mask query $\bar{\boldsymbol{e}}_i = \left(\boldsymbol{e}^{\mathrm{MASK}}, \boldsymbol{e}_i^{\mathrm{pos}}\right)$ given the encoded visible context. $\boldsymbol{e}^{\mathrm{MASK}}$ is a learnable mask token, and $\boldsymbol{e}_i^{\mathrm{pos}}$ the encoded position signal $\boldsymbol{c}_i$; h^{proj} is a projection head, that separately projects each patch embedding to the discrete latent z_i.

Figure 4: Leakage of the coarse shape via the positions of masked patches.

We consider the following benefits of this asymmetric design: (1) without the predictor, positional queries $c_{\mathcal{M}}$ inadvertently leak global spatial structure at the encoder stage (as illustrated in Fig. 4).

By deferring these queries to the predictor they can be effectively 'masked' by disabling attention between them. (2) Our encoder does not process masked embeddings during SSRL. This reduces the distribution mismatch between pre-training and downstream tasks, improving transferability. (3) The asymmetry due to a student predictor mitigates representations collapse and enhances training stability [60, 32]. Without this, the model may learn trivial solutions ignoring any contextual information. For example, a partitioning of space such that the centers $\mathbf{c}_\text{i}$ are uniformly projected across spatial bins $\mathcal{Z}$ would maximize the marginal entropy $H\left(p_\theta(\text{z})\right)$, rendering centering ineffective in overcoming this form of collapse. (4) Due to high mask ratios, the compute heavy encoder only processes a small number of visible patches. By contrast, the predictor that processes all the masked patches can be lightweight, thereby greatly increasing training efficiency.



(a) Uniform (b) Inverse block-wise

Figure 5: Masking strategies.

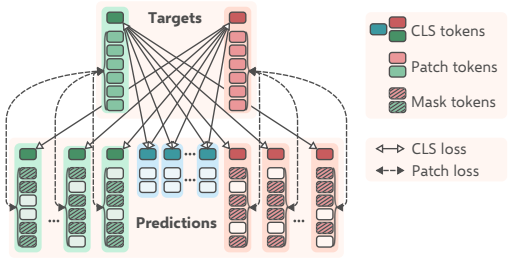


Figure 6: The losses of AsymDSD with multi-mask and multi-crop with both global (red and green) and local (blue) crops.

Masking Strategy. When it comes to the masking strategy, we observe that uniform masking (Fig. 5a) makes it generally easy to infer global structure due to the wide spatial distribution of visible patches. To address this, we implement inverse block-wise masking (Fig. 5b), which retains only a few small contiguous regions. This forces the model to infer the global shape from finer-grained details in a localized area. Specifically, we sample multiple fixed-sized blocks to add to the visible context via k-NN on center points $\boldsymbol{c}$. To account for possible block overlap, the final mask is adjusted by randomly flipping masked or unmasked bits to achieve the target mask ratio.

Multi-Mask. Although the teacher $f_{\theta'}^t$ only performs a forward pass, it is comparatively costly due to full-context encoding. To amortize this cost, a *multi-mask* strategy is applied, where multiple masks $m^{(i)}$ are sampled per input x, averaging the MPM loss across them. This increases the effective batch size at a fraction of the usual cost, as the teacher targets as well as the non-contextualized student patch embeddings can be reused across masks.

3.3 Dual Objective Learning

AsymDSD unifies global invariance learning and local masked point modeling (MPM) within a single framework. Since both objectives operate in a latent space, it enables parameter sharing without architectural entanglement. Specifically, the encoder f_ϕ^enc is shared across objectives, while separate projection heads are maintained for the global and local tasks to accommodate objective-specific dynamics with minimal computational overhead. The predictor g_ψ^pred is not used to refine the CLS token, meaning it is exclusive to the MPM branch. *multi-mask* is applied to the two global crops, whereas the local crops remain unmasked. These masked global views create additional cross-view comparisons for the global objective, and can be seen as a form of implicit denoising, similar to MSN [19]. Figure 6 summarizes the interactions between the outputs in the complete framework.

4 Experiments

We evaluate AsymDSD through extensive experiments across 3D recognition, few-shot classification, and part segmentation, including studies on scalability and ablations.

4.1 ShapeNet Pre-Training

Implementation Details. We follow the common SSRL pre-training protocol involving ShapeNet-Core [36], consisting of 41,952 CAD models across 55 categories. Point clouds are generated by

Table 1: Overall accuracy on **ModelNet40** and **ScanObjectNN**. Where available, the accuracy without voting is reported. `ST` indicates a ViT-S sized standard transformer as described in Section A and Table 7c; `SM` indicates single-modal training; **#P(M)** indicates the number of parameters in millions.

Method	Reference	#P(M)	ST	SM	ModelNet40	ScanObjectNN		
						OBJ_BG	OBJ_ONLY	PB_T50_RS
Supervised Learning Only								
PointNet	CVPR '17 [62]	3.5	×	✓	89.2	73.3	79.2	68.0
PointMLP	ICLR '22 [63]	12.6	×	✓	**94.1**	-	-	85.4±.3
PointNeXt	NeurIPS '22 [64]	1.4	×	✓	94.0	-	-	**87.7±.4**
Adapted ViT-S	Appendix A	22.1	✓	✓	92.9	87.6	86.7	83.5
Self-Supervised Representation Learning - Full Fine-tune								
Point-BERT	CVPR '22 [21]	22.1	✓	✓	93.2	87.43	88.12	83.07
Point-MAE	ECCV '22 [22]	22.1	✓	✓	93.2	90.02	88.29	85.18
PointGPT-S	NeurIPS '23 [24]	22.1	✓	✓	94.0	91.6	90.0	86.9
AsymSD-CLS-S	Section 3.1	22.1	✓	✓	93.6	**92.77**	**90.53**	**88.72**
AsymSD-MPM-S	Section 3.2	22.1	✓	✓	94.0	**92.77**	**91.39**	**88.58**
AsymDSD-S	Section 3.3	22.1	✓	✓	**94.1**	**94.32**	**91.91**	**90.53**
MaskPoint	ECCV '22 [65]	-	×	✓	93.8	89.3	88.1	84.3
Point-M2AE	NeurIPS '22 [23]	15.3	×	✓	94.0	91.22	88.81	86.43
ReCon SM	ICML '23 [26]	43.6	×	✓	93.6	94.15	93.12	89.73
Point-RAE	ACMMM '23 [66]	29.2	×	✓	94.0	**95.53**	**93.63**	90.28
Point-FEMAE	AAAI '24 [25]	27.4	×	✓	94.0	95.18	93.29	90.22
PointMamba	Neurips '24 [67]	12.3	×	✓	93.6	94.32	92.60	89.31
Self-Supervised Representation Learning - Linear								
Point-BERT	CVPR '22 [21]	22.1	✓	✓	91.09±.15	84.17±.30	87.19±.16	74.44±.12
Recon MAE	ICML '23 [22]	43.3	×	✓	90.22±.09	82.77±.30	83.23±.16	74.13±.21
point2vec	GCPR '23 [34]	22.1	✓	✓	92.44±.04	82.75±.54	85.44±.26	74.25±.11
Point-RAE	ACMMM '23 [66]	28.9	×	✓	-	86.15±.33	86.31±.23	78.25±.30
AsymSD-CLS-S	Section 3.1	21.8	✓	✓	91.78±.10	**90.67±.31**	**88.31±.31**	**83.19±.14**
AsymSD-MPM-S	Section 3.2	21.8	✓	✓	**93.55±.05**	**89.00±.20**	**87.80±.14**	**81.04±.14**
AsymDSD-S	Section 3.3	21.8	✓	✓	**92.52±.15**	**89.95±.21**	**88.73±.23**	**83.33±.14**
ACT	ICLR '23 [2]	21.8	✓	×	91.36±.17	85.20±.83	85.84±.15	76.31±.26
ReCon	ICML '23 [22]	43.3	×	×	92.47±.22	89.50±.20	**89.72±.17**	81.36±.14

uniformly sampling 16,384 surface points per mesh and normalizing them to the unit sphere. For each input, we sample two global crops (1,024 points, 64 patches) and four local crops (256 points, 16 patches), with each patch comprising the $K = 32$ nearest neighboring points. Global crops are masked using inverse block-wise masking at a 70% ratio with four masks per crop. The encoder is a ViT-S backbone with RMSNorm and GELU, and the student predictor is a lighter ViT-Ti [61] variant with 6 layers. The projection heads expand embeddings to a 4096-way latent space. Pre-training is run for 300 epochs using AdamW, with a cosine learning rate schedule peaking at 5.0×10^{-4}, and a cosine EMA decay increasing from 0.995 to 1.0 during training. With a single RTX 4090, this takes roughly 18 hours to complete. For additional details, we refer the reader to Appendix B.1.

Object Recognition. We evaluate the downstream performance of AsymDSD on standard 3D object classification benchmarks using three protocols: *From Scratch*, *Linear*, and *Full Fine-tune* [2]. The teacher encoder (i.e., without projection head) is used for all downstream tasks, as it generally outperforms the student. In the *Linear* protocol, we freeze the encoder's weights and add a trainable linear layer atop the encoder. For the *Full Fine-tune* and *From Scratch* settings, we employ a 3-layer MLP head and update all model parameters during training. The inputs to these classification heads are formed by concatenating the CLS-token embedding with the mean and max pooled patch embeddings. All models are trained for 150 epochs using cross-entropy loss with label smoothing set to 0.2, a batch size 32, and a drop path rate of 0.2 [68]. The MLP head uses hidden dimensions of 256, batch normalization, and dropout ($p = 0.5$). Results to these experiments are shown in Table 1.

On **ModelNet40** [38], a clean synthetic dataset of 12k CAD models across 40 categories, **AsymDSD-S** achieves 94.1% accuracy with *Full Fine-tune*, a +1.2% increase compared to training *From Scratch* (**Adapted ViT-S**). Even with a simple linear probe, performance remains strong, indicating strong off-the-shelf representations. In contrast, point-reconstruction methods like MAE underperform by over 2% in the *Linear* setup. On **ScanObjectNN** [37], a real-world scanned object dataset, AsymDSD achieves 90.53% (+7.0%) on the hardest PB_T50_RS split, surpassing all prior methods with a standard transformer by +3.6%. While some methods slightly outperform AsymDSD on

cleaner subsets, they rely on architectural modifications with additional trainable parameters during fine-tuning. In fact, with equal trainable parameters under *Linear* probing, our method outperforms all other self-supervised approaches, including cross-modal methods like ACT [2] and ReCon [26].

Few-Shot Classification. We evaluate few-shot performance on ModelNet40 following the m-way, n-shot protocol of [69]. The model is fine-tuned with a high learning rate (1×10^{-3}) and predictions are based on concatenated mean and max pooled patch embeddings (ignoring the CLS token). As shown in Table 2, **AsymDSD-S** achieves the best result in three out of four configurations, further supporting the off-the-shelve quality of the learned representations in low-data regimes.

Part Segmentation. We also evaluate semantic segmentation on ShapeNet-Part [70]. For this, we employ a PointNet++-style decoder [71] aggregating features from multiple encoder layers, following Point-MAE [22]. As shown in Table 3, **AsymDSD-S** achieves competitive mIoU scores, similar to transformer-based methods like Point-MAE [22] and PointGPT-S [24]. While methods such as point-M2AE outperform in this task, they rely on a U-Net-style architecture, which is generally better suited for dense prediction.

Table 2: Average overall accuracy and standard deviation on **ModelNet40 few-shot** over 10 independent runs per experiment.

Method	ST	5-way		10-way	
		10-shot	20-shot	10-shot	20-shot
Point-BERT [21]	✓	94.6±3.1	96.3±2.7	91.0±5.4	92.7±5.1
MaskPoint [65]	×	95.0±3.7	97.2±1.7	91.4±4.0	93.4±3.5
Point-MAE [22]	✓	96.3±2.5	97.8±1.8	92.6±4.1	95.0±3.0
Point-RAE [34]	×	**97.3±1.6**	98.7±1.3	93.3±4.0	**95.8±3.0**
Point-FEMAE [34]	×	97.2±1.9	98.6±1.3	94.0±3.3	**95.8±2.8**
AsymDSD-S	✓	96.1±2.8	**98.8±1.3**	**94.4±3.5**	**95.8±3.3**

Table 3: Part segmentation results on **ShapeNet-Part**. The mean IoU is over all instances (Inst.) or per-class (Cls.).

Method	ST	mIoU	
		Inst.	Cls.
Point-BERT [21]	✓	85.6	84.1
Point-MAE [22]	✓	**86.1**	**84.4**
PointGPT-S [24]	✓	86.2	84.1
AsymDSD-S	✓	86.0	**84.4**
point-FEMAE [25]	×	86.3	**84.9**
point-M2AE [23]	×	**86.5**	**84.9**

4.2 Scaling Beyond ShapeNet

While ShapeNet is a standard benchmark for 3D SSRL, it is small by today's standards and only comprises synthetic data, constraining the quality of learned representations. To explore the scalability of AsymDSD, we pre-train on a substantially larger and more diverse dataset composed of synthetic and real-world 3D sources. This *Mixture* dataset combines a 10-dataset aggregate (133K instances) with Objaverse [4] (797K instances). In addition, we also explore a larger ViT-B backbone (**AsymDSD-B***), extending beyond the original ViT-S configuration (**AsymDSD-S***). Details of the dataset composition and training setup are provided in Appendix C.1.

Experiments show strong improvements from scaling (Table 4). When moving from ShapeNet to the more extensive *Mixture* dataset, fine-tuning accuracy improves by +0.6% on ModelNet40 and +3.19% on ScanObjectNN, surpassing the previous best PointGPT-L. Linear evaluation results show even greater gains, with improvements of +1.24% and +8.63%, respectively. We further compare with models that have undergone extensive post-pretraining on supervised data (PP) [24], including

Table 4: Results from scaling up pre-training. PP indicates additional post-pretraining with supervised data; CM cross-modal training; **#P(M)** the parameters count in millions.

Method	**Reference**	**#P(M)**	PP	CM	**ModelNet40**	**ScanObjectNN**		
						OBJ_BG	OBJ_ONLY	PB_T50_RS
Full Fine-tune								
PointGPT-B	NeurIPS '23 [24]	92.0	×	×	94.2	93.6	92.5	89.6
PointGPT-L	NeurIPS '23 [24]	310.0	×	×	94.5	95.7	94.1	91.1
AsymDSD-S*	-	22.1	×	×	94.4	**97.07**	**94.83**	92.89
AsymDSD-B*	-	92.1	×	×	**94.7**	96.73	94.32	**93.72**
Point-MAE-B⋆	NeurIPS '23 [24]	120.1	✓	×	94.2	94.2	93.9	90.2
PointGPT-B⋆	NeurIPS '23 [24]	92.0	✓	×	94.4	95.8	95.2	91.9
ReCon++-B⋆	ECCV '24 [72]	177.4	✓	✓	94.6	**98.62**	**96.21**	93.34
Linear								
AsymDSD-S*	-	21.8	×	×	**93.98±.14**	95.22±.17	**94.51±.18**	91.31±.09
AsymDSD-B*	-	91.8	×	×	93.76±.13	**96.16±.11**	93.44±.16	**91.96±.14**

Table 5: Ablations. Accuracy on ScanObjectNN is reported using a linear SVM or *Full Fine-tune*.

(a) Cropping and *multi-crop*.

Method	SVM	FFt
CLS-S	**82.27**	**88.72**
– cropping	73.60 ↓ **8.67**	82.13 ↓ **6.59**
– *multi-crop*	78.38 ↓ **3.89**	85.64 ↓ **3.08**

(b) Mask strategies.

Method	Ratio	SVM	FFt
Inverse bw.	0.6	79.63 ↓ 1.43	88.02 ↓ 0.46
	0.7	80.95 ↓ 0.11	88.20 ↓ 0.38
MPM-S	0.8	81.06	**88.58**
	0.85	**81.37** ↑ 0.29	88.17 ↓ 0.41
Uniform	0.8	79.74 ↓ 1.32	87.99 ↓ 0.59

(c) Predictor. `c`: visible context; `m`: mask tokens

Method	Attention		Mem.	It/s	SVM	FFt
	Self	Cross				
MPM-S	×	`c`	16.2	4.80	**81.06**	**88.58**
	`c+m`	×	17.9	4.32	77.00 ↓ **4.06**	85.98 ↓ **2.60**
– predictor	×	×	23.3	3.11	24.95 ↓ **54.51**	69.74 ↓ **18.84**

(d) *multi-mask*. **Bs.**: Batch size; **Mm.**: Number of masks

Method	Bs.	Mm.	Mem.	It/s	SVM	FFt
MPM-S	128	8	16.2	4.80	**81.06**	**88.58**
– *multi-mask*	1024	1	48.8	1.46	81.16 ↑ 0.10	88.48 ↓ 0.10

the significantly larger cross-modal model ReCon++-B [26]. Remarkably, **AsymDSD-B***, despite its smaller size and purely self-supervised training, outperforms these models on both ModelNet40 and the hardest ScanObjectNN benchmark. These results underscore the effectiveness of AsymDSD in leveraging large-scale unlabeled 3D data to learn highly transferable representations.

4.3 Properties and Ablations

Synergistic Objectives. AsymDSD jointly optimizes two complementary objectives to enhance representation learning. To assess the synergistic effect, we train models on each objective separately with re-tuned hyperparameters. As shown in Table 1, both invariance learning (**AsymSD-CLS-S**) and masked point modeling (**AsymSD-MPM-S**) demonstrate competitive down-stream performance, but their combination (**AsymDSD-S**) exceeds both objectives across all benchmarks after fine-tuning. For linear probing, however, individual objectives may outperform due to task-specific inductive biases favoring certain benchmarks. To further understand this synergy, we examine the attention distance patterns (Figure 7) and observe that our latent MPM demonstrates an attention specialization pattern relatively aligned with the CLS objective. This contrasts with the typical inverted pattern found in masked autoencoders [26]. We hypothesize that this alignment enhances their composability.

Ablations. We highlight several ablations to demonstrate that the main contributions are not some simple add-ons for minor incremental gains, but are core to their objectives. As shown in Table 5a, cropping plays a key role in learning effective representations, with local crops (*multi-crop*) enabling full performance. We also evaluate different masking strategies (Table 5b), finding that our proposed inverse block-wise masking outperforms uniform masking. The method is robust to exact configurations, though best results are observed with masking ratios of 0.8 for MPM and 0.7 for the dual objective. Furthermore, Table 5c shows that the predictor is necessary to overcome collapse. That said, we observe that combining MPM with the global objective is sufficient to stabilizes training. Still, performance is reduced if global shape leakage is not addressed at the predictor (Appendix E). This is further evidenced by the performance gap between our efficient cross-attention-only predictor and the more expressive design with self-attention over all patches including masks (`c+m`). When we disable *multi-mask*, the efficiency of the design becomes especially obvious (Table 5d). There is virtually no performance degradation when using *multi-mask* with similar effective batch size, while reducing the memory usage and throughput by nearly 70%.

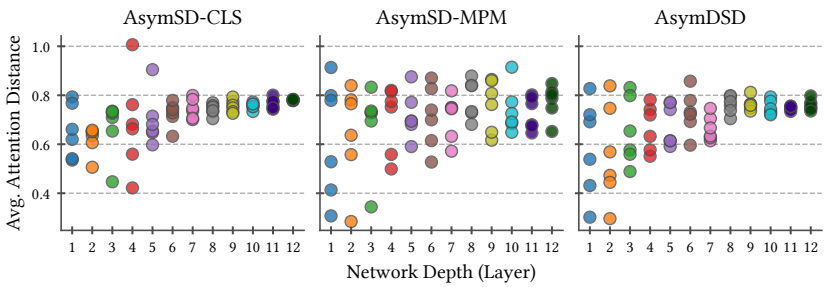


Figure 7: The average attention distance per attention head across the depth of the encoder.

Table 6: Comparison of prediction targets.

Method	Raw points	Latent	ModelNet40		ScanObjectNN	
			SVM	FFt	SVM	FFt
CLS	-	-	91.78	93.64	82.27	88.72
MAE	✓	×	92.91	**94.04**	78.49	87.75
MPM	×	✓	**93.31**	**94.04**	**81.06**	**88.58**
CLS + MAE	✓	×	91.90 ↑ 0.12	93.48 ↓ 0.16	**82.93** ↑ 0.66	88.55 ↓ 0.17
CLS + MPM	×	✓	**93.03** ↑ 1.25	**94.13** ↑ 0.49	82.34 ↑ 0.07	**90.53** ↑ 1.81

Latent targets. Table 6 highlights the impact of shifting the masked modeling targets from the input space (raw points) to the latent space. Our implementation of MAE shows substantial gains over Point-MAE [22], which can be attributed to mitigating global shape leakage through our predictor design and to the scale invariance introduced by variable-sized global crops. This reveals that some of our contributions extend beyond our framework. However, more importantly, as hypothesized, combining global invariance learning (CLS) with MAE (CLS + MAE) provides limited additional benefit and even results in a slight performance drop under full fine-tuning. In contrast, integrating CLS with our latent MPM objective (CLS + MPM) produces a clear synergistic effect, where the model to surpass both objectives when applied independently.

5 Conclusion and Limitations

Conclusion. In this paper, we introduced AsymDSD, a unified self-supervised learning framework for 3D point clouds that integrates masked modeling and invariance learning through asymmetric dual self-distillation. By avoiding reconstruction-based targets and addressing critical issues like shape leakage and representation collapse, AsymDSD enables efficient and semantically rich representation learning. The asymmetric design not only stabilizes training but also improves computational efficiency by decoupling heavy encoding from lightweight prediction. Extensive experiments demonstrate SOTA performance on multiple benchmarks, with strong generalization in low-shot and large-scale settings.

Limitations. While AsymDSD demonstrates strong performance on a wide-range of experiments, there are some limitations to the current work: (1) Pre-training and evaluations have been primarily conducted on object-centered datasets; and (2) the encoder architecture uses a flat hierarchy. These limitations are closely related, as the current architecture does not scale well to larger point clouds typically found in scene-level data. However, AsymDSD can be extended to hierarchical encoders with multi-resolution representations, which are better suited for such settings. Building on the promising results of this study, we view this as a compelling direction for future work.

Acknowledgments

This research was conducted as part of the first author's master's thesis at the University of Groningen, in collaboration with Clear Timber Analytics. The authors would like to thank Hamidreza Kasaei for academic guidance and Alex van Gelder at Clear Timber Analytics for their support and provision of resources that contributed to this work. We also thank the Center for Information Technology of the University of Groningen for providing access to the Hábrók high performance computing cluster.

References

[1] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[2] 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? *arXiv preprint arXiv:2212.08320*, 2022.

[3] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5828–5839, 2017.

[4] Matt Deitke, Dustin Schwenk, Jordi Salvador, Luca Weihs, Oscar Michel, Eli VanderBilt, Ludwig Schmidt, Kiana Ehsani, Aniruddha Kembhavi, and Ali Farhadi. Objaverse: A universe of annotated 3d objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13142–13153, 2023.

[5] Matt Deitke, Ruoshi Liu, Matthew Wallingford, Huong Ngo, Oscar Michel, Aditya Kusupati, Alan Fan, Christian Laforte, Vikram Voleti, Samir Yitzhak Gadre, et al. Objaverse-xl: A universe of 10m+ 3d objects. *Advances in Neural Information Processing Systems*, 36, 2024.

[6] Ben Fei, Weidong Yang, Liwen Liu, Tianyue Luo, Rui Zhang, Yixuan Li, and Ying He. Self-supervised learning for pre-training 3d point clouds: A survey. *arXiv preprint arXiv:2305.04691*, 2023.

[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[8] Mark Chen, Alec Radford, Rewon Child, Jeffrey Wu, Heewoo Jun, David Luan, and Ilya Sutskever. Generative pretraining from pixels. In *International conference on machine learning*, pages 1691–1703. PMLR, 2020.

[9] 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. *Advances in neural information processing systems*, 33:21271–21284, 2020.

[10] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.

[11] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9650–9660, 2021.

[12] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16000–16009, 2022.

[13] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

[14] Priya Goyal, Mathilde Caron, Benjamin Lefaudeux, Min Xu, Pengchao Wang, Vivek Pai, Mannat Singh, Vitaliy Liptchinsky, Ishan Misra, Armand Joulin, et al. Self-supervised pretraining of visual features in the wild. *arXiv preprint arXiv:2103.01988*, 2021.

[15] Mannat Singh, Quentin Duval, Kalyan Vasudev Alwala, Haoqi Fan, Vaibhav Aggarwal, Aaron Adcock, Armand Joulin, Piotr Dollár, Christoph Feichtenhofer, Ross Girshick, et al. The effectiveness of mae pre-pretraining for billion-scale pretraining. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5484–5494, 2023.

[16] David Fan, Shengbang Tong, Jiachen Zhu, Koustuv Sinha, Zhuang Liu, Xinlei Chen, Michael Rabbat, Nicolas Ballas, Yann LeCun, Amir Bar, et al. Scaling language-free visual representation learning. *arXiv preprint arXiv:2504.01017*, 2025.

[17] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[18] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

[19] Mahmoud Assran, Mathilde Caron, Ishan Misra, Piotr Bojanowski, Florian Bordes, Pascal Vincent, Armand Joulin, Mike Rabbat, and Nicolas Ballas. Masked siamese networks for label-efficient learning. In *European Conference on Computer Vision*, pages 456–473. Springer, 2022.

[20] Hanchen Wang, Qi Liu, Xiangyu Yue, Joan Lasenby, and Matt J Kusner. Unsupervised point cloud pre-training via occlusion completion. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9782–9792, 2021.

[21] 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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19313–19322, 2022.

[22] 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*, pages 604–621. Springer, 2022.

[23] 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. *Advances in neural information processing systems*, 35:27061–27074, 2022.

[24] Guangyan Chen, Meiling Wang, Yi Yang, Kai Yu, Li Yuan, and Yufeng Yue. Pointgpt: Auto-regressively generative pre-training from point clouds. *arXiv preprint arXiv:2305.11487*, 2023.

[25] Yaohua Zha, Huizhen Ji, Jinmin Li, Rongsheng Li, Tao Dai, Bin Chen, Zhi Wang, and Shu-Tao Xia. Towards compact 3d representations via point feature enhancement masked autoencoders. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 6962–6970, 2024.

[26] Zekun Qi, Runpei Dong, Guofan Fan, Zheng Ge, Xiangyu Zhang, Kaisheng Ma, and Li Yi. Contrast with reconstruct: Contrastive 3d representation learning guided by generative pretraining. *arXiv preprint arXiv:2302.02318*, 2023.

[27] Tim Salimans, Andrej Karpathy, Xi Chen, and Diederik P Kingma. Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications. *arXiv preprint arXiv:1701.05517*, 2017.

[28] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In *International conference on machine learning*, pages 8821–8831. Pmlr, 2021.

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

[30] 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*, pages 1298–1312. PMLR, 2022.

[31] Alexei Baevski, Arun Babu, Wei-Ning Hsu, and Michael Auli. Efficient self-supervised learning with contextualized target representations for vision, speech and language. In *International Conference on Machine Learning*, pages 1416–1429. PMLR, 2023.

[32] Mahmoud Assran, Quentin Duval, Ishan Misra, Piotr Bojanowski, Pascal Vincent, Michael Rabbat, Yann LeCun, and Nicolas Ballas. Self-supervised learning from images with a joint-embedding predictive architecture. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15619–15629, 2023.

[33] Jinghao Zhou, Chen Wei, Huiyu Wang, Wei Shen, Cihang Xie, Alan Yuille, and Tao Kong. ibot: Image bert pre-training with online tokenizer. *arXiv preprint arXiv:2111.07832*, 2021.

[34] Karim Abou Zeid, Jonas Schult, Alexander Hermans, and Bastian Leibe. Point2vec for self-supervised representation learning on point clouds. *arXiv preprint arXiv:2303.16570*, 2023.

[35] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. *Advances in neural information processing systems*, 33:9912–9924, 2020.

[36] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015.

[37] Mikaela Angelina Uy, Quang-Hieu Pham, Binh-Son Hua, Thanh Nguyen, and Sai-Kit Yeung. Revisiting point cloud classification: A new benchmark dataset and classification model on real-world data. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 1588–1597, 2019.

[38] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1912–1920, 2015.

[39] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. arxiv e-prints, art. *arXiv preprint arXiv:1911.05722*, 2019.

[40] Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 15750–15758, 2021.

[41] Mahmoud Assran, Mathilde Caron, Ishan Misra, Piotr Bojanowski, Armand Joulin, Nicolas Ballas, and Michael Rabbat. Semi-supervised learning of visual features by non-parametrically predicting view assignments with support samples. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8443–8452, 2021.

[42] Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, and Stéphane Deny. Barlow twins: Self-supervised learning via redundancy reduction. In *International conference on machine learning*, pages 12310–12320. PMLR, 2021.

[43] Adrien Bardes, Jean Ponce, and Yann LeCun. Vicreg: Variance-invariance-covariance regularization for self-supervised learning. *arXiv preprint arXiv:2105.04906*, 2021.

[44] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103, 2008.

[45] Zhenda Xie, Zheng Zhang, Yue Cao, Yutong Lin, Jianmin Bao, Zhuliang Yao, Qi Dai, and Han Hu. Simmim: A simple framework for masked image modeling. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9653–9663, 2022.

[46] Hangbo Bao, Li Dong, Songhao Piao, and Furu Wei. Beit: Bert pre-training of image transformers. *arXiv preprint arXiv:2106.08254*, 2021.

[47] Jason Tyler Rolfe. Discrete variational autoencoders. *arXiv preprint arXiv:1609.02200*, 2016.

[48] Saining Xie, Jiatao Gu, Demi Guo, Charles R Qi, Leonidas Guibas, and Or Litany. Pointcontrast: Unsupervised pre-training for 3d point cloud understanding. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part III 16*, pages 574–591. Springer, 2020.

[49] Zaiwei Zhang, Rohit Girdhar, Armand Joulin, and Ishan Misra. Self-supervised pretraining of 3d features on any point-cloud. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10252–10263, 2021.

[50] Mohamed Afham, Isuru Dissanayake, Dinithi Dissanayake, Amaya Dharmasiri, Kanchana Thilakarathna, and Ranga Rodrigo. Crosspoint: Self-supervised cross-modal contrastive learning for 3d point cloud understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9902–9912, 2022.

[51] Siyuan Huang, Yichen Xie, Song-Chun Zhu, and Yixin Zhu. Spatio-temporal self-supervised representation learning for 3d point clouds. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6535–6545, 2021.

[52] Adriano Cardace, Riccardo Spezialetti, Pierluigi Zama Ramirez, Samuele Salti, and Luigi Di Stefano. Self-distillation for unsupervised 3d domain adaptation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 4166–4177, 2023.

[53] 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. *arXiv preprint arXiv:2010.11929*, 2020.

[54] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[55] Michael Tschannen, Josip Djolonga, Paul K Rubenstein, Sylvain Gelly, and Mario Lucic. On mutual information maximization for representation learning. *arXiv preprint arXiv:1907.13625*, 2019.

[56] Ralph Linsker. Self-organization in a perceptual network. *Computer*, 21(3):105–117, 1988.

[57] Idan Achituve, Haggai Maron, and Gal Chechik. Self-supervised learning for domain adaptation on point clouds. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pages 123–133, 2021.

[58] Alexandre Sablayrolles, Matthijs Douze, Cordelia Schmid, and Hervé Jégou. Spreading vectors for similarity search. *arXiv preprint arXiv:1806.03198*, 2018.

[59] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[60] Yuandong Tian, Xinlei Chen, and Surya Ganguli. Understanding self-supervised learning dynamics without contrastive pairs. In *International Conference on Machine Learning*, pages 10268–10278. PMLR, 2021.

[61] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *International conference on machine learning*, pages 10347–10357. PMLR, 2021.

[62] CR Qi, H Su, K Mo, and LJ Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. cvpr 2017. *arXiv preprint arXiv:1612.00593*, 2016.

[63] Xu Ma, Can Qin, Haoxuan You, Haoxi Ran, and Yun Fu. Rethinking network design and local geometry in point cloud: A simple residual mlp framework. *arXiv preprint arXiv:2202.07123*, 2022.

[64] Guocheng Qian, Yuchen Li, Houwen Peng, Jinjie Mai, Hasan Hammoud, Mohamed Elhoseiny, and Bernard Ghanem. Pointnext: Revisiting pointnet++ with improved training and scaling strategies. *Advances in Neural Information Processing Systems*, 35:23192–23204, 2022.

[65] Haotian Liu, Mu Cai, and Yong Jae Lee. Masked discrimination for self-supervised learning on point clouds. In *European Conference on Computer Vision*, pages 657–675. Springer, 2022.

[66] Yang Liu, Chen Chen, Can Wang, Xulin King, and Mengyuan Liu. Regress before construct: Regress autoencoder for point cloud self-supervised learning. In *Proceedings of the 31st ACM International Conference on Multimedia*, pages 1738–1749, 2023.

[67] Dingkang Liang, Xin Zhou, Wei Xu, Xingkui Zhu, Zhikang Zou, Xiaoqing Ye, Xiao Tan, and Xiang Bai. Pointmamba: A simple state space model for point cloud analysis. *arXiv preprint arXiv:2402.10739*, 2024.

[68] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q Weinberger. Deep networks with stochastic depth. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part IV 14*, pages 646–661. Springer, 2016.

[69] Charu Sharma and Manohar Kaul. Self-supervised few-shot learning on point clouds. *Advances in Neural Information Processing Systems*, 33:7212–7221, 2020.

[70] 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 Transactions on Graphics (ToG)*, 35(6):1–12, 2016.

[71] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems*, 30, 2017.

[72] Zekun Qi, Runpei Dong, Shaochen Zhang, Haoran Geng, Chunrui Han, Zheng Ge, Li Yi, and Kaisheng Ma. Shapellm: Universal 3d object understanding for embodied interaction. In *European Conference on Computer Vision*, pages 214–238. Springer, 2024.

[73] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[74] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[75] Yu-Qi Yang, Yu-Xiao Guo, Jian-Yu Xiong, Yang Liu, Hao Pan, Peng-Shuai Wang, Xin Tong, and Baining Guo. Swin3d: A pretrained transformer backbone for 3d indoor scene understanding. *arXiv preprint arXiv:2304.06906*, 2023.

[76] Peng-Shuai Wang. Octformer: Octree-based transformers for 3d point clouds. *ACM Transactions on Graphics (TOG)*, 42(4):1–11, 2023.

[77] Xiaoyang Wu, Li Jiang, Peng-Shuai Wang, Zhijian Liu, Xihui Liu, Yu Qiao, Wanli Ouyang, Tong He, and Hengshuang Zhao. Point transformer v3: Simpler, faster, stronger. *arXiv preprint arXiv:2312.10035*, 2023.

[78] Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, et al. Mixed precision training. *arXiv preprint arXiv:1710.03740*, 2017.

[79] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[80] Biao Zhang and Rico Sennrich. Root mean square layer normalization. *Advances in Neural Information Processing Systems*, 32, 2019.

[81] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

[82] Iro Armeni, Ozan Sener, Amir R Zamir, Helen Jiang, Ioannis Brilakis, Martin Fischer, and Silvio Savarese. 3d semantic parsing of large-scale indoor spaces. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1534–1543, 2016.

[83] Shuran Song, Samuel P Lichtenberg, and Jianxiong Xiao. Sun rgb-d: A rgb-d scene understanding benchmark suite. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 567–576, 2015.

[84] Agrim Gupta, Piotr Dollar, and Ross Girshick. Lvis: A dataset for large vocabulary instance segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5356–5364, 2019.

[85] Minghua Liu, Ruoxi Shi, Kaiming Kuang, Yinhao Zhu, Xuanlin Li, Shizhong Han, Hong Cai, Fatih Porikli, and Hao Su. Openshape: Scaling up 3d shape representation towards open-world understanding. *Advances in neural information processing systems*, 36:44860–44879, 2023.

[86] Junsheng Zhou, Jinsheng Wang, Baorui Ma, Yu-Shen Liu, Tiejun Huang, and Xinlong Wang. Uni3d: Exploring unified 3d representation at scale. *arXiv preprint arXiv:2310.06773*, 2023.

[87] Huan Fu, Rongfei Jia, Lin Gao, Mingming Gong, Binqiang Zhao, Steve Maybank, and Dacheng Tao. 3d-future: 3d furniture shape with texture. *International Journal of Computer Vision*, 129:3313–3337, 2021.

[88] Jasmine Collins, Shubham Goel, Kenan Deng, Achleshwar Luthra, Leon Xu, Erhan Gundogdu, Xi Zhang, Tomas F Yago Vicente, Thomas Dideriksen, Himanshu Arora, et al. Abo: Dataset and benchmarks for real-world 3d object understanding. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 21126–21136, 2022.

[89] Tong Wu, Jiarui Zhang, Xiao Fu, Yuxin Wang, Jiawei Ren, Liang Pan, Wayne Wu, Lei Yang, Jiaqi Wang, Chen Qian, et al. Omniobject3d: Large-vocabulary 3d object dataset for realistic perception, reconstruction and generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 803–814, 2023.

[90] Stefan Stojanov, Anh Thai, and James M Rehg. Using shape to categorize: Low-shot learning with an explicit shape bias. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1798–1808, 2021.

[91] Anthony G Francis, Brandon Kinman, Krista Ann Reymann, Laura Downs, Nathan Koenig, Ryan M Hickman, Thomas B McHugh, and Vincent Olivier Vanhoucke. Google scanned objects: A high-quality dataset of 3d scanned household items. 2022.

[92] Antti Tarvainen and Harri Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. *Advances in neural information processing systems*, 30, 2017.

[93] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The claims are substantiated in the Sec. 3 and Sec. 4.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: There is a dedication section with limitations in Sec. 5.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

Justification: The paper does not include theoretical results.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The core methodology can be reproduced based on Sec. 3. The method is generally robust against exact hyperparameters, but we provide many details in the Appendix B.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: Data is publicly available. We include code with configurations to reproduce results.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: The general experimental settings are presented in Sec. 4. Appendix B contains tables with many precise details.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [No]

Justification: We do not test for statistical significance, but do include standard deviations on some results.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We disclose the computer resources in Sec. 4. More details are in Appendix C.2.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The conducted research is conform the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

    Justification: There is no direct societal impact, as we propose a novel learning framework for 3D point clouds.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.

- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: There is no such immediate risk.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We cite all original works including datasets. Our work does not redistribute existing assets.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: We do not release new assets.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: Our work does not involve crowdsourcing nor researching with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: Our work does not involve crowdsourcing nor researching with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [NA]

    Justification: The core method development does not involve LLMs.

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.

A Overall Model Pipeline

Although many dedicated models architectures for point cloud data have been devised, there is a lack of a unified architecture akin to those established in NLP and 2D CV. In these areas, model architectures have converged over time to a few dominant designs [73, 74, 53]. However, in recent times, a line of work on 3D SSRL has emerged [22, 21, 34, 24] that relies on a relatively simple model design that integrates the standard transformer architecture [74, 53]. This model architecture is detailed in this section. A detailed overview of this pipeline with AsymDSD pre-training is shown in Figure 8.

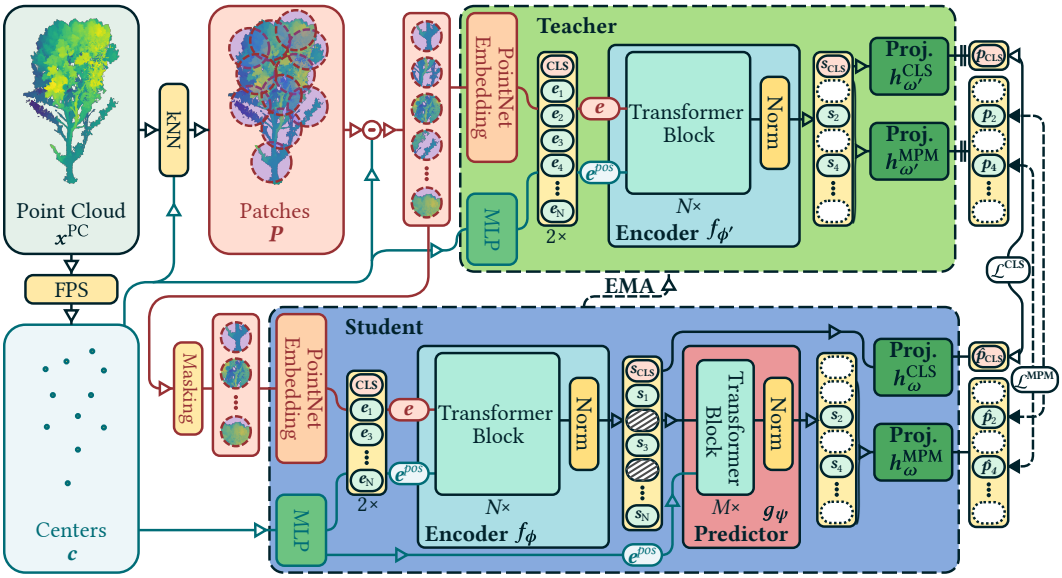


Figure 8: Overview of the processing pipeline for AsymDSD for a single point cloud through both the *Teacher* and *Student* network. The red colored arrows and modules indicate the stream of local structural information. The blue colors indicate the stream of global positional information. These streams get mixed at the encoder and predictor networks.

A.1 Patch Tokenization

The first step of the processing pipeline involves the abstraction of the point set to a smaller set of point patches to obtain a manageable set of units for further processing. These patches are local groups of points that are comparable to image patches in vision transformers [53]. These patches are subsequently embedded to obtain a set of patch tokens.

A.1.1 Input

The input is a point cloud $\mathcal{S}$, consisting of a finite collection of pairs of point positions $p^{(i)} \in \mathcal{P} \subset \mathbb{R}^3$ and F point features $f^{(i)} \in \mathcal{F} \subset \mathbb{R}^F$:

$$\mathcal{S} = \left\{ \left(p^{(i)}, f^{(i)} \right) \mid p^{(i)} \in \mathcal{P}, f^{(i)} \in \mathcal{F}, i = 1, \ldots, n \right\}. \tag{8}$$

However, for simplicity we consider the point cloud $\mathcal{S}$ in tensor representation $\boldsymbol{x}^{\mathrm{PC}}$:

$$\boldsymbol{x}^{\mathrm{PC}} = [\boldsymbol{p} \mid \boldsymbol{f}] \in \mathbb{R}^{N \times (3+F)}, \tag{9}$$

with point positions $\boldsymbol{p} \in \mathbb{R}^{N \times 3}$ and accompanying features $\boldsymbol{f} \in \mathbb{R}^{N \times F}$.

A.1.2 Patchify

To form the patches, first, N_c center points $\boldsymbol{c}$ are sampled via farthest point sampling (FPS). FPS is a procedure that iteratively samples points that are most distant from the already sampled points,

starting with a randomly sampled point. Subsequently, k-nearest neighbour (KNN) is employed to find the K-nearest points in $\boldsymbol{x}^{\mathrm{PC}}$ based on their corresponding positions $\boldsymbol{p}$ from each center $\boldsymbol{c}$. This procedure is referred to as *patchify* and can be expressed mathematically as:

$$\boldsymbol{c} = \mathrm{FPS}(\boldsymbol{p}), \qquad \boldsymbol{c} \in \mathbb{R}^{N_c \times 3}; \tag{10}$$

$$\boldsymbol{P} = \mathrm{KNN}\left(\boldsymbol{c}, \boldsymbol{p}, \boldsymbol{x}^{\mathrm{PC}}; K\right), \qquad \boldsymbol{P} \in \mathbb{R}^{N_c \times K \times (3+F)}. \tag{11}$$

To disentangle the global positional information from the local structural information, the reference frame of each patch is translated to its respective center:

$$\boldsymbol{X}^{\boldsymbol{P}} = [\boldsymbol{P}_{xyz} - \boldsymbol{c} | \boldsymbol{P}_f], \qquad \boldsymbol{X}^{\boldsymbol{P}} \in \mathbb{R}^{N_c \times K \times (3+F)}; \tag{12}$$

with $\boldsymbol{P}_{xyz}$ the point positions, and $\boldsymbol{P}_f$ the point features of the patches.

This *patchify* procedure is reflected as a part of the complete processing pipeline for AsymDSD in Figure 8.

A.1.3 Patch Embedding

The obtained patches $\boldsymbol{X}^{\boldsymbol{P}}$ are themselves small point clouds. Accordingly, before further processing, they must be projected to an embedding space. This embedding process effectively boils down to the compression of the point cloud into a single feature vector describing its shape. For this purpose a small *PointNet Embedding* model is used, as shown in Figure 9a.

This embedding model first projects the points in each patch to a higher D^1_{patch}-dimensional space through a simple shared multi-layer perceptron (MLP), and takes the maximum of this projection over the K points. This essentially partitions the Euclidean space into regions. To make the partitioning dependent on the contents of the point cloud, the maximum feature vector is concatenated to the projected point features before being projected and pooled once more to obtain the final patch embeddings $\boldsymbol{e}^{\boldsymbol{P}}$. In other words:

$$\boldsymbol{Z}^{\boldsymbol{P}} = \mathrm{MLP}_1(\boldsymbol{X}^{\boldsymbol{P}}), \qquad \boldsymbol{Z}^{\boldsymbol{P}} \in \mathbb{R}^{N_c \times K \times D^1_{\mathrm{patch}}}; \tag{13}$$

$$\boldsymbol{e}^{\boldsymbol{P}} = \max_j \left(\mathrm{MLP}_2 \left(\left[\boldsymbol{Z}^{\boldsymbol{P}} \middle| \max_k \left(\boldsymbol{Z}^{\boldsymbol{P}}_{:,k} \right) \right] \right)_{:,j} \right), \qquad \boldsymbol{e}^{\boldsymbol{P}} \in \mathbb{R}^{N_c \times D_{\mathrm{embed}}}; \tag{14}$$

where MLP_i is a multi-layer perceptron consisting of two linear layers interjected with a normalization and non-linear activation function.

A.1.4 Position Embedding

The separated positions of the patches are similarly prepared for downstream processing. In contrast to textual language and images where tokens or patches are associated with discrete positions, the center points of each point patch are embedded in a continuous $\mathbb{R}^3$ Euclidean space. While many different methods have been proposed to reinject the positional information of point clouds [75–77], a simple and efficient absolute positional embedding (APE) is used here. The embedding is defined by a learnable mapping $f^{\mathrm{pos}} : \mathbb{R}^3 \mapsto \mathbb{R}^{D_{\mathrm{embed}}}$, and is implemented with a simple two-layer MLP:

$$\boldsymbol{e}^{\mathrm{pos}} = \mathrm{MLP}(\boldsymbol{c}), \qquad \boldsymbol{e}^{\mathrm{pos}} \in \mathbb{R}^{N_c \times D_{\mathrm{embed}}}. \tag{15}$$

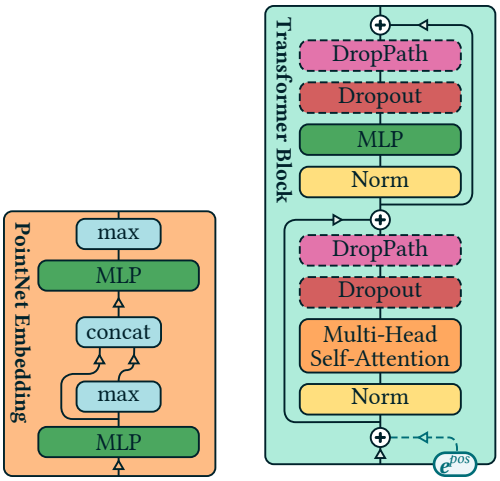
A.2 Contextualizing Encoder

At the core of the pipeline is the transformer encoder, which contains the majority of the parameters and carries out the bulk of the processing work. This encoder utilizes global self-attention across the patch tokens, thereby incorporating shape information from the entire point cloud to obtain globally contextualized embeddings.

A.2.1 Input

To obtain the input to the tranformer encoder, a learnable class embedding $\boldsymbol{e}^{\mathrm{CLS}} \in \mathrm{R}^{1 \times D_{\mathrm{embed}}}$ is prepended to the patch embedding tokens, yielding the input $\boldsymbol{z}^0$ of the encoder model:

$$\boldsymbol{z}^0 = \left[\boldsymbol{e}^{\mathrm{CLS}}; \boldsymbol{e}^{\boldsymbol{P}}\right], \qquad \boldsymbol{z}^0 \in \mathbb{R}^{(1+N_c) \times D_{\mathrm{embed}}}. \tag{16}$$



(a) PointNet Embedding (b) Transformer Block

Figure 9: Building blocks of the processing pipeline.

This additional CLS token is not associated with a position in $\mathbb{R}^3$ and serves to build up a global representation, which is beneficial for non-localalized tasks such as classification or the global self-distillation objective of AsymDSD.

A.2.2 Transformer Encoder

A standard transformer encoder [74] with pre-normalization is used, following the overall model structure of ViT [53]. It consists of a stack of L transformer blocks (Figure 9b). Due to the expected importance of positional information of the patches, the position embedding is readded to the input of each block. This diverges from the typical approach where the position embedding is added once before the first block of the transformer. Notably, the model maintains a fixed embedding size throughout the depth of the network.

B Additional Implementation Details of AsymDSD

B.1 ShapeNet Pre-Training

Pre-training details including pre-processing, model and training hyperparameters are provided in Table 7.

B.2 Inverse Block-wise masking for point clouds

Block-wise masking has been explored for point cloud data in several studies. However, the common implementation involves sampling a single block [21, 22, 34]. AsymDSD generalizes this by sampling any number of blocks of a pre-determined size B_s expressed in the number of patches. Furthermore, we consider inverse block-wise masking, following Baevski et al. [31], where the blocks instead indicate which patches to keep.

The difficulty with sampling multiple blocks is that they may partially overlap, which makes it difficult to mask the desired ratio of patches. To address this issue, the mask ratio is slight increased by adjust ratio A_r [31] to increase the number of blocks N_B to be sampled. Specifically:

$$N_B = \text{round}\left(N_c * \frac{(1 - M_r) + A_r}{B_s}\right), \tag{17}$$

where N_B is the number of blocks, N_c the number of patches, M_r the mask ratio, A_r the adjust ratio, and finally B_s the block size.

The blocks are subsequently generated by sampling N_B center positions, and accumulating the B_s nearest patches according to the L_2-distance. This may lead to over- or under-masking depending

on the amount of overlap. However, this is resolved by randomly swapping the mask bits until the desired mask ratio is achieved. The adjust ratio A_r can be chosen such that the number of swaps to be performed is minimized.

Table 7: AsymDSD's hyperparameters for ShapeNet pre-training.

(a) The data pre-processing parameters including cropping and masking.

Parameter	Symbol	Value	
Data Pre-processing			
# Points		16 384	
Augmentation-1		Rotate z-axis	
Augmentation-2		Anistropic Scaling $[0.8, 1.2]$	
Normalization		Unit Sphere	
Cropping			
		Global	Local
# Crops	N_G, N_L	2	4
# Points	N	1024	256
# Patches	N_c	64	16
# Patch Points	K	32	32
Crop Fraction		$[0.4, 1.0]$	$[0.05, 0.4]$
Masking			
Mask Sampler		Inverse block-wise	
Multi Mask	N_{mm}	4	
Mask Ratio	M_r	0.7	
Block Size	B_s	6	
Adjust Ratio	A_r	0.1	

(b) The training hyperparameters.

Parameter	Symbol	Value
Training		
Batch Size	$\mathcal{B}$	128
Epochs		300
Precision		FP16 mixed [78]
Optimizer		
Optimizer		AdamW [79]
LR Schedule	λ_{lr}	Cosine Annealing
Base Learning Rate		5.0×10^{-4}
# LR Warmup Epochs		10
Momentum Decay	β_1, β_2	0.9, 0.999
Weight Decay		0.05
KoLeo Scale	α	0.01
Gradient Clip Norm		10.0
Self-Distillation		
EMA Schedule	η	Cosine
EMA Start, End	$[\eta^0, \eta^E]$	[0.995, 1.000]
Centering Momentum	η_l	0.9
Student Temp	τ_s	0.1
Teacher Temp Schedule		Linear Warmup
Teacher Temp CLS	τ_t^{CLS}	$[0.04, 0.07]$
Teacher Temp Patch	τ_t^{patch}	$[0.05, 0.07]$
# Teacher Warmup Epochs		10

(c) The hyperparameters and details of the model.

Parameter	Symbol	Value
Shared Model Defaults		
Normalization		RMSNorm [80]
Activation		GELU [81]
Linear Bias		False
Patch Embedding (Sec. A.1.3)		
MLP_1 Dims		128, 256
MLP_2 Dims		512, 384
Position Embedding (Sec. A.1.4)		
MLP Dims		128, 384
Contextualizing Encoder (Sec. A.2)		
Transformer Block		Transformer Encoder
Embedding Dim	D_{embed}	384
MLP Expansion Dim		1536
# Layers	L	12
# Attention Heads	H	6
Predictor		
Transformer Block		Transformer Decoder
Embedding Dim	D_{embed}	192
MLP Expansion Dim		768
# Layers	L	6
# Attention Heads	H	3
Projection head		
MLP Dims		1024, 1024, 256
Output Dim	N_{tok}	4096
Linear Bias		True

C Scaling Pre-Training

C.1 Mixture Dataset

To scale the amount of training data, 3D models and scans from various sources were accumulated to make one large diverse datasets. Table 8 provides an overview of these datasets and their basic properties. The first 10 listed datasets comprise a total of 133 668 instances. When combined with the Objaverse dataset [4], they form *Mixture*, resulting in a total of 930 752 instances.

To provide some more details on the compilation process: for synthetic datasets without pre-existing point clouds, we uniformly sampled $16\,\mathrm{k}$ points from the surface of each mesh. In the case of scanned scene datasets with annotated objects—specifically S3DIS [82] and SUN RGB-D [83]—individual objects were cropped from the scenes and saved as individual instances. Notably, object instances with less than $2\,\mathrm{k}$ points were thrown away—these are predominantly of the category *clutter*. Additionally, all objects were rotated to their natural upright position in a shared reference frame. Any other features, such as color or normals, were not used.

C.2 Training details

These models that are pre-trained on *Mixture* are referred to as **AsymDSD-S*** for the ViT-S-sized model and **AsymDSD-B*** for the ViT-B model. The architectural details of the larger model are shown in Table 10, with parameter counts for both models listed in Table 9. Notably, AsymDSD-B scales the patch embedding to accommodate the larger ViT-B contextualizing encoder, and upgrades the predictor from a ViT-Ti to a ViT-S, with half the usual number of layers.

AsymDSD-S* was trained for 100 epoch on a batch size of 128, totaling $727\,\mathrm{k}$ optimization steps, in roughly 100 hours on a single A100 GPU. AsymDSD-B* was trained for both 50 epochs, taking around 175 hours, respectively, on the same hardware.

C.3 Evaluation on Objaverse-LVIS

The Objaverse dataset contains a subset of approximately $47\,\mathrm{k}$ objects annotated with one of the 1,156 categories from the LVIS dataset [84]. Unlike other datasets, it does not come with a predefined training or test split and is typically used for zero-shot evaluation in language-aligned models [85, 86]. Since our model does not produce language-aligned representations, we assess its representational quality through few-shot probing using both linear and kNN classifiers.

In particular, 10 instances are randomly sampled per category, and the remaining instances are added to the test set. We exclude any category with 10 or fewer instances, resulting in a total of 1060 remaining categories. Again, we exclude any additional features such as color and sample 1024 points per instance. The few-shot sampling strategy was repeated 10 times to remove most of the noise from sampling or training. The results from these experiments are shown in Table 11.

Table 8: Datasets for scaling pre-training. † indicates that object instances are sampled from scenes.

Dataset	# Instances	# Classes	Type
ShapeNetCore v2 [36]	52 470	55	Synthetic
3D-FUTURE [87]	16 560	50	Synthetic
ScanObjectNN [37]	16 034	15	Scanned
ModelNet40 [38]	9843	40	Synthetic
S3DIS† [82]	8948	14	Scanned
SUN RGB-D† [83]	8451	-	Scanned
Amazon Berkeley Objects [88]	7953	-	Synthetic
OmniObject3D [89]	5911	216	Synthetic
Toys4K [90]	4000	105	Synthetic
Google Scanned Objects [91]	1030	-	Scanned
Objaverse [4]	797 084	1156+	Mixed
Mixture	930 752	-	**Mixed**

Table 9: The total number of parameters of each module.

Module	Symbol	# Parameters (M)	
		AsymDSD-S	AsymDSD-B
Patch Embedding	f_ϕ^{embed}	0.5	6.5
Transformer Encoder	f_ϕ^{ctx}	21.2	85.0
Predictor	f_ψ^{pred}	2.9	11.2
Projection Head	f_ω^{proj}	2×2.8	2×3.2

Table 10: The hyperparameters and details of AsymDSD-B.

Parameter	Value
Patch Embedding (Sec. A.1.3)	
MLP_1 Dims	128, 256, 512
MLP_2 Dims	1024, 768
Position Embedding (Sec. A.1.4)	
MLP Dims	128, 768
Contextualizing Encoder (Sec. A.2)	
Transformer Block	Transformer Encoder
Embedding Dim	768
MLP Expansion Dim	3072
# Layers	12
# Attention Heads	6
Drop Path [68]	0.1
Predictor	
Transformer Block	Transformer Decoder
Embedding Dim	384
MLP Expansion Dim	1536
# Layers	6
# Attention Heads	6

Table 11: Top$k$ few-shot performance on **Objaverse LVIS** subset. The number of shots per category is 10 and the average is reported over 10 independent runs. kNN uses the 5-nearest neighbors.

Method	Linear			kNN		
	Top1	Top3	Top5	Top1	Top3	Top5
AsymDSD-S*	37.75	57.43	64.70	33.30	49.77	54.27
AsymDSD-B*	**38.36**	**58.22**	**65.43**	**33.68**	**50.24**	**54.68**

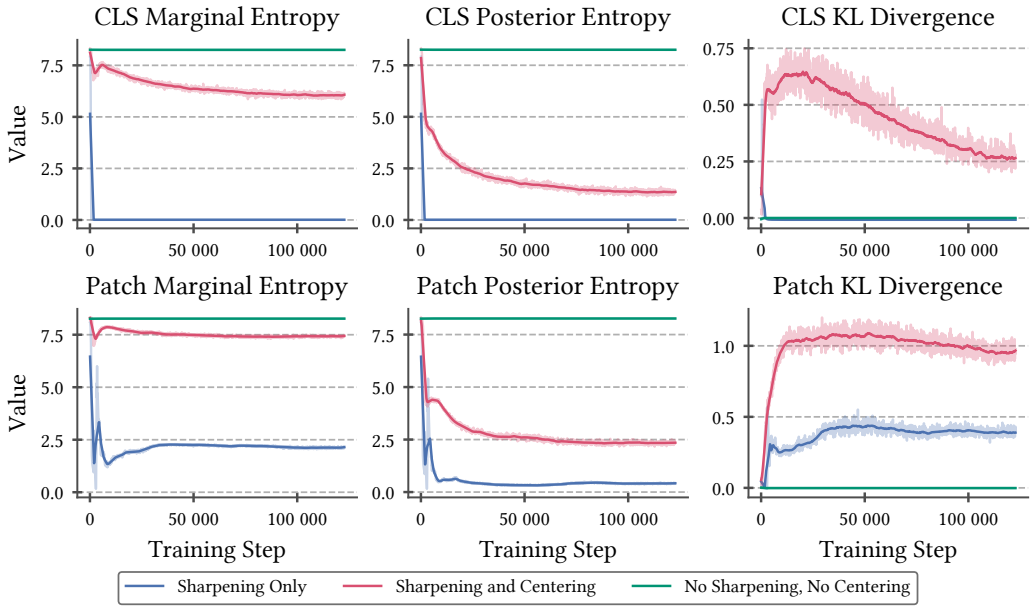


Figure 10: Marginal and posterior entropy, and KL divergence during training.

D Properties and Ablations of AsymDSD

This section presents additional experiments and results to gain deeper insights into the properties of AsymDSD.

D.1 Student-Teacher dynamics

The dynamics that unfold between the student and teacher are central to the effectiveness of SSRL with AsymDSD. In particular, we demonstrate the occurrence of representation collapse and highlight the importance of centering and sharpening mechanisms in mitigating this issue. In addition, we look at the teacher performance relative to the student.

D.1.1 Sharpening and Centering

For AsymDSD, sharpening and centering on the discrete targets is a central method of defense against the main modes of collapse. To demonstrate the effectiveness of these techniques, we trained AsymDSD without sharpening and centering, only with sharpening, and with both.

To determine wether collapse occurs in these setups, we compute the empirical marginal and posterior entropy over the batches $\mathcal{B}$ during training:

$$H\left(p^t_{\theta'}(\mathrm{z}_i)\right) \approx H\left(\sum_{x\in\mathcal{B}} p^t_{\theta'}(\mathrm{z}_i \mid x)\right), \tag{18}$$

$$\mathbb{E}_{\mathrm{x}\sim p(\mathrm{x})}[H(p^t_{\theta'}(\mathrm{z}_i \mid \mathrm{x}))] \approx \sum_{x\in\mathcal{B}} H\left(p^t_{\theta'}(\mathrm{z}_i \mid \mathrm{x})\right), \tag{19}$$

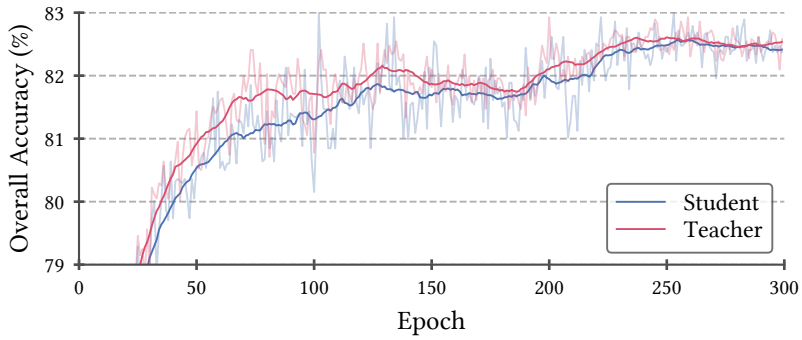


Figure 11: Student versus teacher performance. It shows plots of the overall accuracy on the hardest subset of ScanObjectNN with a linear SVM.

where $i = \text{CLS}$ or $i \in \mathcal{M}$ for the CLS-token or the patch tokens respectively. These metrics are plotted in Figure 10 alongside the KL divergence from the two training objectives $R^{\text{CLS}}(\theta, \theta')$ and $R^{\text{MPM}}(\theta, \theta')$. Note that we can simply decompose the cross-entropy to obtain the KL divergence: $D_{\text{KL}}(\text{x} \parallel \text{y}) = H(\text{x}, \text{y}) - H(\text{x})$.

Without sharpening and centering, the posterior collapses to the uniform distribution over both the CLS- and patch tokens, as indicated by the maximum entropy for $N_{\text{tok}} = 4096$ of $\log(4096) \approx 8,318$. On the other hand, when sharpening the targets, the marginal entropy falls to zero on the CLS-token, which indicates that the model always assigns a probability of 1.0 to the same token. This effect is not observed for the patch tokens, which demonstrate a non-zero marginal entropy larger than the posterior entropy. In this scenario we also observe non-zero KL divergence. We believe this greater stability on the patch tokens to be a result of the predictor module, which has been shown to be a key component in stabilizing training (Tab. 5c) [60, 32].

When combining sharpening with centering, as per AsymDSD, both modes of collapse are avoided, as the marginal entropy now stays high while the posterior entropy goes down during training, as desired. After all, the difference between the two quantifies the mutual information between the input x and latent z_i.

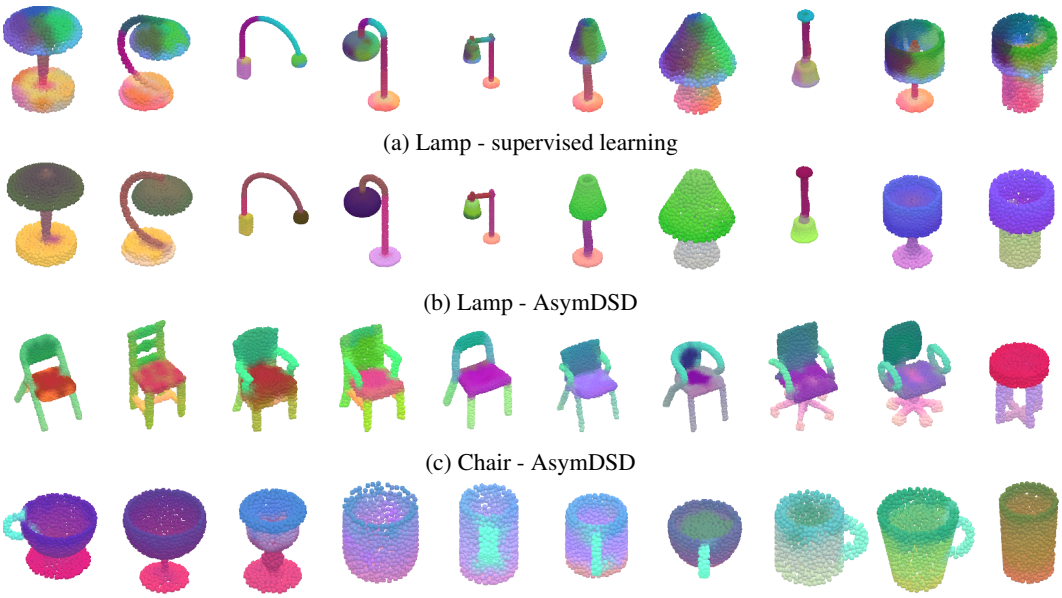
D.1.2 Outperforming Teacher

A property that is sometimes witnessed with joint embedding architectures with a momentum teacher, is the teacher outperforming the student with the idea that the teacher guides the student towards higher quality representations [92, 11]. As shown in Figure 11, AsymDSD demonstrates this effect with the teacher on average outperforming the student during training when evaluated with a linear SVM on ScanObjectNN. That said, this performance difference was not observed on the ModelNet40 dataset. However, this might be related to fact that peak accuracy with a linear SVM on ModelNet40 is obtained at around 10% of the total training duration.

D.2 Representational Quality

For a qualitative assessment of the representations of the AsymDSD's pre-trained encoder, we project patch embeddings to RGB space with t-SNE. Specifically, patch embeddings of the AsymDSD-B* pre-trained encoder are computed for 200 samples from the same object category in the ModelNet40 dataset. Each point is colored according to the inverse distance weighted average of the three nearest RGB patch embeddings.

The visualizations are shown for three object categories in Figure 12. We also included visualizations from a model that is trained with supervised learning on the ModelNet40 dataset. It stands out that AsymDSD shows high congruency among the projected embeddings from patches that comprise a semantic part of an object. This uniformity is also manifested between parts from different objects, e.g. the cyan colored ears of the cups, or the green cone shaped lamp covers. Interestingly, the latter example of cone shaped lamp covers also demonstrates strong scale invariance. We believe that cropping and multi-crop plays a key role in learning this invariance, as it leads to varying patch



(a) Lamp - supervised learning

(b) Lamp - AsymDSD

(c) Chair - AsymDSD

(d) Cup - AsymDSD

Figure 12: Zero-shot coloring of points according to inverse distance weighted nearest patch embeddings from AsymDSD-B* projected to RGB space with t-SNE. The projection is computed with 200 instances from the same class with a perplexity of 30.

densities for the same object. Notably, these properties with strong semantic separation are not present with the model that is trained with standard supervision. This supervised model instead shows a strong positional bias among the projected embeddings.

E Additional Ablations

E.1 Predictor in AsymDSD

Table 12 presents results from ablation experiments on the dual-objective formulation involving the predictor module. These results suggest that jointly optimizing both objectives improves robustness: even when the predictor is removed (cf. Table 5c), downstream performance does not collapse. Nonetheless, there is a modest drop in accuracy, and throughput decreases significantly. We hypothesize that this collapse-resistance stems from global invariance learning mitigating the model's collapse towards representing only the positional queries—given the variability of such queries across different crops.

Nevertheless, some performance degradation is expected when the predictor is removed due to leakage of the overall object shape through all mask queries. This effect should become pronounced when unmasked local crops are removed and the predictor is also excluded—i.e., when all instances expose a large portion of the coarse shape. Under this configuration, results indeed show a substantial drop in performance, most notably in linear SVM accuracy.

Table 12: Ablations on **AsymDSD-S**.

Method	Fig.	MC	Attention		Mem.	It/s	ModelNet40		ScanObjectNN	
			Self	Cross			SVM	FFt	SVM	FFt
AsymSD	13c	✓	×	c	25.6	2.57	**93.03**	**94.13**	**82.34**	**90.53**
	13a	✓	c+m	×	26.8	2.42	92.67 ↓ 0.36	93.84 ↓ 0.29	82.13 ↓ 0.21	89.45 ↓ 1.08
– predictor	-	✓	×	×	31.9	2.07	92.87 ↓ 0.17	94.00 ↓ 0.13	81.82 ↓ 0.52	89.49 ↓ 1.04
	13c	×	×	c	22.8	3.01	93.03 → 0.0	93.84 ↓ 0.29	81.99 ↓ 0.35	89.52 ↓ 1.01
– predictor	-	×	×	×	29.3	2.51	92.18 ↓ **0.85**	93.64 ↓ **0.49**	79.88 ↓ **2.46**	88.13 ↓ **2.40**

In fact, when using a predictor and optimizing the joint objective, *multi-crop* may no longer be essential. In particular, inverse block-wise masking can simulate localized contexts similar to those provided by local crops, but without incurring the additional computational cost of processing them separately. However, under the current implementation, the variable nature of local crops allows for significantly smaller contexts compared to what is achievable through masking applied to global crops. This likely explains the observed performance improvements when local crops are included alongside the dual objective. However, these findings suggest a promising direction for future work: exploring the use of variable mask ratios as a potential alternative to multi-crop strategies.

E.2 Objective Function

The MPM objective R^{MPM} is presented as a cross-entropy objective between the teacher and student posterior over the masked patches (Eq. 5). However, as demonstrated in other works [32], the high-dimensional student representations (i.e., before projection to discrete latent z) can also be directly regressed onto the teacher representations without representational collapse ensuing.

Table 13 presents results comparing these approaches. We find that direct regression with a smooth L_1-loss ($\beta = 2$) [93] without the KL divergence-based R^{MPM} objective—and thus no MI-maximizing measures on a discrete latent projection—does not lead to collapsed representations. However, we observe from hyper-parameters sweeps across several training runs that it is difficult to stabilize training. This is reflected in down-stream performance that eventually starts decreasing during training. Interestingly, when regression is combined with R^{MPM}, it stabilizes training and improves performance across all benchmarks compared to training only on R^{MPM}. We therefore use this setting combining the two losses in **AsymSD-MPM-S**. **AsymDSD-S** only uses cross-entropy minimization, as we did not observe down-stream improvement by adding an additional regression loss.

Table 13: Loss functions for MPM. `CE` indicates the cross-entropy objective R^{MPM}. `RG` indicates a regression loss.

CE	RG	ModelNet40		ScanObjectNN	
		SVM	FFt	SVM	FFt
✓	×	92.10 ↓ 0.53	93.76 ↓ 0.28	78.97 ↓ 0.49	87.75 ↓ 0.83
×	✓	90.51 ↓ **2.12**	93.44 ↓ **0.60**	69.74 ↓ **9.72**	87.37 ↓ **1.21**
✓	✓	**92.63**	**94.04**	**79.46**	**88.58**

E.2.1 Additional Predictor Designs

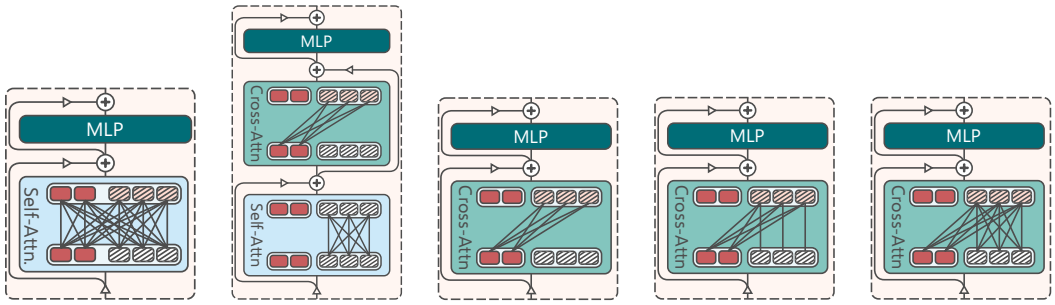
The predictor module is a central component of MPM with Asymmetric Self-Distillation. It was argued that it not only enhances training efficiency but also improves the overall training architecture by mitigating issues such as global shape leakage, distribution mismatch, and representation collapse. These results were presented in Table 5c, but we experimented with several alternative predictor designs. The results of these experiments are presented in Table 14, with the designs shown in Figure 13.

While it was already shown that the typical transformer encoder implementation (Fig. 13a) leads to significantly reduce performance. We also tested cross-attention with concatenation of the query token itself (Fig. 13d), but observe that this yields no significant benefit, at the cost of higher memory usage and lower throughput.

Multi-Block. So far the two extremes of separately predicting all masked patches ($p_\theta(\mathrm{z_i} \mid \tilde{\mathbf{x}}, \mathbf{c_i}),\ \forall i \in \mathcal{M}$) or predicting all masked patches at once ($p_\theta(\mathrm{z_i} \mid \tilde{\mathbf{x}}, \mathbf{c}_\mathcal{M}),\ \forall i \in \mathcal{M}$) have been explored. This can be generalized to predicting any number of masked patches at once [32]. Block-wise masking is particularly suitable alongside such generalized prediction objective, given that it provides local neighborhoods of masked regions to be 'unmasked'. In this way, the prediction is performed by providing the positional queries for a block of patches, allowing a coordinated build-up of representations over a region that extends beyond the bounds of a single patch without revealing the global shape of the object.

For *multi-block*, we use blocks with size $B_s = 9$, using two suitable designs for a coordinated build up, as shown in Figure 13b and 13e. Although these designs achieved strong downstream

performance, they offered no clear advantage over the simpler approach of predicting each patch token independently, while coming with the cost of lower throughput.



(a) Self-attention on context + mask $\mathcal{O}\left((S+E)^2\right)$

(b) Self-attention + cross-attention $\mathcal{O}\left(E^2+SE\right)$

(c) Cross-attention on context $\mathcal{O}\left(SE\right)$

(d) Cross-attention on context + self $\mathcal{O}\left(SE^2\right)$

(e) Cross-attention on context + mask $\mathcal{O}\left(SE^2\right)$

Figure 13: Different transformer block designs for the predictor. The time complexity of the attention computations is included, with $S = |\tilde{\mathcal{M}}|$ and $E = |\mathcal{M}|$ the number of context and masked patches respectively.

Table 14: Ablations on the predictor of **AsymSD-MPM-S**. `RG` indicates a regression loss. `MB` indicates the use of multi-block. Of the attention tokens, `C` is the visual context, `M` the mask queries, `S` the query token itself. Mem. is the total memory usage with a batch size of 128 (with multi-mask set to 8) in GiB; and It/s the throughput in iterations per second. For more details of the design refer to the indicated figures.

Method	Fig.	MB	Attention		Mem.	It/s	ModelNet40		ScanObjectNN	
			Self	Cross			SVM	FFt	SVM	FFt
(**MPM-S**)	13c	×	×	`c`	16.2	4.80	**93.31**	**94.04**	**81.06**	**88.58**
	13d	×	×	`c+s`	17.5	4.65	93.03 ↓ 0.28	94.04 → 0.0	81.06 → 0.0	87.96 ↓ 0.62
	13a	×	`c+m`	×	17.9	4.32	88.17 ↓ **5.14**	92.87 ↓ **1.17**	77.00 ↓ **4.06**	85.98 ↓ **2.60**
+ Multi-Block	13b	✓	`m`	`c`	15.6	3.68	93.19 ↓ 0.12	93.88 ↓ 0.16	81.85 ↑ 0.79	87.51 ↓ 1.07
	13e	✓	×	`c+m`	15.4	4.01	93.07 ↓ 0.24	94.29 ↑ 0.25	80.95 ↓ 0.11	88.17 ↓ 0.41