Supplementary Material

A More results

ShapeNetPart In Table 1, we compare the categorical mIoU on ShapeNetPart with other methods. With a PointViT backbone, we get the highest class mIoU at 84.4% and the highest instance mIoU at 86.0%, outperforming previous self-supervised learning approaches (OcCo [12] and Point-BERT [16]). It also outperforms standard train-from-scratch point cloud backbones like PointNet++ [9] and DGCNN [13]. For all categories, our method either has the highest accuracy or is among the best. Thanks to our dense discriminative pretraining objective, in which we densely classify points over the 3D space, we are able to obtain good performance when scaling to dense prediction tasks like part segmentation.

Methods	cls.	ins.	aero	bag	cap	car	chair	earp.	guit.	knif.	lamp	lapt.	mot.	mug	pist.	rock.	skt.	table
PointNet [8]	80.4	83.7	83.4	78.7	82.5	74.9	89.6	73.0	91.5	85.9	80.8	95.3	65.2	93.0	81.2	57.9	72.8	80.6
PN++ [9]	81.9	85.1	82.4	79.0	87.7	77.3	90.8	71.8	91.0	85.9	83.7	95.3	71.6	94.1	81.3	58.7	76.4	82.6
DGCNN [13]	82.3	85.2	84.0	83.4	86.7	77.8	90.6	74.7	91.2	87.5	82.8	95.7	66.3	94.9	81.1	63.5	74.5	82.6
PointViT	83.4	85.1	82.9	85.4	87.7	78.8	90.5	80.8	91.1	87.7	85.3	**95.6**	73.9	94.9	83.5	61.2	74.9	80.6
OcCo [12]	83.4	85.1	83.3	85.2	**88.3**	79.9	90.7	74.1	**91.9**	87.6	84.7	95.4	75.5	94.4	84.1	63.1	75.7	80.8
PN-BERT [16]	84.1	85.6	**84.3**	84.8	88.0	79.8	91.0	**81.7**	91.6	**87.9**	85.2	**95.6**	75.6	94.7	84.3	63.4	76.3	81.5
MaskPoint (Ours)	**84.4**	**86.0**	84.2	**85.6**	88.1	**80.3**	**91.2**	79.5	**91.9**	87.8	**86.2**	95.3	**76.9**	**95.0**	**85.3**	**64.4**	**76.9**	**81.8**

Table 1: Part segmentation results on ShapeNetPart [15]. **Bold** and underline numbers denote best and second best performance, respectively.

ScanNet Table 2 reports per-class average precision on 18 classes of ScanNetV2 with a 0.25 box IoU threshold. Relying on purely geometric information, our method exceeds 3DETR [7] in detecting objects like curtain, garbagebin, table, desk, *etc*, where geometry is a strong cue for recognition. These results indicate that our mask based discriminative pretraining framework is effective in learning strong geometric representations. More importantly, our model outperforms 3DETR on classes where it has relatively low AP, *e.g.*, picture, door, curtain, refrigerator, *etc*, which demonstrates the usefulness of pretraining: with the pretrained knowledge relevant to those hard classes, the model is able to make more accurate predictions than the training from the scratch baseline.

Model	$\mathbf{AP}_{25}$	cab.	bed	cha.	sofa	tab.	door	win.	boo.	pic.	cou.	desk	cur.	ref.	sho.	toi.	sink	bat.	gar.
3DETR [7]	62.2	50.2	87.0	86.0	87.1	61.6	46.6	40.1	54.5	9.1	62.8	69.5	48.4	50.9	68.4	97.9	67.6	85.9	45.8
Ours	63.4	51.8	82.5	85.9	86.8	69.8	50.9	36.9	47.3	10.7	59.6	76.3	65.9	55.6	66.4	99.1	61.5	83.7	49.8
Ours (12×)	64.2	49.5	81.0	87.2	86.3	65.2	51.3	42.6	56.7	16.2	56.8	73.8	59.6	56.0	77.0	97.8	66.6	85.0	47.7

Table 2: 3D object detection scores per category on the ScanNetV2 dataset, evaluated with bbox mIoU 0.25. Ours (12×): 12 encoder blocks.

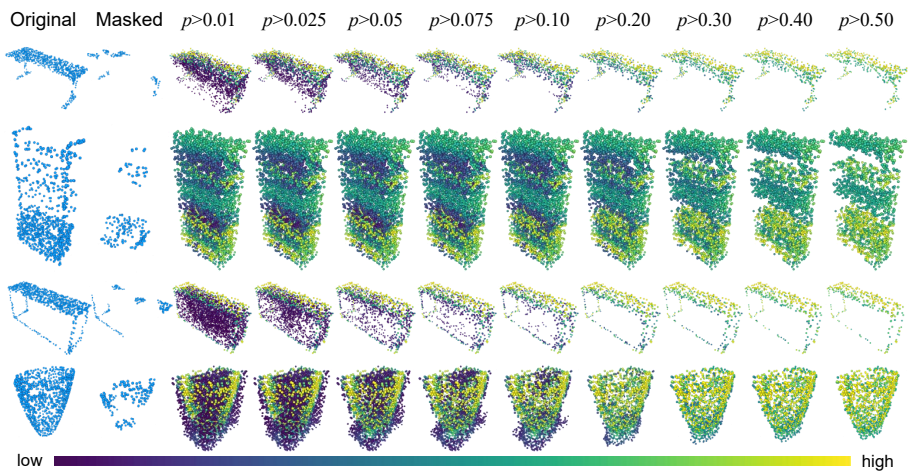


Fig. 1: **Reconstruction results.** We densely perform the discriminative occupancy classification task in 3D space, and visualize the predicted occupancy probability. By varying the confidence threshold $\hat{p}$, we show that our model is able to predict a continuous probability distribution of the occupancy function.

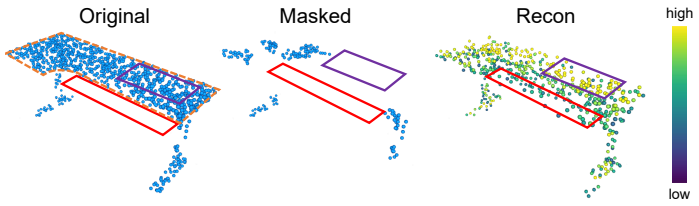


Fig. 2: **A closer look at occupancy distribution.** Although there are no points present in both **red** and **purple** regions of the masked point cloud, the reconstructed probability distribution correctly reflects that of the original point cloud: a lower occupancy in **red** region, and a higher occupancy in **purple** region.

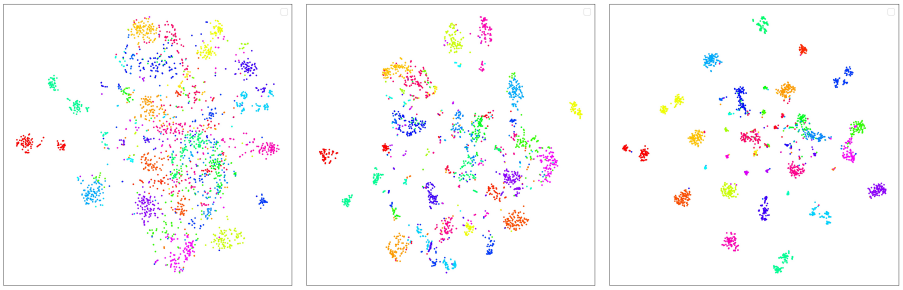
More reconstruction visualizations We densely perform the discriminative occupancy classification task in 3D space, and visualize in Fig. 1 the predicted occupancy probability. In different columns, we vary the occupancy threshold τ, and only show the points with occupancy probability prediction that is higher than the given threshold. We can see that our model is able to output a continuous probability distribution of the occupancy function, even if it is only trained with discrete occupancy values from the sampled points.

When we take a closer look at the occupancy distribution, we find several interesting clues on how the model is modeling the probability distribution impressively well. We show our findings in Fig. 2. There are no points present in both red and purple regions of the masked point cloud, while in the original point cloud, there are points present in the purple region, and no points are in the red

region. In the reconstructed probability distribution, the model predicts a low occupancy probability in the red region, and a high occupancy in the purple region.

We find such predictions align with how a human might understand the scene. First, although there are no points in the purple region of the masked point cloud, given the partial view of the top-left region of the desk top, and the regions where the desk legs are present, it is very likely that there are points present in the purple region (top-right region of the desk top). As for the red region, the model's prediction can be interpreted as follows: usually desk tops are rectangle-shaped; however, there do exist desks whose surface shrinks inside the region where the person sits. Given that there is not a decisive evidence that indicates how this particular desk instance is shaped, the model produces predictions with probability around 0.7, which is lower than other regions that are more certain (yellow points in Fig. 2 "Recon", with p>0.9).

These two intriguing and encouraging visualizations suggest that our pretrained model is capable of modeling a continuous occupancy probability distribution, and it has learned a deep understanding of the input scene.



(a) Training from scratch (b) Pretrained on ShapeNet (c) Finetuned on ModelNet40

Fig. 3: t-SNE visualization of the encoder features for ModelNet40 under three settings: (a) training from scratch, (b) pretrained on ShapeNet, and (c) finetuned on ModelNet40.

t-SNE visualizations We show the t-SNE visualizations of the extracted feature vectors from our approach in Fig. 3. We use the class token from the encoder output as the high dimensional feature representation for t-SNE. Three setting are adopted here: (a) training from scratch, (b) pretraining on ShapeNet [1], and (c) finetuning on ModelNet40 [14].

When training the ModelNet40 classification model from scratch, the resulting features from different categories become heavily entangled, which can leads to less interpretable and robust predictions for new test-time inputs. In contrast, when pretraining the model on ShapeNet using our proposed MaskPoint, the features are much more distinguishable from each other. Furthermore, after finetuning on ModelNet40, the projected features from different classes become

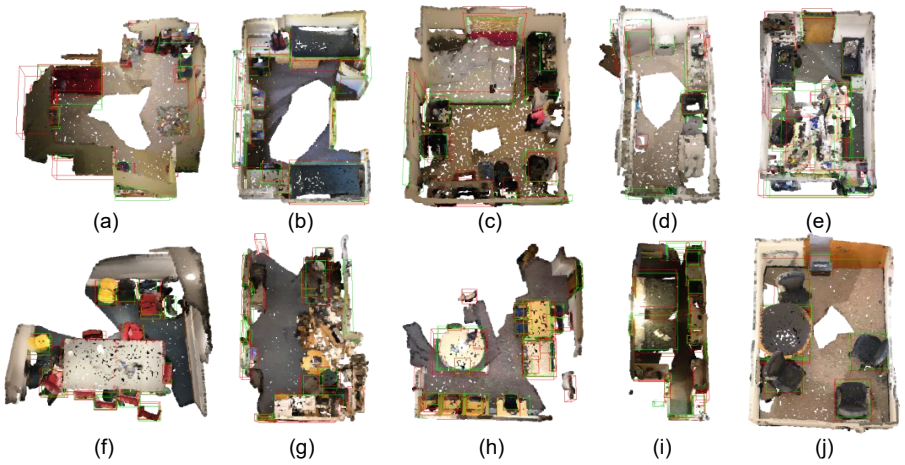


Fig. 4: **Qualitative results of 3D object detection on ScanNetV2 [2].** We show ground truth in **green** and predictions in **red** bounding boxes.

clearly separable from each other, which indicates the effectiveness of our approach. Interestingly, the feature clusters in our approach are quite tight. Such feature layout indicates that we can learn a more *compact and disjoint decision boundary*, which has been evaluated to be critical in machine learning applications such mixup [17,10] and uncertainty estimation in deep learning [3].

3D object detection visualizations We show 3D object detection visualizations of ScanNetV2 in Figure 4 with green ground truth bounding boxes and red predicted bounding boxes. Our model is capable of precisely localizing the obejct (Fig. 4b and Fig. 4j). Results indicate that our masked based discriminative pretraining can not only produce high-quality bounding boxes for the previously annotated objects, but also discover objects that are not annotated. For example, in Fig. 4c, our model produces the a bounding box for the bookshelf in the lower region; in Fig. 4f, it correctly locates the sofa in the center of the room.

B Additional Implementation Details

B.1 Pretraining

Transformer Encoder. We follow the standard Transformer design in [11,16] to construct our point cloud Transformer backbone, PointViT. It consists of a linear stack of 12 Transformer blocks, where each Transformer block contains a multi-head self-attention (MHSA) layer and a feed-forward network (FFN). LayerNorm (LN) is adopted in both layers. Following [16], we use the MLP-based positional embedding. We set the Transformer hidden dimension to 384, MHSA head number to 6, expansion rate of FFN to 4, stochastic drop path [5] rate to 0.1.

Module	Block	C_{in}	C_{out}	k	N_{out}	C_{middle}
Positional Embed.	MLP	3	384			128
Point Classify Head	MLP	384	2			64
Classification Head	MLP	768	N_{cls}			512, 256
Segmentation Head	MLP	387	384			384×4
	DGCNN	384	512	4	128	
	DGCNN	512	384	4	128	
	DGCNN	384	512	4	256	
	DGCNN	512	384	4	256	
	DGCNN	384	512	4	512	
	DGCNN	512	384	4	512	
	DGCNN	384	512	4	2048	
	DGCNN	512	384	4	2048	

Table 3: Detailed module design of MaskPoint. C_{in}/C_{out} denotes the input/output channels, C_{middle} denotes the hidden channels of MLP modules, N_{out} denotes the cardinality of the output point/feature set, k is the number of neighbors used in the k-NN operator.

Feed-forward network (FFN). Following [16], we use a two-layer MLP with ReLU and dropout as the feed-forward network. Dropout rate is set to 0.1.

Positional Embeddings. Following [16], we use a two-layer MLP with GELU [4] as the positional embedding module. All Transformer modules share the same positional embedding MLP module. Detailed configuration is shown in Table 3.

Point Classification Head. We use a simple two-layer MLP with GELU [4] for the point classification pretext task in pretraining. We use the binary focal loss [6] to balance the information from positive and negative samples. Detailed configuration is shown in Table 3.

ScanNet-Medium Pretraining Note that for ScanNet-Medium pretraining, we use the encoder with 3 Transformer blocks, where each block still consists of a MHSA layer and a FFN layer. LN and MLP positional embedding are also utilized in the encoder. Following the downstream architecture of 3DETR [7], we set the hidden dimension to be 256, the number of MHSA heads to be 4, and Dropout rate to be 0.1 for the Transformer. The hidden dimension is set to be 128 for the FFN layer.

For other settings such as positional embedding and classification head, the setting is exactly the same as the ShapeNet pretraining setting.

B.2 Finetuning

Classification We use a three-layer MLP with dropout for the classification head. The input feature to the classification head consists of two parts from the Transformer encoder: (1) the CLS token; (2) the max-pooled feature of other

config	value
epochs	300
optimizer	AdamW
learning rate	5e-4
weight decay	5e-2
LR schedule	cosine decay
warmup epochs	3
augmentation	Scale/Translate
batch size	128
# points	1024
# patches	64
patch size	32
mask ratio	0.90
mask type	random

Table 4: Pretraining setting on ShapeNet [1].

config	value
epochs	300
optimizer	AdamW
learning rate	5e-4
weight decay	5e-2
LR schedule	cosine decay
warmup epochs	10
augmentation	Scale/Translate
batch size	32(cls), 16(seg)
# points	1024(cls), 2048(seg)
# patches	64(cls), 128(seg)
patch size	32

Table 5: Finetuning setting on classification (cls) and segmentation (seg).

output features. These two features are concatenated together and fed into the classification head. Detailed configuration is shown in Table 3.

Part Segmentation The standard Transformer only has a single-scale feature output, which is not suitable for common head designs for dense prediction tasks like segmentation. Following [16], after getting the feature outputs from the Transformer encoder, we perform segmentation in two steps: (1) generating a multi-scale feature pyramid from the Transformer encoder outputs; (2) applying a standard feature propagation head for point cloud segmentation on the generated multi-scale feature maps to generate dense predictions.

We obtain the feature maps $f_{\{4,8,12\}} \in \mathbb{R}^{N_3 \times d}$ from the 4th, 8th, 12th layer, and our goal is to convert them to a feature pyramid with different cardinality $N_{\{0,1,2,3\}}$, where N_0 is the cardinality of the original point cloud $\mathcal{P}$, and $N_{\{1,2,3\}}$ are the desired cardinality of the feature maps at different scales; in our case, $N_{\{0,1,2,3\}} = \{2048, 512, 256, 128\}$.

First, we use furthest point sampling (FPS) to downsample the original point cloud $\mathcal{P}_0$ to different resolutions: $\mathcal{P}_{\{1,2,3\}} \in \mathbb{R}^{N_{\{1,2,3\}} \times 3}$, then a feature propagation module is used to upsample the feature maps $f_{\{4,8,12\}}$ to the corresponding cardinality $f^{up}_{\{4,8,12\}} \in \mathbb{R}^{N_{\{1,2,3\}} \times d}$.

After obtaining the multi-scale feature maps, we then apply the DGCNN module to propagate the features through different scales, $\hat{f}_4 = \text{DGCNN}(f^{up}_{\{4,8,12\}})$. Another feature propagation layer is then applied on $\hat{f}_4$ for upsampling to the highest resolution $\hat{f}_0 \in \mathbb{R}^{N_0 \times d}$.

Finally, we apply a pointwise MLP classifier on the features at the highest resolution $\hat{f}_0$ to obtain the segmentation results. Detailed configuration is shown in Table 3.

3D Object Detection We strictly follow the setting of the original 3DETR [7] model as the downstream 3D object detector. The points are first donwsampled

to 2048 points using a Set-Aggregation (SA) layer. The encoder is composed of 3 standard Transformer blocks. The decoder is comprised of 8 Transformer blocks using cross attention. During finetuning, only the weights of the SA layer and the encoder are transferred to the downstream tasks. The finetuning epoch number is 1080, the optimizer is AdamW with learning rate of 5×10^{-4} and weight decay of 0.1, the batch size is 8.

References

1. Chang, A.X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., et al.: Shapenet: An information-rich 3d model repository. arXiv preprint arXiv:1512.03012 (2015)
2. Dai, A., Chang, A.X., Savva, M., Halber, M., Funkhouser, T., Nießner, M.: Scannet: Richly-annotated 3d reconstructions of indoor scenes. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 5828–5839 (2017)
3. Du, X., Wang, X., Gozum, G., Li, Y.: Unknown-aware object detection: Learning what you don't know from videos in the wild. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (2022)
4. Hendrycks, D., Gimpel, K.: Gaussian error linear units (gelus). arXiv preprint arXiv:1606.08415 (2016)
5. Huang, G., Sun, Y., Liu, Z., Sedra, D., Weinberger, K.Q.: Deep networks with stochastic depth. In: European conference on computer vision. pp. 646–661. Springer (2016)
6. Lin, T.Y., Goyal, P., Girshick, R., He, K., Dollar, P.: Focal loss for dense object detection. In: ICCV (2017)
7. Misra, I., Girdhar, R., Joulin, A.: An End-to-End Transformer Model for 3D Object Detection. In: ICCV (2021)
8. Qi, C.R., Su, H., Mo, K., Guibas, L.J.: Pointnet: Deep learning on point sets for 3d classification and segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 652–660 (2017)
9. Qi, C.R., Yi, L., Su, H., Guibas, L.J.: Pointnet++: Deep hierarchical feature learning on point sets in a metric space. arXiv preprint arXiv:1706.02413 (2017)
10. Thulasidasan, S., Chennupati, G., Bilmes, J.A., Bhattacharya, T., Michalak, S.: On mixup training: Improved calibration and predictive uncertainty for deep neural networks. Advances in Neural Information Processing Systems **32** (2019)
11. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L.u., Polosukhin, I.: Attention is all you need. In: Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (eds.) Advances in Neural Information Processing Systems. vol. 30. Curran Associates, Inc. (2017), https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf
12. Wang, H., Liu, Q., Yue, X., Lasenby, J., Kusner, M.J.: Unsupervised point cloud pre-training via occlusion completion. In: ICCV (2021)
13. Wang, Y., Sun, Y., Liu, Z., Sarma, S.E., Bronstein, M.M., Solomon, J.M.: Dynamic graph cnn for learning on point clouds. Acm Transactions On Graphics (tog) **38**(5), 1–12 (2019)
14. Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., Xiao, J.: 3d shapenets: A deep representation for volumetric shapes. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 1912–1920 (2015)

15. Yi, L., Kim, V.G., Ceylan, D., Shen, I.C., Yan, M., Su, H., Lu, C., Huang, Q., Sheffer, A., Guibas, L.: A scalable active framework for region annotation in 3d shape collections. ACM Transactions on Graphics (ToG) **35**(6), 1–12 (2016)
16. Yu, X., Tang, L., Rao, Y., Huang, T., Zhou, J., Lu, J.: Point-bert: Pre-training 3d point cloud transformers with masked point modeling. arXiv preprint arXiv:2111.14819 (2021)
17. Yun, S., Han, D., Oh, S.J., Chun, S., Choe, J., Yoo, Y.: Cutmix: Regularization strategy to train strong classifiers with localizable features. In: ICCV (2019)