PerLA: Perceptive 3D language assistant

Supplementary Material

A. Introduction

In this supplementary material, we begin by thoroughly describing the components of PerLA that are distinct from the perspective scene encoder (Appendix B). Next, we offer a more technical explanation of Hilbert-based serialization and partitioning for point clouds, including detailed algorithms and an analysis of computational efficiency (Appendix C). Moreover, we provide the complete list of datasets that are involved in model training and testing (Appendix D). In addition, we include an extended analysis performed during the submission phase, and offer additional implementation details and deeper discussion of the results (Appendix E). Finally, we present additional interesting qualitative results of PerLA in comparison with state-of-the-art competitors.

B. More Details of PerLA

In this section, we describe in detail additional components in PerLA, including the off-the-shelf 3D encoder within our perceptive scene encoder, the multimodal prompts involved in the 3DLA interaction as well as their corresponding prompt encoders, and the multimodal adapter that integrates both multimodal prompt and the 3D scene representation to form query tokens that are interpretable by the LLM. Lastly, we explain the next token prediction loss.

B.1. 3D encoder

We adopt the same 3D encoder architecture as in LL3DA [9] to process the 3D point cloud. This 3D encoder first tokenizes the input into 2,048 point tokens, by uniformly sampling across the input point cloud using a set-abstraction layer [45]. The point tokens are then passed through three cascaded transformer encoder blocks, employing masking radius of 0.16, 0.64, and 1.44, respectively. To further refine the token representation, an additional set-abstraction layer is introduced between the first two transformer blocks, downsampling the tokens to 1,024. The final output of this 3D encoder is a feature matrix of shape $\mathbb{R}^{1,024\times256}$, where each of the point tokens is encoded as a 256-dimensional feature vector.

B.2. Multimodal prompts

Prompts in PerLA are multimodal and are designed to simulate user-driven interactions within 3D environments. Specifically, we consider both visual and textual prompts as in prior work LL3DA [9]. The *visual prompts* involve visual cues coming from the 3D content, *e.g.*, user clicks or bounding boxes around objects, while the *textual prompts* involve user instructions expressed in natural language format. Such multimodal prompts allow PerLA to interpret and respond effectively to intuitive user inputs. In the following, we present how each type of prompts is encoded.

Scene-aware visual prompt encoder. The visual prompt encoder aims to process visual prompts into representations that are then easier to be processed by the LLM. In addition to prior work which applies positional encoding followed by an MLP layer to process the visual prompts (as described in Eq. 3 of [9]), we further enhance this approach by augmenting the positional encoding with the detail-enriched global scene representations $\hat{\mathcal{F}}^g$, obtained by our perceptive scene encoder. Therefore, the visual prompt encoder is more aligned to the scene representation, helping to improve the model performance as empirically proved in Tab. I.

Specifically, each user click is first normalized to a range of $[0, 1]$ based on the dimensions of the input 3D scene, where $p_{\text{click}} \in \mathbb{R}^3$. We then encode p_{click} using 3D Fourier positional embeddings, denoted as $\text{pos}(p_{\text{click}})$. The box annotation is represented by the ROI feature $f_{\text{box}} \in \mathbb{R}^d$ with the center point p_{box} extracted by a pre-trained 3D object detector [8]. We first merge the two types of visual prompts with the scene representations regarding the neighborhood points, and then we use an MLP to project the merged representations as follows:

$$\boldsymbol{f}_{\text{cli}} = \text{MLP}_{\text{cli}}\left(\text{pos}(p_{\text{cli}}), h\left(\left\{\hat{\boldsymbol{f}}_i^g \mid \boldsymbol{p}_j^l \in \mathcal{N}(p_{\text{cli}})\right\}\right)\right),$$

$$\boldsymbol{f}_{\text{box}} = \text{MLP}_{\text{box}}\left(f_{\text{box}}, h\left(\left\{\hat{\boldsymbol{f}}_i^g \mid \boldsymbol{p}_j^l \in \mathcal{N}(p_{\text{box}})\right\}\right)\right),$$

where $h(\cdot)$ is max-pooling, and $\mathcal{N}(\cdot)$ denotes the $K^l/2$ nearest neighbors.

Textual prompt encoder. Textual prompts provide task-specific instructions to 3DLAs. For 3D dense captioning, we instruct the model to perform one of two tasks: "describe" or "describe and localize" the object, while for 3D question answering, we use textual instructions that ask the model to either "answer" or "answer and localize the related objects." Specifically, we encode the input text prompt $\mathcal{I}^t$ using a transformer architecture inspired by BLIP-2 [9, 31]. This transformer is initialized with a pre-trained BERT model to handle word and positional embeddings, producing text representations $\mathcal{F}^e \in \mathbb{R}^{T\times d_e}$.

B.3. Multimodal adapter

Since the 3D and language representations reside in distinct latent spaces, the multimodal adapter (MMA) aims to bridge the gap between outputs of frozen unimodals. MMA

aggregates such multimodal information with a fixed set of 32 learnable query tokens. Specifically, we implement MMA with a Q-Former architecture [31] with transformer layers, featuring 12 attention heads per layer. In each layer, these queries interact with the encoded visual prompts, $[f_{\text{cli}}; f_{\text{box}}]$, and the textual instructions, I_t, through a shared self-attention mechanism. Next, the learnable query tokens and visual prompts interact with our detail-enriched scene representation, $\hat{\mathcal{F}}^g$, via cross-attention. The output of the MMA is a set of 32 queries, denoted as $Q \in \mathbb{R}^{32\times768}$, which are then projected into the latent space of LLM through a simple linear projector.

B.4. Next token prediction loss

We employ standard language modeling conditioned on the text prompt $\mathcal{I}^t$, visual prompt $\mathcal{I}^v$, and point cloud $\mathcal{P}$, to train on a large text corpus $\mathcal{O}_1, \mathcal{O}_2, \cdots, \mathcal{O}_T$ by performing a next-token prediction task. The goal is to maximize the probability of $\mathcal{O}_{i+1}$ (the next token) conditioned on the sequence of prior tokens $\mathcal{O}_{i:1} = \mathcal{O}_i, \cdots, \mathcal{O}_1$, $\mathcal{I}^t$, $\mathcal{I}^v$ and $\mathcal{P}$. The learning objective, $\mathcal{L}_{pred}$, minimizes the cross-entropy loss as follows:

$$\mathcal{L}_{\text{pred}} = -\sum_i \log P_\theta \left(\mathcal{O}_{i+1} \mid \mathcal{O}_{1:i}; \mathcal{I}^t; \mathcal{I}^v; \mathcal{P}\right), \quad (8)$$

where θ represents the learnable parameters of PerLA.

C. Hilbert-based Serialization and Partition

We choose Hilbert-based serialization for partitioning unordered point clouds for its efficiency and effectiveness in handling large point clouds. While grid partitioning is faster ($O(N)$), the Hilbert curve offers superior spatial coherence and is computationally simpler than KD-trees or octrees, making it particularly beneficial for tasks such as clustering and spatial indexing (more details please refer [42]). Additionally, computing point indices with Hilbert-based serialization is inherently parallelizable, as each point can be processed independently. Sorting and partitioning steps can also benefit from such parallel algorithms.

In the following subsections, we describe the Hilbert-based serialization process in detail, and provide analysis on its computational complexity.

C.1. Algorithm details

We outline the steps for partitioning a point cloud using the Hilbert curve, as detailed in Algorithm 1. This approach leverages the locality-preserving properties of the Hilbert curve to organize and group points into spatially consistent parts. The key steps include: normalizing the point cloud to fit within a unit cube, discretizing the unit cube into a grid, calculating Hilbert indices (by converting grid indices to binary, applying Gray code transformation, and interleaving bits), sorting the points based on their Hilbert indices, and partitioning the point cloud accordingly.

Algorithm 1 Hilbert-based serialization and partition

Require: Point cloud $\mathcal{P}=\{\boldsymbol{p}_i\in\mathbb{R}^3 | i=1,2,\ldots,N\}$, resolution d.
Ensure: Hilbert indices $\mathcal{H} = \{h_i \mid i = 1, 2, \ldots, N\}$.

1: **Normalize the points:** $\boldsymbol{p}_i = \frac{\boldsymbol{p}_i - \boldsymbol{p}_{\min}}{\boldsymbol{p}_{\max} - \boldsymbol{p}_{\min}}, \quad \forall \boldsymbol{p}_i \in \mathcal{P}$
2: **Discretize the unit cube:** $\boldsymbol{p}_i^{\text{grid}} = \lfloor \boldsymbol{p}_i \cdot 2^d \rfloor$ where 2^d defines the resolution of the grid.
3: **Convert grid indices to binary:** Represent each grid index (x, y, z) with d bits.
$t = (b_{t,d-1}, b_{t,d-2}, \ldots, b_{t,0}), \quad t \in \{x, y, z\}$
4: **Transform to Gray code:** Convert binary indices to Gray code to ensure spatial locality: $g_i = b_i \oplus b_{i+1}$, for $i = 0, \ldots, d-2$
5: **Interleave bits (see Algorithm 2):** Interleave the Gray code bits of x, y, and z to form a single integer h_i:
$h_i = \text{Interleave}(g_x, g_y, g_z)$
6: **Apply recursive rotations:** Use the Hilbert curve recursive structure to reorder the interleaved bits, ensuring continuity of the curve.
7: **Output the Hilbert index:** Combine the reordered bits to compute the Hilbert index h_i for each point.
8: **Sort by Hilbert index:** Sort the points $\mathcal{P}$ based on their Hilbert indices $\mathcal{H}$:
$\mathcal{P}_{\text{sorted}} = \text{Sort}(\mathcal{P}, \mathcal{H})$
9: **Partition the point cloud:** Divide the sorted points into L spatially coherent partitions: $\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_L$, where each partition contains approximately N/L points.

Algorithm 2 Interleaving bits, *i.e.*, Interleave$(\cdot)$

Require: Coordinates x, y, $z \in \mathbb{N}$, each represented with d bits.
Ensure: Hilbert index h.

Initialize $h \leftarrow 0$.
for $i = 0$ to $d-1$ **do**
  Extract the i-th bit from x, y, and z by $b_{x,i} = (x \gg i)\&1, b_{y,i} = (y \gg i)\&1, b_{z,i} = (z \gg i)\&1$.
  Interleave the bits into h:
  $h \leftarrow h|(b_{x,i}\ll(3i))|(b_{y,i}\ll(3i+1)) \mid (b_{z,i}\ll(3i+2))$.
end for
return h

C.2. Computation efficiency

We analyze the computational efficiency of Hilbert curve partitioning in terms of both time and space complexity:
Time complexity is $O(N \cdot d + N \log N)$ (dominated by sorting for large N) and is influenced by the following key factors: 1) *Mapping Points to Hilbert Curve:* normalizing, binary conversion, Gray code conversion, and bit interleaving take $O(N \cdot d)$, where N is the number of points and d is the resolution in bits. 2) *Sorting:* sorting the Hilbert indices requires $O(N \log N)$ using efficient sorting algorithms. 3) *Partitioning:* dividing the sorted points into k parts is $O(N)$.

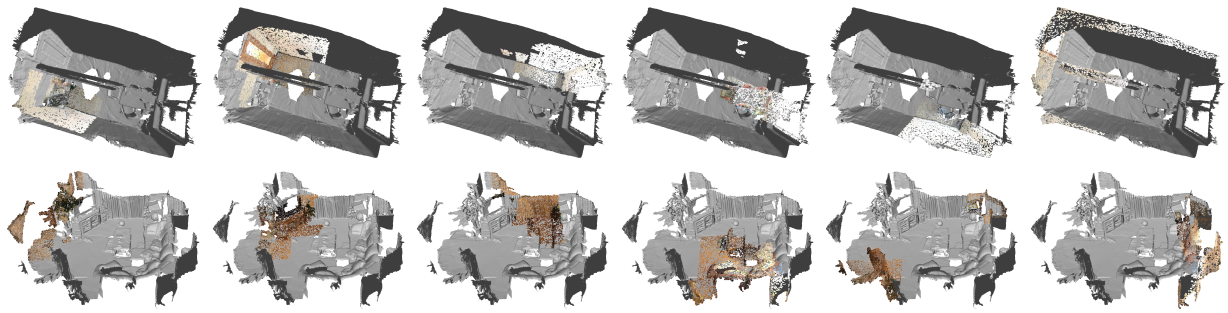

Figure E. Visualization of two qualitative examples demonstrating scene partitioning using Hilbert-based serialization. The images illustrate the stepwise refinement of point cloud partitions, with each row corresponding to a different scene example. From left to right, the partitions (highlighted with brownish color) evolve as the serialization method groups spatially adjacent points.

Table G. Computational analysis among 3D-LLM [21], LL3DA [9], and PerLA.

Cost	3D-LLM [21]	LL3DA [9]	PerLA
Hard Drive (GB)↓	74563.49	5.92	41.58
Time per Scene (s)↓	48203.14	6.24	13.79

Space complexity is $O(N \cdot d)$, determined by the following factors: 1) storing points and Hilbert indices requires $O(N)$, and 2) binary representations and intermediate data take $O(N \cdot d)$.

Tab. G compares the total hard drive usage and processing time of different methods. For 3D-LLM [21], multi-view representations are extracted from ScanNet videos every 20 frames, following the protocol outlined in LL3DA [9]. The results demonstrate the efficiency of PerLA, achieving a good balance between storage requirements and processing time, significantly outperforming 3D-LLM and comparable to LL3DA in computational cost.

Hilbert-based nearest-neighbor search. Our k-NN search search is constrained by geometric segments. We first serialize the union of the global and local point clouds $\mathcal{P}^g \cup \mathcal{P}^l$, using a Hilbert curve ordering. This ordering maps each point's multi-dimensional coordinates to a one-dimensional Hilbert bit d, thereby preserving spatial locality. Next, we apply geometric partitioning [27] on the original point clouds to generate geometric labels $\mathcal{Y}^g$ and $\mathcal{Y}^l$. For each label, we compute its center (*i.e.*, the mean coordinate of all points in that label) and incorporate this center information into the serialized index as high-order bits via bit shifting. In practice, this is achieved by computing a combined metric for each point:

$$\text{combined} = \texttt{label\_offset} \times \text{label} + d,$$

where `label_offset` is chosen to be larger than the maximum possible Hilbert distance. Once the union is sorted per batch using this combined index, we perform an approximate k-NN search.

C.3. Qualitative examples with partitioning.

In Fig. E, we provide two qualitative examples of point cloud partitioning, achieved through Hilbert-based serialization. This approach groups spatially adjacent points into partitions that preserve locality, effectively encoding spatial relationships within the 3D scenes. The top row depicts the progressive partitioning of the first scene, while the bottom row shows the same process for a second scene. From left to right, the images demonstrate how the Hilbert-based method refines partitions, capturing the hierarchical structure of the point cloud. These examples highlight the ability of Hilbert-based serialization to produce coherent partitions that align with the underlying spatial organization of the scenes.

D. Dataset Details

During the training phase, we leverage the training set of the ScanNet portion from the 3DLLM dataset [21]. Additionally, we incorporate data from complementary datasets, including ScanQA [2], ScanRefer [6], and Nr3D [1]. The dataset details are provided below.

3D-LLM dataset [21] comprises: i) 1,033 textual descriptions across 517 scenes, ii) 1,864 lines of embodied task planning spanning 510 scenes, and iii) 2,955 lines of multi-turn embodied dialogues across 517 scenes.

ScanQA dataset [2] is a 3D question-answering benchmark built on top of the ScanRefer [6] dataset, designed to evaluate the ability of models to understand 3D scenes through natural language queries. The dataset contains 6,857 unique questions paired with 30,769 answers spanning 806 reconstructed indoor environments from ScanNet [14]. Each question focuses on objects within the scene, addressing a variety of topics such as object attributes, spatial relationships, and scene semantics. On average, each scene contains 8.5 questions, encouraging models to reason about object-level details and contextual relationships within complex 3D environments.

ScanRefer dataset [6] is a 3D language grounding benchmark built on the ScanNet [14] dataset, consisting of 1,613 RGB-D scans across 806 unique indoor environments. The dataset provides natural language descriptions for objects in reconstructed 3D scenes, with a total of 51,583 descriptions covering 800 ScanNet scenes. Each object is annotated with an average of 4.67 descriptions, ensuring comprehensive linguistic diversity. On average, each scene contains 13.81 objects and 64.48 descriptions, spanning over 250 types of common indoor objects. Among these, 41,034 descriptions explicitly mention object attributes such as color, shape, size, and spatial relationships, making the dataset a rich resource for evaluating fine-grained language grounding in complex 3D environments.

Nr3D dataset [1] is a benchmark for 3D object localization tasks in natural language, built on the ScanNet [14] dataset. It contains 41,503 unique natural language descriptions referring to 5,578 objects across 707 ScanNet scenes. Each description is designed to unambiguously identify a target object in the context of its surrounding scene, incorporating spatial relationships and object attributes such as color, shape, and size. On average, each object is associated with 7.4 descriptions, providing comprehensive linguistic diversity. The dataset focuses on common indoor objects, making it suitable for evaluating fine-grained understanding of object attributes and spatial reasoning in 3D scenes.

E. Additional analysis

E.1. Details with increasing tokens (LL3DA†)

In the main paper (Tab. 3), we introduce a variant of LL3DA†, which combines both global and local information by leveraging query tokens generated from local and global regions. LL3DA† serves as a baseline of enriching global context with local details by extending the number of query tokens. LL3DA† first extracts 3D representations from different partitions of the scene. It then applies Farthest Point Sampling (FPS) to select 1,024 points, along with their corresponding point-level representations, from the union of these partitions. These sampled representations are processed through the multimodal adapter to produce 32 local tokens. Then, we can generate an additional 32 tokens (global tokens) from the point cloud of the entire scene. We finally obtain a total of 64 tokens by concatenating such local and global tokens and processed through a self-attention layer, *i.e.*, $\mathcal{Q}^{64\times768} = f_{att}\left(\text{cat}[\text{MMA}(\mathcal{F}^g, \mathcal{I}^t), \text{MMA}(\mathcal{F}^l, \mathcal{I}^t)]\right)$ (where the number of tokens increases from 32 to 64). The self-attention mechanism enables interaction and information exchange between the 64 tokens, enhancing the representation of both global and local representations. We evaluate the performance of LL3DA† on the ScanQA validation dataset, comparing it with the original LL3DA and our PerLA. As shown in Tab. H, increasing the number of tokens enables LL3DA† to achieve a modest performance gain over the reproduced LL3DA (repr.), as expected. However, its effectiveness remains limited compared to PerLA. This is because PerLA is specifically designed to capture both global context and local details during the scene encoding phase. Furthermore, it is trained with carefully crafted loss functions that ensure stable and efficient learning. Our findings align with Idefics2 [28]. Idefics2 shows that reducing the number of visual tokens through attention-based pooling significantly enhances computational efficiency during training and inference while improving performance on downstream tasks.

Table H. Ablation study on the impact of the increasing the number tokens on the ScanQA [2] validation dataset.

Method	C↑	B4↑	M↑	R↑
LL3DA (repr.)	74.37	13.50	15.09	36.31
LL3DA†	74.54	12.89	15.11	36.96
PerLA	**78.13**	**14.49**	**17.44**	**39.60**

Table I. Ablation study on the impact of the scene-aware prompt encoder on the ScanQA [2] validation dataset.

Method	C↑	B4↑	M↑	R↑
without scene-aware	78.01	14.46	17.32	39.46
with scene-aware	**78.13**	**14.49**	**17.44**	**39.60**

E.2. Scene-aware visual prompt encoder

In the visual prompt encoder, we integrate the detail-enriched scene representations to enhance the prompt representations for clicks and object bounding boxes as explained in Appendix B. To evaluate the efficiency of the scene-aware visual prompt encoder, we conducted experiments on the ScanQA [2] validation dataset. As shown in Tab. I, by integrating scene representations with the visual prompts, while not being a major contributor to the performance, it brings consistent marginal improvements across all evaluation metrics.

E.3. Increase number of tokens

To evaluate the impact of the number of learnable query tokens, we experimented with various numbers of learnable query tokens, from 32 to 128, in the Q-Former (different from LL3DA†). We trained each model *from scratch on the ScanQA training set* and evaluated its performance on the validation set for both LL3DA and PerLA. We observed that increasing the number of tokens to 96 and 128 led to loss divergence (to infinity) in both LL3DA and PerLA. To address this instability, we incorporated mirror descent-based regularization [58]. Tab. J presents the results. Both LL3DA and PerLA achieve better performance with increasing query tokens, where the performance gain tends to saturate from

Table J. Ablation study on the impact of the numbers of learnable query tokens on the ScanQA [2] validation dataset.

Method	32				64				96				128			
Metric	C↑	B4↑	M↑	R↑	C↑	B4↑	M↑	R↑	C↑	B4↑	M↑	R↑	C↑	B4↑	M↑	R↑
LL3DA	73.2	12.8	14.9	36.0	73.7	13.6	15.2	35.5	74.2	13.8	15.1	36.1	74.6	13.5	15.2	36.0
PerLA	74.4	13.7	15.8	36.5	75.1	14.2	15.9	36.6	76.4	14.3	16.3	40.0	77.0	14.3	16.4	38.2

96 to 128 tokens. Moreover, our proposed method PerLA consistently outperforms LL3DA across all token configurations and evaluation metrics.

E.4. Performance for two stage training

As in prior works [9, 21], PerLA uses a two-stage training strategy, where the model is pre-trained on an ensemble dataset comprising diverse 3D tasks. This ensemble dataset allows the model to develop a broad understanding of various 3D scenarios, building the base as a 3D *generalist* model, such as scene description, dense captioning, and question answering. Then, instruction-following fine-tuning is further applied to the generalist model to further enhance its performance on specialized downstream tasks, such as 3D dense captioning and 3D question answering, using task-specific datasets.

We evaluate the performance of PerLA under this two-stage training paradigm, as summarized in Tab. K, together with several baseline methods. The first three rows in Tab. K display the performance of models trained from scratch as task-specific experts. The next three rows show the results of models fine-tuned on individual tasks, initialized from the generalist model's weights. The final row reports the performance of the generalist model without fine-tuning, where a single set of weights is used to handle all tasks.

Our generalist model (the last row without fine-tuning) demonstrates strong task differentiation capabilities, excelling in tasks such as 3D dense captioning and 3D question answering when provided with appropriate textual instructions and visual prompts. For instance, the fine-tuned model achieves notable improvements on ScanRefer (69.41 C@0.5) and ScanQA (78.13 CiDEr), showcasing its ability to leverage the generalist pre-training to boost downstream task performances. However, the generalist model exhibits relatively lower performance on Nr3D compared to ScanRefer, likely because these datasets address the same dense captioning task, and explicit differentiation between them was not included during training. Despite this, the model achieves higher scores on ScanRefer (69.41 C@0.5), potentially indicating a preference for dataset-specific characteristics or structural differences between the datasets. Importantly, the generalist weights serve as a strong initialization for fine-tuning specific tasks. For example, the fine-tuned model on ScanRefer reaches 69.41 C@0.5, significantly outperforming the model trained from scratch (63.02 C@0.5). This highlights the advantages of pretraining as a generalist in boosting task-specific performance. Overall, the results demonstrate that such two-stage training approach enables effective multi-task learning while maintaining robust performance across individual tasks, even when faced with diverse datasets and objectives.

E.5. More results on Scene Description, Embodied Dialogue and Embodied Planning

To provide a comprehensive evaluation, we present additional results on the tasks of scene description, embodied dialogue, and embodied planning using the ScanNet subset of the 3D-LLM dataset [21], which has been used as part of our pre-training dataset. Consistent with the dataset split defined prior work [9], scenes with IDs less than 600 are used for training, while the remaining scenes are reserved for validation. We evaluate these tasks using the same metrics as in the main paper: BLEU-n [43] (1-4), CiDER [55], METEOR [3], and Rouge-L [34].

As shown in Tab. L, for scene description, PerLA outperforms all baselines, including LL3DA, across all metrics. Notably, it achieves substantial improvements in CiDER (+3.07) and METEOR (+1.54) compared to LL3DA.

For embodied dialogue, while LL3DA has already scored quite competitive performance, PerLA achieves further improvements across all metrics, with significant margins in BLEU-4 (+1.34), CiDER (+2.66), and METEOR (+6.05). In embodied planning, PerLA achieves the highest scores across all metrics, surpassing LL3DA, with notable improvements in CiDER (+16.93) and METEOR (+6.41)

These results affirm that PerLA, by enhancing the capability of 3DLAs in perceiving 3D scene details, can consistently improve performance on various downstream tasks, being beneficial to not only perception tasks but also tasks in general embodied context, such as planning.

E.6. Qualitative results

We provide additional qualitative visualization results (Fig. F) on two 3D scene understanding tasks: 3D question answering and 3D dense captioning. The visualization includes (a) question answering on ScanQA [2], (b) dense captioning on ScanRefer [6], and (c) dense captioning on Nr3D [1]. These examples underscore the effectiveness of PerLA in producing more accurate responses when addressing questions specific to a given 3D scene.

For the 3D question aswering task on ScanQA, PerLA is able to correctly interpret natural language questions and respond with accurate answers that are grounded in the 3D scene. For example, when asked, "What color is the chair

Table K. Performance for Two-Stage Training. The first three rows report the performance of PerLA trained from scratch as task-specific experts on their respective training datasets. The subsequent three rows present the results of models fine-tuned on each dataset using weights initialized from the generalist model trained across all task-specific datasets. The final row evaluates the generalist model's performance without fine-tuning. ScanRefer [6] and Nr3D [1] are evaluated for dense captioning, while ScanQA [2] is evaluated for question answering. The results demonstrate the effectiveness of our generalist model in multi-task scenarios and its strong performance after fine-tuning.

Method	ScanRefer@0.5				Nr3D@0.5				ScanQA			
	C↑	B4↑	M↑	R↑	C↑	B4↑	M↑	R↑	C↑	B4↑	M↑	R↑
ScanRefer (scratch)	63.02	35.02	25.61	54.09	-	-	-	-	-	-	-	-
Nr3D (scratch)	-	-	-	-	48.37	28.36	25.72	55.97	-	-	-	-
ScanQA (scratch)	-	-	-	-	-	-	-	-	74.44	13.69	15.78	36.49
ScanRefer (fine-tuned)	**69.41**	**38.02**	**29.07**	**56.80**	-	-	-	-	-	-	-	-
Nr3D (fine-tuned)	-	-	-	-	**55.06**	**31.24**	**28.52**	**59.13**	-	-	-	-
ScanQA (fine-tuned)	-	-	-	-	-	-	-	-	**78.13**	**14.49**	**17.44**	**39.60**
w/o fine-tuning	66.10	37.03	27.33	54.92	51.57	28.40	26.24	56.32	76.88	14.05	16.07	37.97

Table L. Quantitative Comparisons on Scene Description, Embodied Dialogue, and Embodied Planning on the ScanNet part of 3D-LLM [21] with a beam size = 4 for beam search.

Task	Method	BLEU-1↑	BLEU-2↑	BLEU-3↑	BLEU-4↑	CiDER↑	METEOR↑	Rouge-L↑
Scene Description	OPT-1.3B [66]	15.79	6.10	2.07	0.84	0.00	8.40	11.70
	OPT-2.7B [66]	19.97	7.59	3.62	1.13	0.00	6.60	12.32
	OPT-6.7B [66]	24.40	9.93	3.64	1.13	0.06	8.99	16.96
	LLAMA-7B [54]	19.26	7.69	2.79	0.92	0.20	7.00	12.31
	LL3DA [9]	29.94	21.56	14.93	10.02	1.32	12.31	27.08
	PerLA	**31.29**	**23.67**	**16.23**	**12.14**	**4.39**	**13.85**	**28.79**
Embodied Dialogue	OPT-1.3B [66]	2.44	1.05	0.46	0.23	0.31	5.62	4.83
	OPT-2.7B [66]	3.88	1.56	0.73	0.39	0.38	7.38	6.28
	OPT-6.7B [66]	3.59	1.65	0.81	0.43	0.25	6.88	6.16
	LLAMA-7B [54]	4.08	1.80	0.90	0.50	0.27	7.81	6.68
	LL3DA [9]	48.14	39.83	34.83	31.32	260.07	27.21	47.69
	PerLA	**49.91**	**41.10**	**36.06**	**32.66**	**262.73**	**33.26**	**48.24**
Embodied Planning	OPT-1.3B [66]	1.26	0.59	0.26	0.13	0.16	0.24	3.56
	OPT-2.7B [66]	2.02	0.99	0.49	0.26	0.10	3.59	4.35
	OPT-6.7B [66]	2.03	1.06	0.53	0.28	0.00	3.65	3.94
	LLAMA-7B [54]	2.24	1.13	0.55	0.29	0.04	3.53	4.71
	LL3DA [9]	45.07	33.04	24.96	19.15	196.78	19.87	45.58
	PerLA	**48.96**	**36.19**	**27.82**	**22.42**	**213.71**	**26.28**	**47.57**

near the desk?" or "How many doors are in the room?", PerLA can accurately identify relevant objects and their attributes. While in contrast, baseline models like LL3DA [9] struggle with questions requiring multi-step reasoning or fine-grained scene comprehension, often producing incomplete or incorrect answers.

For the 3D dense captioning task, PerLA also demonstrates exceptional descriptive capabilities, accurately capturing object attributes (e.g., "the rectangular brown desk" and "the round table in the center of the room") and spatial relationships. In comparison, LL3DA [9] might produce incomplete or inaccurate descriptions. PerLA 's ability to provide accurate output is particularly evident in challenging cases, such as "There is a rectangular whiteboard. It is on the wall." in ScanRefer, and small or partially reconstructed objects, such as "The plant on the desk next to the window." in Nr3D. Moreover, in the example of "The door that is open." and "The backpack closest to the bed." in Nr3D, PerLA showcases its advanced fine-grained spatial understanding of the object. generating outputs such as "the door that is open" and "the backpack closest to the bed." These examples highlight its superior understanding of both object attributes and spatial relationships, further distinguishing PerLA as a more percepti solution for 3D scene understanding tasks.

Figure F. Qualitative results for 3D scene understanding tasks, including (a) question answering on ScanQA [2], (b,c) dense captioning on ScanRefer [6], and (d,e) dense captioning on Nr3D [1]. On ScanQA, PerLA successfully identifies and reasons about objects and their relationships in the scene. For example, when asked, "What is the color of the sofa chair next to a queen-sized bed?", PerLA accurately answers by localizing the relevant chair and determining its color. Similarly, for complex spatial queries like "Which item is to the left of the bookshelf? PerLA shows a clear understanding of spatial relationships, providing correct and concise answers. For ScanRefer, PerLA demonstrates robust descriptive capabilities by capturing object attributes (e.g., "the rectangular brown desk" and "the round table in the center of the room") and spatial relationships. Compared to LL3DA [9], which often generates incomplete or erroneous descriptions, PerLA excels in producing detailed and accurate outputs. Similarly, on Nr3D, PerLA showcases fine-grained spatial reasoning, with outputs like "the door that is open" and "the backpack closest to the bed," emphasizing its superior understanding of object attributes and spatial relationships.