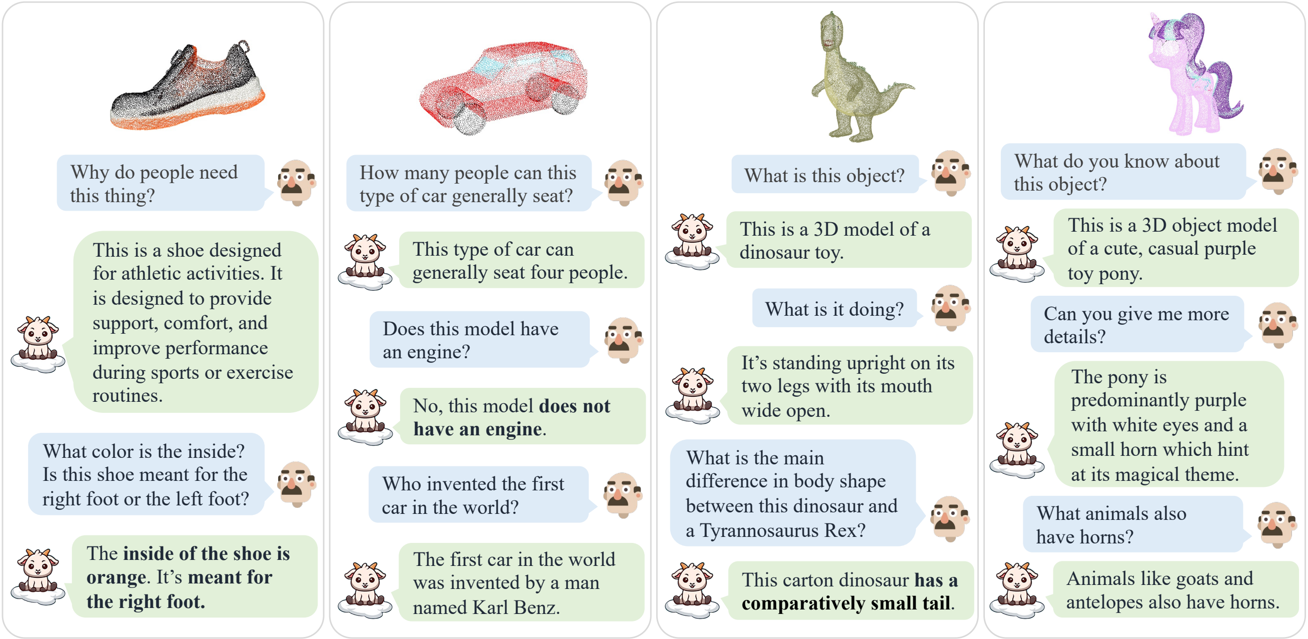
PointLLM: Empowering Large Language Models to Understand Point Clouds

Runsen Xu[1,2] Xiaolong Wang[3] Tai Wang[2] Yilun Chen[2] Jiangmiao Pang[2]✉ Dahua Lin[1,2]
[1]The Chinese University of Hong Kong [2]Shanghai AI Laboratory [3]Zhejiang University
{runsenxu,dhlin}@ie.cuhk.edu.hk, xlking@zju.edu.cn
{wangtai,chenyilun,pangjiangmiao}@pjlab.org.cn



Why do people need this thing?

This is a shoe designed for athletic activities. It is designed to provide support, comfort, and improve performance during sports or exercise routines.

What color is the inside? Is this shoe meant for the right foot or the left foot?

The **inside of the shoe is orange**. It's **meant for the right foot.**

How many people can this type of car generally seat?

This type of car can generally seat four people.

Does this model have an engine?

No, this model **does not have an engine**.

Who invented the first car in the world?

The first car in the world was invented by a man named Karl Benz.

What is this object?

This is a 3D model of a dinosaur toy.

What is it doing?

It's standing upright on its two legs with its mouth wide open.

What is the main difference in body shape between this dinosaur and a Tyrannosaurus Rex?

This carton dinosaur **has a comparatively small tail**.

What do you know about this object?

This is a 3D object model of a cute, casual purple toy pony.

Can you give me more details?

The pony is predominantly purple with white eyes and a small horn which hint at its magical theme.

What animals also have horns?

Animals like goats and antelopes also have horns.

Figure 1. **Demonstrations of PointLLM.** We introduce PointLLM, a multimodal large language model capable of understanding colored point clouds of objects. It perceives object types, geometric structures, and appearance without concerns for ambiguous depth, occlusion, or viewpoint dependency. Through this understanding and existing prior knowledge, PointLLM responds to user instructions accurately, as evidenced by its performance on these unseen samples. To illustrate the advantages of point clouds, we emphasize key points in our model's responses, which are typically challenging to acquire from images, due to the aforementioned problems.

Abstract

The unprecedented advancements in Large Language Models (LLMs) have created a profound impact on natural language processing but are yet to fully embrace the realm of 3D understanding. This paper introduces PointLLM, a preliminary effort to fill this gap, thereby enabling LLMs to understand point clouds and offering a new avenue beyond 2D visual data. PointLLM processes colored object point clouds with human instructions and generates contextually appropriate responses, illustrating its grasp of point clouds and common sense. Specifically, it leverages a point cloud encoder with a powerful LLM to effectively fuse geometric, appearance, and linguistic information. We collect a novel dataset comprising 660K simple and 70K complex point-text instruction pairs to enable a two-stage training strategy: initially aligning latent spaces and subsequently instruction-tuning the unified model. To rigorously evaluate our model's perceptual abilities and its generalization capabilities, we establish two benchmarks: Generative 3D Object Classification and 3D Object Captioning, assessed through three different methods, including

---

✉ Corresponding author.

human evaluation, GPT-4/ChatGPT evaluation, and traditional metrics. Experiment results show that PointLLM demonstrates superior performance over existing 2D baselines. Remarkably, in human-evaluated object captioning tasks, PointLLM outperforms human annotators in over 50% of the samples. Codes, datasets, and benchmarks are available at https://github.com/OpenRobotLab/PointLLM.

1. Introduction

Recent years have witnessed the emergence of large language models (LLMs) [4, 6, 31–33, 38, 43, 44], demonstrating awe-inspiring abilities in natural language processing. These models have become versatile tools, acting as generalized interfaces [15] to perform an array of complex tasks [4, 38]. However, the mastery over text-based tasks is just one aspect of what LLMs can achieve. A new horizon emerges as researchers begin to explore multi-modal LLMs, capable of processing various forms of data such as audio [17] and images [1, 18, 24, 27, 32, 54, 56].

The next step in this evolution lies in understanding the 3D structures. Imagine a scenario where one can interactively create and edit 3D content through simple verbal commands [22, 30], bypassing the need for specialized software, or can instruct a robot to manipulate objects using natural language [10]. These applications require LLMs with a nuanced and accurate understanding of 3D structures.

While existing efforts to integrate LLMs with 2D images provide a pathway to understanding 3D [7, 27, 53], they face difficulties such as ambiguous depth estimation, occlusion, and viewpoint-dependent perception. To address these issues, options like carefully selecting suitable views or employing multi-view images exist. However, suitable views may be elusive due to arbitrary orientations of objects, and multi-view images can lead to increased model overhead and complexity. On the other hand, point clouds, as a universal and efficient representation of 3D, provide a compelling solution. They offer direct access to geometric and appearance, fostering a comprehensive understanding of 3D shapes, resilient handling of occlusion, and view-invariant analysis. Yet, despite their advantages, the coupling of point clouds with LLMs remains underexplored.

In this work, we pave the way to empower large language models to understand point clouds, with a preliminary focus on 3D objects. Specifically, we present PointLLM, which accepts colored object point clouds with human instructions and generates accurate responses, reflecting its understanding of point clouds and common sense, as illustrated in Fig. 1. Enhancing LLMs' understanding of 3D object point clouds presents three key difficulties: the absence of training data, the necessity of building a suitable model architecture, and the lack of comprehensive benchmarks and evaluation methods, each of which we address as follows.

Data collection. We collect a large-scale point-text instruction following dataset, containing 660K brief-description instructions for 660K object point clouds, and 70K complex instructions for 15K object point clouds. The training data that guides the model in extracting meaningful representations from point clouds and responding to user instructions are especially rare in the context of object point clouds, and manual collection can be both time-consuming and expensive. To circumvent this issue, we utilize the recently introduced Cap3D [29], a large-scale 3D object captioning dataset built upon Objaverse [9]. Employing GPT-4's [32] reasoning abilities and its world model, we prompt GPT-4 to generate varied instruction following data based on the contexts provided by the captions.

Model and training. We introduce PointLLM, which employs a pre-trained point cloud encoder for encoding point clouds into tokens and utilizes a powerful pre-trained large language model for reasoning and generating context-appropriate responses. Our training features a two-stage strategy [27]: initial alignment of the latent spaces between the point cloud encoder and the large language model, followed by instruction-tuning the unified model. This methodology ensures an effective fusion of both geometric and appearance information from 3D point clouds with the linguistic capabilities of the language model.

Benchmarks and Evaluation. We establish two distinct benchmarks: Generative 3D Object Classification and 3D Object Captioning, accompanied by a diverse evaluation framework, to assess the model's understanding of point clouds. Due to the generative nature of the model's outputs, we format the classification task in a generative manner, where the model is prompted to directly output the object type. Our model engages in object classification through close-set zero-shot classification on ModelNet40 [48] and open-vocabulary classification on Objaverse [9], along with Objaverse-based captioning. As defining a single evaluation metric for generative tasks is difficult, we employ three types of evaluation methods, including human evaluation, GPT-4/ChatGPT [31] evaluation, and traditional metric [3, 12, 26, 34, 39] evaluation to rigorously assess our model's perceptual and generalization capabilities.

Experiment results indicate that our PointLLM demonstrates substantially better performance over 2D baselines, and in over 50% of tested samples of object captioning, it gains higher scores than human annotators in human evaluation. To supplement these quantitative evaluations, we also present a range of qualitative examples, offering a broader perspective on our model's real-world performance.

2. Related Work

Multi-modal large language models. Multi-modal Large Language Models (MLLMs) are designed to comprehend and interpret a wide range of information that extends beyond mere text-based data [51]. They aim at having the capability to interpret diverse modalities, including but not limited to images [13, 18, 27, 45, 56], audio [17], motion [21], etc. They can generate contextually relevant free-form text responses under zero-shot and few-shot scenarios. Broadly, the models can be classified into two categories.

The first category includes models that employ a large language model to interface with individual, modality-specific models or APIs [14, 17, 35, 42, 47]. This approach circumvents the need for additional model training but is heavily dependent on the availability and capabilities of pre-existing models or APIs.

The second category pertains to models that employ an end-to-end training strategy. There are two prominent paradigms within this category. The first involves training the model from scratch, similar to text-only LLMs, using large-scale multi-modal corpora and datasets [18, 36]. The second paradigm builds on pre-trained LLMs and unimodal encoders, thereby avoiding training from scratch [1, 2, 7, 10, 11, 13, 23, 24, 27, 40, 54–56]. This strategy typically involves a two-stage process: initial alignment of the unimodal encoder with the LLM's feature space, followed by instruction-based fine-tuning.

In our work, we adhere to the alignment and tuning strategy, with the goal of constructing an MLLM capable of understanding 3D object point clouds.

Object point cloud understanding with language. The emergence of models like CLIP [37], which bridges visual and textual modalities, has inspired similar efforts in the 3D object domain [19, 53, 57]. For instance, PointCLIP [53] leverages depth image projections of point clouds for 3D recognition tasks with pre-trained 2D CLIP models. ULIP [49] and ULIP-2 [50] take a more direct approach by training a point cloud encoder to align with CLIP representations using point cloud, image, and text triplets. Particularly, ULIP-2 enhances model performance through an automated data generation pipeline that captions rendered images of point clouds with an image captioning model [24, 25]. Recent endeavors like Cap3D [29] and UniG3D [41] also use a similar approach to generate point-text datasets to further facilitate object point cloud understanding and generation. Concurrent with our work, 3D-LLM [16] seeks to enable large language models to comprehend the 3D world. Unlike our approach, which focuses on directly understanding 3D point clouds, 3D-LLM employs multi-view images as input, relying on pre-trained 2D foundation models to extract features, thereby not engaging directly with the 3D data.

In our work, we adopt the strategy from ULIP-2 for pre-training our point cloud encoder and use the Cap3D dataset for data collection. Our generative model aims to provide a direct and comprehensive understanding of object point clouds, supporting open-ended and free-form interactions, rather than focusing on conventional discriminative tasks.

Table 1. **Instruction Following Template.** {System Prompt} is the system prompt used by the pre-trained LLM, {p_tokens} are point tokens, and {Instruction} and {Response} denote user instructions and model responses. Losses are computed only for model responses and the end-of-sentence token </s>.

{System Prompt}	
USER: ASSISTANT:	<p_start>{p_tokens}<p_end>{Instruction 1} {Response 1}</s>
USER: ASSISTANT:	{Instruction 2} {Response 2}</s>
USER: ASSISTANT:	{Instruction 3} {Response 3}</s>

3. Methodology

This section elucidates our strategy for the automatic generation of point-text instruction-following data. We then delve into the architecture of our model, PointLLM, which inputs an object point cloud and user instruction and outputs corresponding responses. Lastly, we detail our loss function and two-stage training strategy.

3.1. Point-Text Instruction Following Data

The daunting challenge in the development of an end-to-end multi-modal LLM is procuring large-scale multi-modal instruction-following data, vital for representation learning, aligning latent spaces, and orienting the model to adhere to human intentions [1, 7, 25, 27, 56]. However, manual labeling of such data is cost-prohibitive and labor-intensive. To overcome this, we propose an automated data generation technique utilizing the large-scale point cloud captioning dataset, Cap3D [29], with the assistance of GPT-4 [32]. The generated dataset adheres to a uniform instruction following template, shown in Tab. 1, and consists of brief-description instructions and complex instructions, which aid in latent space alignment and instruction tuning, respectively.

Brief-description instructions. The Cap3D [29] dataset provides two variations of captions for the 3D objects in Objaverse [9]: those generated by image-captioning models and those annotated by humans. While there are 660K objects accompanied by generated captions, only 40K samples have human-annotated captions. For brief-description instruction, we utilize the model-generated split due to the need for a larger data volume for aligning the latent spaces of point cloud and text modalities [27]. We created a list of 30 instructions to instruct the model to provide a succinct description of a given 3D object point cloud. A random instruction from this list is chosen as the user instruction, and the caption from Cap3D is used directly as the model response, forming a single-round instruction following sample. This results in 660K brief-description instruction data, each corresponding to a unique object point cloud.

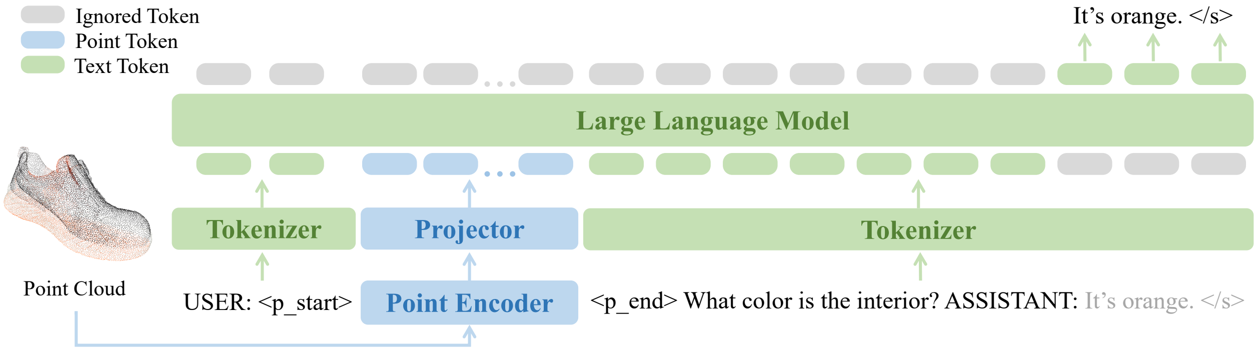


Figure 2. **An overview of PointLLM.** The point encoder extracts features from the input point cloud and projects them to the latent space of the LLM backbone. The LLM backbone processes sequences of point tokens and text tokens, and generates the predicted tokens as the output. The model is trained with a cross-entropy loss that is only computed on the tokens corresponding to the model responses.

Complex instructions. Beyond brief descriptions, it's crucial that the model learns to understand objects from a variety of angles, responding accurately to diverse human instructions. To facilitate this, we employ GPT-4 to produce complex instruction-following data. Specifically, a caption from Cap3D is used to stimulate GPT-4 into crafting a more comprehensive description that identifies the object's type, appearance, functionalities, and any other inferable information. Similar to the process for generating brief-description instructions, we also curate a set of 30 distinct prompts, each pushing the model to describe the 3D object in depth. One of these prompts is randomly coupled with the newly crafted description, forming a training sample. GPT-4 is further used to generate conversations (*i.e.*, Q&A pairs) that delve into diverse aspects of the object based on the captions. For example, questions might probe the object's functionality or the materials it's made from, and the corresponding answers should be informative and comprehensive. For each object, GPT-4 generates 3 single-round conversations and 1 multi-round conversation with 3 Q&A pairs, all ensuring logical relevance.

Prioritizing data quality, we select 15K captions from the Cap3D human-annotated split for data generation, each with captions of more than five words. After filtering incorrect GPT-4 outputs, we generate 70K complex instruction samples, including 15K detailed descriptions, 40K single-round conversations, and 15K multi-round conversations. The instruction lists, GPT-4 system prompt, and a data generation example can be found in Appendix A.

3.2. Model Architecture

As shown in Fig. 2, our PointLLM is a generative model that aims to complete multi-modal sentences that contain both point clouds and texts. The model consists of three main components: a pre-trained point cloud encoder f_{pe}, a linear projector f_{proj}, and a pre-trained large language model (LLM) backbone f_{llm}. The point cloud encoder is responsible for transforming the input point clouds into a sequence of point features that can be processed by the LLM backbone. The liner projector projects the point features into point tokens having the same feature dimensions as text tokens. The LLM backbone is a decoder-only transformer model that predicts the next token in the sentence given the text and point tokens.

The point cloud encoder f_{pe} takes as input a point cloud $P \in \mathbb{R}^{n \times d}$, where n is the number of points and d is the dimension of each point. The output of the encoder is a sequence of point features $X = (x_1, x_2, \dots, x_m) \in \mathbb{R}^{m \times c}$, where m is the number of point features and c is the feature dimension. The point features are further projected into point tokens by the linear projector f_{proj}. The projector is a linear layer that maps the point features X to point tokens $Y = (y_1, y_2, ..., y_m) \in \mathbb{R}^{m \times c'}$, where c' is the dimension of the point tokens, which is the same as the text tokens.

The LLM backbone f_{llm} accepts a sequence of tokens, composed of both text and point tokens. This mixed sequence of tokens is denoted as $Z = (z_1, z_2, ..., z_k) \in \mathbb{R}^{k \times c'}$, where k is the total number of tokens. Utilizing a self-attention mechanism, the LLM backbone is capable of understanding the contextual relationships between different types of tokens, enabling it to generate responses based on both text and point cloud inputs. Formally, the output of the LLM backbone f_{llm} is a sequence of predicted tokens $\hat{Z} = (\hat{z}_1, \hat{z}_2, ..., \hat{z}_k) \in \mathbb{R}^{k \times c'}$. The prediction of the i-th token, $\hat{z}_i$, is conditioned on all previous tokens, $Z_{<i} = (z_1, ..., z_{i-1})$. This can be expressed mathematically as

$$\hat{z}_i = f_{llm}(Z_{<i}). \tag{1}$$

Afterwards, each $\hat{z}_i$ is passed through a final linear layer followed by a softmax operation, mapping the hidden states into a probability distribution over the vocabulary. This additional layer is denoted as $f_{vocab} : \mathbb{R}^{c'} \rightarrow \mathbb{R}^{V}$, where V is the size of the vocabulary. The final prediction $\tilde{z}_i$ for the i-th token is then the word in the vocabulary with the highest probability:

$$\tilde{z}_i = \arg\max_{w \in \text{vocab}} f_{vocab}(\hat{z}_i)[w]. \tag{2}$$

3.3. Training

Loss function. We train PointLLM by minimizing the negative log-likelihood of the text token at each position. Our loss function is only computed on text tokens that constitute the model's responses, including the end-of-sentence token </s>. This restriction excludes the tokens from human instructions, ensuring that the model focuses on learning to generate accurate and coherent responses. The end-to-end nature of this training approach enables PointLLM to effectively integrate point cloud and text modalities, leading to improved performance in instruction-following tasks.

Two-stage training. Our training procedure comprises two stages, each focusing on different aspects of the model.

During the first stage, termed the **feature alignment stage**, we freeze the parameters of the point cloud encoder and the LLM, and train only the linear projector. At this stage, the training process uses brief-description instructions, aiming to align point features with the text token space effectively. This stage also includes the adjustment of token embeddings for the two newly added special tokens <p_start> and <p_end>.

In the second stage, referred to as the **instruction tuning stage**, we freeze the point cloud encoder while jointly training the linear projector and the LLM. This second stage uses complex instructions and helps the model to build its ability to understand and respond to complex instructions including point cloud data.

4. Benchmarks and Evaluation

Evaluating the performance of a multi-modal LLM is challenging, as it's difficult to define a single metric that can capture the quality and diversity of the generated outputs. Moreover, existing benchmarks for 3D object understanding are mostly based on discriminative tasks like close-set classification or retrieval, which do not fully reflect the generative nature and open-vocabulary setting of our model. Therefore, we propose two novel benchmarks to assess our model's perceptual abilities and generalization power: Generative 3D Object Classification and 3D Object Captioning. We adopt various evaluation methods for assessing performances including human evaluation, GPT-4/ChatGPT evaluation, and traditional metric evaluation. We use GPT-4 or ChatGPT as an evaluator, as they demonstrate abilities to align with human judgment accurately. Please refer to Appendix B.1 for the detailed prompts for GPT-4/ChatGPT and Appendix B.2 for the human verification of the GPT evaluators' correctness.

4.1. Generative 3D Object Classification

The task of generative 3D object classification is to prompt the model to generate the object type given its point cloud. We consider two scenarios for this task: close-set zero-shot classification and open-vocabulary classification.

Close-set zero-shot classification. In this scenario, the object type belongs to a fixed set of categories, and the model never sees any samples of this dataset during training. This tests the model's ability to generalize to unseen domains using its prior knowledge. We use the test split of the ModelNet40 [48] dataset as our source of data, which contains point clouds of 40 different object categories.

Initially, we consider formatting this task as a multiple-choice problem, including indexed candidate category names in the prompt, and prompting our model to select one of the 40 categories given the point cloud as input. However, since our model is not designed for multiple-choice problems but for real-world usage where it can generate any word or phrase as output, we cannot directly parse its response for evaluation. Therefore, we use ChatGPT as a post-processor to select one of the ModelNet40 categories based on the model's answer. If ChatGPT selects the correct option, then we consider the model's classification correct; otherwise, we consider it incorrect. In the meantime, we find that including category names in the prompt results in meaningless responses from InstructBLIP [7], which is the model we compare with, making meaningful comparisons challenging. Consequently, we opt for a more generalized prompt, without including the candidate lists in the prompt. This allows us to make balanced comparisons.

Including candidate lists in the prompt, we have also tried to calculate the conditional probability of different options given the model's output following [46], but this method did not work well for our model. As our instruction-following training data lacks such scenarios where it's needed to choose from a fixed set of options, our model always produces very low probabilities on these options with biased results. For example, among the options "00" to "39", our model predicts very low probabilities and among these low probabilities, "00" and "39" are the highest most of the time, which leads to biased predictions. Therefore, we choose to use ChatGPT for post-processing.

Open-vocabulary classification. In this scenario, the object type is not limited to a predefined set of categories, but can be any word or phrase that identifies the object. This reflects the real-world setting where new objects can appear at any time, and the model needs to be able to recognize them without retraining. We use the human-annotated split of the Cap3D [29] dataset as our source of data, which contains point clouds of various objects from the Objaverse [9] dataset with human-annotated captions. We randomly select 200 objects from the split and use the human captions as ground truth labels. We prompt our model with the same prompts used for the close-set zero-shot benchmark with point clouds as input and collect the model's output for each object. Then we use GPT-4 as an evaluator to classify whether the model's response and the human caption are referring to the same object type. We do not require the model's response to match exactly with the human caption, as long as it conveys the same object type. For example, if the human caption is "a blue mug", then "a cup", "a coffee mug", or "a ceramic cup" are all correct predictions. We opt for GPT-4 over ChatGPT in this scenario due to the former's superior ability to identify the same object type. ChatGPT tends to produce more false negatives, meaning that it considers two words or phrases are not referring to the same object type, even when they are, while GPT-4 demonstrates accurate recognition.

4.2. 3D Object Captioning

3D object captioning involves generating a natural language description of an object, given its point cloud representation. This description can range from a simple sentence to a detailed paragraph, encompassing the object's type, attributes, functions, and more. In our evaluation, we utilize the same 200 objects from the Cap3D dataset previously used for the open-vocabulary classification scenario, and prompt our model to caption them. Human-annotated captions corresponding to these objects serve as reference ground truths.

For a comprehensive and robust evaluation, we employ three distinct methods to assess performance in this task:

1. **Human evaluation.** Captions for a given object, derived from various models and the human-annotated reference, are randomly shuffled. Two human evaluators independently assess and score these captions while visually exploring the Objaverse [9] objects using the official Objaverse explorer. Scores are assigned based on the accuracy and completeness of the information described, with one point allocated for each correct piece of information such as category, color, shape, use, material, etc., and partial scores between 0-1 for partially correct points. For example, a description of a "black wheel" would receive two points if the object is a wheel and it is indeed black. Within each group of captions for the same object, the final scores are then adjusted based on the comparative quality of each caption, ensuring a clear distinction between good and bad captions. The scores reported in this paper are the average of the two evaluators' assessments.
2. **GPT-4 evaluation.** Acknowledging that human evaluation is both time-consuming and costly, we also employ GPT-4 as an evaluator. Given a model-generated caption and its corresponding human reference, GPT-4 identifies the aspects mentioned in the human caption and calculates the percentage of these aspects that are either correctly mentioned or partially matched in the model's caption, scoring from 0 to 100.
3. **Traditional metric evaluation.** In addition to the above, we employ traditional metrics such as BLEU-1 [34], ROUGE-L [26], and METEOR [3]. Though widely used, these metrics often fall short in accurately evaluating generative tasks, as they primarily measure the overlap of n-grams or their varieties, and account less for the semantic similarity or diversity of the captions. Therefore, to mitigate this limitation, we incorporate two additional data-driven metrics, Sentence-BERT [39] and SimCSE [12] similarity, which compute the similarity of sentence embeddings between model-generated and human captions.

These diverse evaluation approaches provide a multifaceted perspective on models' understanding of point clouds, shedding light on both the quantitative accuracy and qualitative richness of the generated captions.

5. Experiment Results

5.1. Implementation and Training Details

Implementation details. We use the LLaMA model [44] as our LLM backbone, with the 7B and 13B Vicuna [5] checkpoint as the default settings. Point-BERT [52], pre-trained with ULIP-2 [50] on the Objaverse [9] dataset, serves as our point encoder. ULIP-2 is a method for aligning the latent space of Point-BERT to that of CLIP [37] through contrastive learning, endowing Point-BERT with a strong zero-shot capability for 3D object recognition. As the original implementation of ULIP-2 only supports point clouds with spatial coordinates (xyz), we re-train Point-BERT with color information (xyzrgb), following the same procedure outlined in the ULIP-2 paper. For training Point-BERT, we employ ViT-L/14 from OpenCLIP [20] and use point clouds from the Cap3D [29] dataset, which contains 660K objects. We filter out 3000 objects from this dataset and reserve them for future testing. These 3000 objects are not used during any stage of the entire model training and the 200 objects utilized for our benchmarks are part of these 3000 unseen objects to prevent information leakage. We utilize $n = 8192$ points and $d = 6$ dimensions for each point cloud. We assign a black color to point clouds from ModelNet40, as they lack color information. The point encoder outputs $m = 513$ point features, each with $c = 1152$ dimensions, and the projector maps them to point tokens, each with $c' = 5120$ dimensions (7B model) or $c' = 5120$ dimensions (13B model), which align with the text tokens of LLaMA. After adding two additional special tokens, the vocabulary size of PointLLM is $V = 32003$.

Training details. All experiments are conducted on 8 ×

Table 2. **Generative 3D object classification results on the ModelNet40 test split and Objaverse datasets.** The results show the classification accuracy for different models, under the **I**nstruction-typed (I) prompt “What is this?” and the **C**ompletion-typed (C) prompt “This is an object of ”. Point clouds with RGB are used for PointLLM models, while single-view images are used for InstructBLIP.

Model	Input	ModelNet40 (I)	ModelNet40 (C)	Objaverse (I)	Objaverse (C)
InstructBLIP-7B (Vicuna-7B) [7]	Singe-View Image	19.53	31.48	38.50	38.00
InstructBLIP-13B (Vicuna-13B) [7]	Singe-View Image	25.97	31.40	35.50	35.00
PointLLM-7B (Vicuna-7B)	Point Cloud (+RGB)	41.33	40.84	**47.50**	**44.00**
PointLLM-13B (Vicuna-13B)	Point Cloud (+RGB)	**43.92**	**44.77**	45.00	40.00

Table 3. **3D object captioning results**. Models are evaluated using human evaluation, GPT-4 evaluation, and traditional metrics. A primary focus is placed on human and GPT-4 evaluation, along with data-driven metrics (Sentence-BERT and SimCSE), as conventional measures like BLEU, ROUGE-L, and METEOR may not sufficiently capture the semantic richness and diversity of the generated captions.

Model	Human	GPT-4	Sentence-BERT	SimCSE	BLEU-1	ROUGE-L	METEOR
InstructBLIP-7B (Vicuna-7B) [7]	17.50	32.70	42.42	43.85	11.24	13.93	14.92
InstructBLIP-13B (Vicuna-13B) [7]	17.49	31.15	42.85	44.12	**12.57**	**15.01**	**15.96**
PointLLM-7B (Vicuna-7B)	**37.38**	**39.20**	48.20	**48.67**	8.00	11.10	15.20
PointLLM-13B (Vicuna-13B)	30.64	39.16	**49.12**	48.35	9.65	12.79	15.25

80G A100 GPUs with BF16 data type, leveraging flash-attention [8], the AdamW [28] optimizer, and a cosine learning rate scheduler. For the feature alignment stage, we train our model for 3 epochs with a batch size of 128 and a learning rate of 2e-3. For the instruction tuning stage, we also train our model for 3 epochs, but with a batch size of 32 and a learning rate of 2e-5. For the 13B model, the two stages take about 20 and 5 hours to complete, respectively.

5.2. Generative 3D Object Classification

In Tab. 2, we present the classification accuracy of the models on our proposed generative 3D object classification tasks. This includes the close-set zero-shot classification on ModelNet40 [48] and open-vocabulary classification on Objaverse [9]. We draw comparisons with InstructBLIP [7], a powerful multi-modal LLM capable of receiving a single image and generating textual output. For InstructBLIP’s image inputs, we employ rendered images of ModelNet point clouds and Objaverse objects. We prompt InstructBLIP and our PointLLM with the same prompts of two types: the **I**nstruction-typed (I) prompt “What is this?” and the **C**ompletion-type (C) prompt “This is an object of ”. Additionally, we report the zero-shot performance of our reproduced Point-BERT model with ULIP-2 training.

A glance at Tab. 2 reveals PointLLM’s superiority over InstructBLIP across both ModelNet40 and Objaverse datasets, under both prompt types. This underscores the benefit of engaging directly with point clouds in understanding 3D objects compared to single-view images. Point clouds with color information, capturing 3D geometry and object appearance, can fend off challenges posed by occlusion, distortion, and viewpoint variation. Leveraging a pre-trained point encoder and a large language model backbone, PointLLM efficiently translates point cloud information into natural language, conveying the object’s identity.

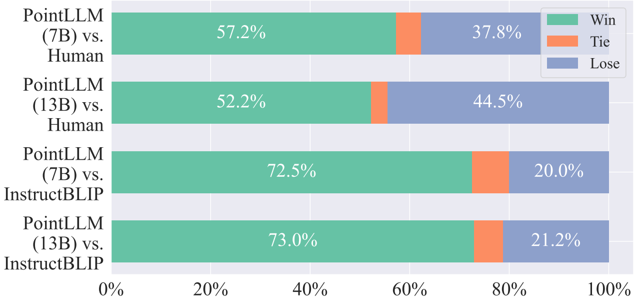


Figure 3. **Win rates comparison between PointLLM and human annotations or InstructBLIP**. Notably, PointLLM outperforms human annotations in more than half of the testing samples and exhibits a substantial advantage over the InstructBLIP models.

The zero-shot performance on ModelNet40 further illustrates our model’s aptitude for generalization. Even though ModelNet40 comprises point clouds unseen during training, PointLLM recognizes them using its pre-existing knowledge and perception abilities honed during our two-stage training. This adaptability to unseen domains and novel objects, without necessitating retraining, speaks to our model’s robustness.

5.3. 3D Object Captioning

In Tab. 3, we present the results of our 3D object captioning benchmark, comprising four distinct models across a range of metrics. All the scores are averaged across ob-

Table 4. **Traditional metrics for different captions**. The table demonstrates the limitations of BLEU-1, ROUGE-L, and METEOR in evaluating captions. The referenced ground truth caption is compared against captions (without modification) from InstructBLIP-13B and PointLLM-13B. These metrics may not accurately reflect the correctness of the generated captions.

Caption	BLEU-1	ROUGE-L	METEOR
Private jet	**100.00**	**100.00**	**100.00**
there is a black jet engine in a dark background	10.00	18.18	17.86
This is a 3D model of a cartoon -style commercial airplane.	0.00	0.00	0.00

jects. We prompt all the models with "Briefly caption this 3D model." The human-evaluated scores are multiplied by 10 to align with the scale of other metrics.

In the evaluation of 3D object captioning, our models demonstrate a substantially enhanced performance over InstructBLIP across various evaluation metrics. This improvement is particularly evident in the human and GPT-4 evaluations, where our models exhibit a greater capacity to provide accurate information about the object. Such evaluations focus on the essence of the object and measure the ability of the models to understand and convey its intricate details. Additionally, the Sentence-BERT and SimCSE results further reinforce our models' capabilities, verifying that they can generate captions that are semantically more similar to the ground truth than InstructBLIP.

We observe that our 13B model does not consistently outperform the 7B model. This suggests the language model's capacity may not be the bottleneck. Rather, the challenge might lie in extracting point cloud information for comprehension by the language model. This emphasizes the importance of effective point cloud representation learning and transformation, which may be more significant than merely increasing the model's size.

We further investigate the human evaluation data to ascertain how our models compare with ground truth human annotations and InstructBLIP. We calculate the win rates within each evaluator and average them. As depicted in Fig. 3, our models' win rates against human annotations and InstructBLIP are particularly enlightening. Both versions of PointLLM manage to outperform human annotations in over half of the testing samples (win rates of 57.25% for 7B and 52.25% for 13B). This reinforces the effectiveness of our approach in capturing and conveying the salient details of 3D objects. Furthermore, PointLLM significantly surpasses InstructBLIP, with win rates of 72.5% for 7B and 73% for 13B, underscoring the model's superior ability in 3D object captioning.

Limitations of traditional metrics. The final aspect of our analysis lies in the limitations of traditional metrics like BLEU-1, ROUGE-L, and METEOR in evaluating the generated captions for 3D objects, as presented in Tab. 4.

In the given example, the referenced ground truth describes a "Private jet," while the second caption from InstructBLIP-13B incorrectly identifies the object as a "jet engine" and the third caption from PointLLM-13B accurately identifies it as an "airplane." Despite the semantic inaccuracy in the second caption, it receives higher scores compared to the third caption, which correctly identifies the object but gains a zero score.

This discrepancy showcases the shortcomings of traditional metrics in capturing semantic similarity and diversity. They primarily measure overlap of n-grams or its varieties, and may fail to account for the essence of the captions, which is critical in tasks like 3D object captioning. Therefore, we primarily rely on more accurate and robust metrics such as human evaluation, GPT-4 evaluation, Sentence-BERT, and SimCSE for this task.

5.4. Qualitative Results

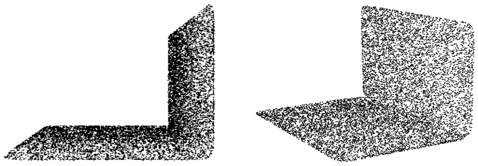
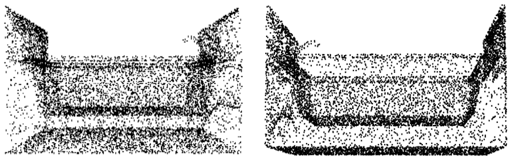
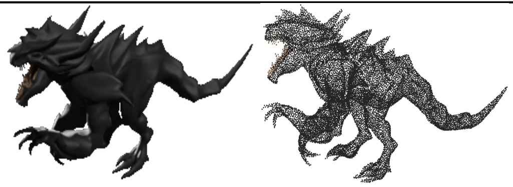
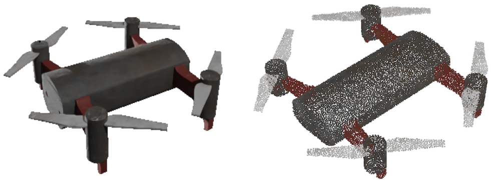
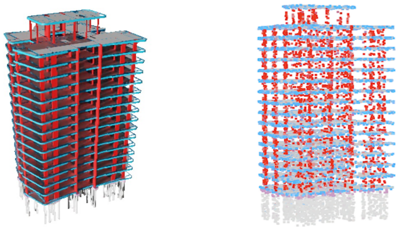
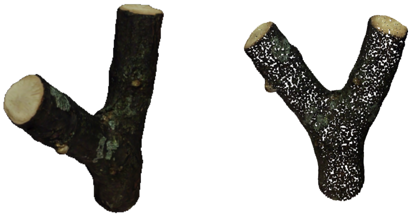
In this section, we present the qualitative results of our PointLLM-13B model, compared with InstructBLIP-13B [7] on our proposed tasks. We also show dialogues between PointLLM-13B and a human user. All samples used in this analysis were unseen during training.

Samples 1 and 2 in Tab. 5 illustrate two typical failure cases of InstructBLIP. Sample 1 highlights InstructBLIP's inability to estimate depth information, leading to the misclassification of a laptop as the letter 'L'. Sample 2 reveals InstructBLIP's struggle with occlusion, resulting in the failure to identify a bathtub. These errors stem from the constraints of single-view image input. It's possible that an appropriate view or multi-view images might mitigate the issue. However, determining the optimal view can be impractical as the object can have arbitrary orientations, and multi-view images will increase model complexity and overhead. In contrast, point clouds bypass these challenges by providing direct access to object geometry without concerns over ambiguous depth, occlusion, or viewpoint dependency.

Samples 3 and 4 further demonstrate InstructBLIP's failure cases on Objaverse in object classification. For object captioning on Objaverse (Samples 5 and 6), our model supplies more precise and detailed descriptions than InstructBLIP, even surpassing human-annotated ground truth.

Fig. 4 showcases dialogues between PointLLM-13B and a human user, which reveal our model's capacity to understand point clouds' shapes, appearances, functionalities, and more. Notably, our model is unaffected by occlusion, capable of discerning the car's internal two-seat structure and identifying a logo on the back of a shoe, tasks challenging for image inputs. Furthermore, our model engages with human instructions using common sense and avoids biases, as seen in its refusal to declare a 'best' shoe brand. Collec-

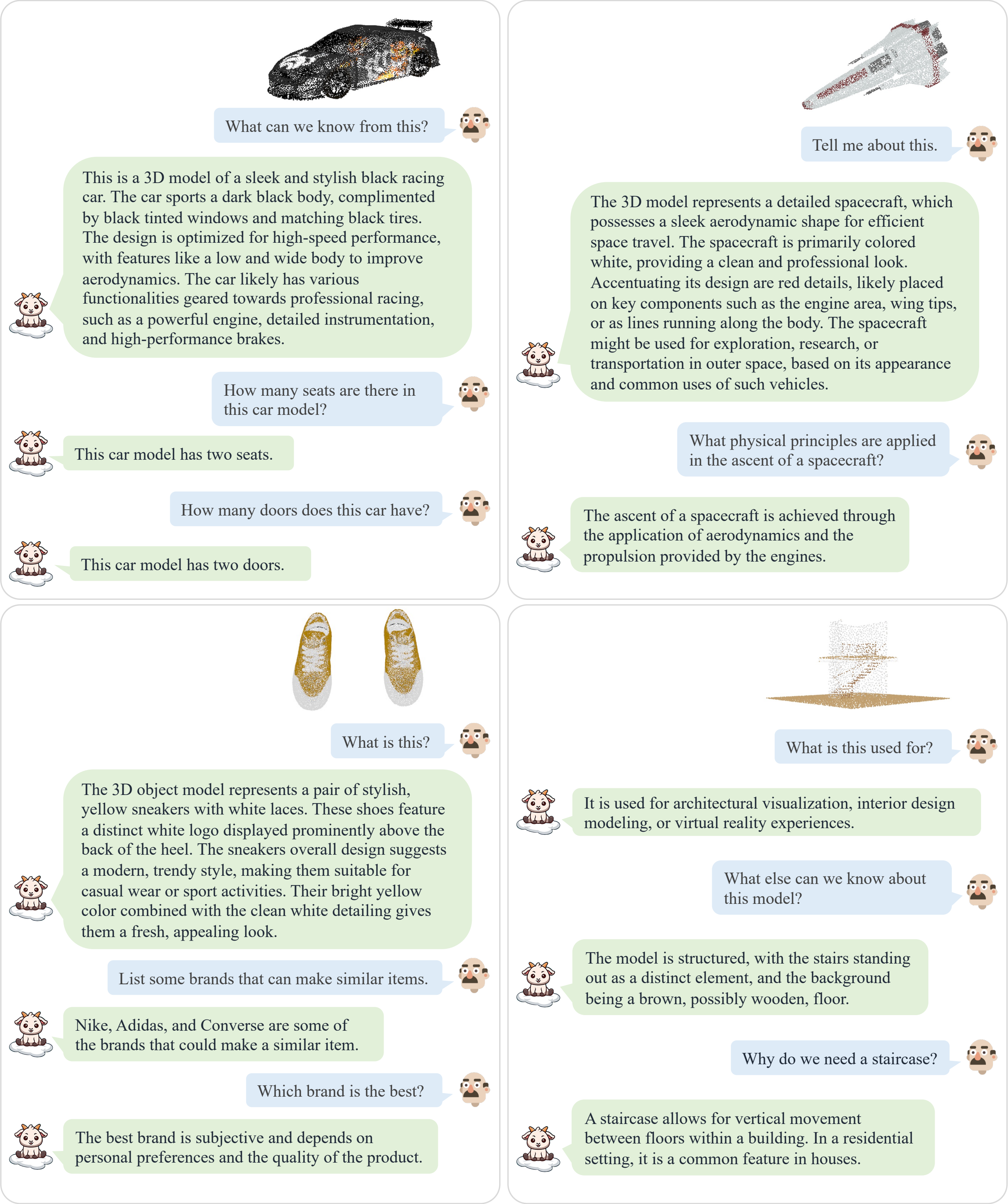
Table 5. **Qualitative comparisons with InstructBLIP and ground truths on our benchmark.** We show the classification and captioning results of both models on ModelNet40 [48] and Objaverse [9], as well as the ground truth. Samples 1-2 and 3-4 show classification on ModelNet40 and Objaverse, respectively. Samples 5-6 show object captioning on Objaverse. The results use prompts specified in Sec. 5. The first image of each sample is the input of InstructBLIP and we also show point clouds from other views for reference. These samples show our PointLLM produces more accurate and detailed results than image-based InstructBLIP and even human-annotated ground truths.

Samples 1,2		
Ground Truth	Laptop	Bathtub
User	What is this?	This is an object of
InstructBLIP-13B	symbol letter l	3d image of a tunnel
PointLLM-13B	The 3D model represents a notebook computer, typically a laptop.	The 3D object model depicts a modernistic grey bathtub.
Samples 3,4		
Uid	e85ebb729b02402bbe3b917e1196f8d3	252f3b3f5cd64698826fc1ab42614677
Ground Truth	A cartoon black monster like a dragon.	A drone with grey body and four red arms.
User	What is this?	This is an object of
InstructBLIP-13B	black panther	3d object
PointLLM-13B	The 3D model depicts a dark gray, animated toy dragon with a menacing appearance.	This is a 3D model of a cartoon-style quadcopter drone.
Samples 5,6		
Uid	69865c89fc7344be8ed5c1a54dbddc20	97367c4740f64935b7a5e34ae1398035
Ground Truth	A storyed building.	A cartoon look like a tree in 3d.
User	Briefly caption this 3D model.	Briefly caption this 3D model.
InstructBLIP-13B	a high resolution image of a construction of a tower with a bright blue color	a stick is attached to the back of a tree
PointLLM-13B	The 3D object is a tall, multicolored building model with blue and red details.	The 3D model represents a green tree that's been cut at the base, with a distinctive white ring marking where the cut occurred.

tively, these samples validate PointLLM-13B's proficiency in understanding point clouds and responding to human instructions both accurately and effectively.

6. Conclusions and Future Directions

In this work, we have taken a step towards enabling large language models (LLMs) to comprehend 3D object point clouds. We addressed this challenge by developing PointLLM which leverages a point cloud encoder with a powerful large language model, allowing for an effective fusion of point cloud information and natural language understanding. To facilitate training, we utilized GPT-4 to generate instruction-following data, resulting in over 660K brief-description instructions and 70K complex-instruction data for object point clouds. We thoroughly evaluated our model



What can we know from this?

This is a 3D model of a sleek and stylish black racing car. The car sports a dark black body, complimented by black tinted windows and matching black tires. The design is optimized for high-speed performance, with features like a low and wide body to improve aerodynamics. The car likely has various functionalities geared towards professional racing, such as a powerful engine, detailed instrumentation, and high-performance brakes.

How many seats are there in this car model?

This car model has two seats.

How many doors does this car have?

This car model has two doors.

Tell me about this.

The 3D model represents a detailed spacecraft, which possesses a sleek aerodynamic shape for efficient space travel. The spacecraft is primarily colored white, providing a clean and professional look. Accentuating its design are red details, likely placed on key components such as the engine area, wing tips, or as lines running along the body. The spacecraft might be used for exploration, research, or transportation in outer space, based on its appearance and common uses of such vehicles.

What physical principles are applied in the ascent of a spacecraft?

The ascent of a spacecraft is achieved through the application of aerodynamics and the propulsion provided by the engines.

What is this?

The 3D object model represents a pair of stylish, yellow sneakers with white laces. These shoes feature a distinct white logo displayed prominently above the back of the heel. The sneakers overall design suggests a modern, trendy style, making them suitable for casual wear or sport activities. Their bright yellow color combined with the clean white detailing gives them a fresh, appealing look.

List some brands that can make similar items.

Nike, Adidas, and Converse are some of the brands that could make a similar item.

Which brand is the best?

The best brand is subjective and depends on personal preferences and the quality of the product.

What is this used for?

It is used for architectural visualization, interior design modeling, or virtual reality experiences.

What else can we know about this model?

The model is structured, with the stairs standing out as a distinct element, and the background being a brown, possibly wooden, floor.

Why do we need a staircase?

A staircase allows for vertical movement between floors within a building. In a residential setting, it is a common feature in houses.

Figure 4. **Dialogues between PointLLM-13B and a human user.** This figure showcases the interaction between PointLLM-13B and a human, emphasizing the model's ability to understand point clouds' shapes, appearances, functionalities, etc. The model's resilience to occlusion enables it to perceive detailed features such as the interior structure of a car or a logo on a shoe, which may be challenging for image inputs. Additionally, PointLLM-13B demonstrates an ability to respond to human instructions with common sense, avoiding biases.

through distinct benchmarks for generative 3D object classification and captioning with various evaluation methods, providing both quantitative and qualitative insights into its abilities. Our model and accompanying resources are open-sourced, inviting the broader community to further explore and enhance this new frontier of multimodal AI.

Future directions. Currently, our model operates within the domain of understanding point clouds and generating text outputs. The next step involves expanding these capabilities to generate 3D point clouds as outputs, allowing for natural language-guided 3D object creation and interactive editing. This transformation can unlock applications in human-computer collaborative 3D generation, streamlining the process of 3D creation and reducing the dependency on specialized tools and expertise. Such advancements may open up possibilities for more accessible 3D design across various applications and will bring 3D perception and creation closer to a wider range of users and industries.

A. Data Collection

Instruction lists. The 30 pre-defined instructions used to prompt the model to briefly and elaborately describe the objects are shown in Tab. 6 and Tab. 7 respectively. These prompts are generated with the assistance of GPT-4 and are coupled with captions to form our description-type data.

Data generation with GPT-4. In Tab. 8 we show an example of using GPT-4 for data generation as well as the system prompt of GPT-4. The input is one human-written caption provided by Cap3D [29] and the outputs are one expanded detailed caption, three single-round conversations, and one multi-round conversation. The system prompt is used for all samples, which guides the model to analyze existing captions based on the general knowledge of 3D objects and generate detailed captions, diverse Q&As, and logically connected multi-round conversations.

B. Benchmarks and Evaluation

B.1. GPT Evaluation Prompts

Close-set zero-shot classification. In this task, we use ChatGPT to post-process the model output by selecting the most probable class index from the 40 ModelNet40 [48] categories. The process is detailed in Tab. 9, where {candidiate_lists} refers to the ModelNet40 category list, and {model_output} refers to the model's response. ChatGPT is required to directly output the category index, category name, and a short reason for the choice. If the description doesn't clearly refer to any one of the categories, ChatGPT must make an educated guess based on the information provided. If ChatGPT cannot infer, then "-1" is returned and a random index will be chosen as the model's classification prediction. We do not use a system prompt for ChatGPT but directly input the prompt.

Open-vocabulary classification. In this task, we use GPT-4 as an evaluator to classify whether the model's response and the human caption are referring to the same object type. The process is outlined in Tab. 10, where {ground_truth} and {model_output} refer to the human caption and the model's response. We do not require the model's response to match exactly with the human caption, as long as it conveys the same object type. We also directly input the prompt for GPT-4 instead of using a system prompt.

Object captioning. In this task, we utilize GPT-4 as an evaluator to assess model-generated captions against human-generated captions (ground truth) of 3D models. GPT-4 is tasked with identifying aspects mentioned in the human caption and calculating the percentage of these aspects that are either correctly mentioned or partially matched in the model's caption on a scale of 0 to 100, with each aspect contributing equally to the score. The evaluation process is detailed in Tab. 11, where {ground_truth} refers to the human caption, and {model_output} refers to the model's response.

B.2. Human Verification of GPT Evaluation

To verify the effectiveness of using GPT models for evaluation, the first author manually checks the evaluation results of ChatGPT and GPT-4.

In the close-set classification task on ModelNet40, the author finds the following:

1. ChatGPT consistently outputs in the desired format, selecting the category or "-1" and providing a reason.
2. When the model output clearly refers to or hints at a category with salient information regarding one of the candidate categories, ChatGPT can accurately identify the corresponding category based on the model's output, showing a high degree of consistency with human-selected options. False negatives or false positives are rare in these cases.
3. If the model output is ambiguous, ChatGPT's selection appears random, aligning with our expectations for handling such cases in close-set classification tasks.

For open-vocabulary classification and object captioning tasks on Objaverse, the author finds that ChatGPT underperforms in identifying the same object concept, acting as a strict judge, and producing more false negatives in classification. It often considers two words or phrases not to refer to the same object type, even when they do. In contrast, GPT-4 demonstrates accurate recognition. After reviewing 50 samples of classification results, the first author has 100% consistency with GPT-4's evaluations. As a result, we opt to use GPT-4 for the open-vocabulary and object captioning tasks on Objaverse. Examples of GPT evaluation can be found in Tab. 9, Tab. 10, and Tab. 11.

Table 6. **The instruction list for brief descriptions.** An instruction from the list is randomly selected and coupled with a human-written caption from Cap3D [29] to form a brief-description instruction following sample.

- Summarize the 3D point cloud object briefly.
- What kind of object is depicted by this point cloud?
- Provide a short explanation of this 3D structure.
- What does this collection of points represent?
- Offer a succinct summary of this 3D object.
- Can you give a brief overview of this point cloud?
- Characterize the object this point cloud is illustrating.
- Share a brief interpretation of this 3D point cloud.
- Provide an outline of this 3D shape's characteristics.
- What object is this point cloud rendering?
- Deliver a quick description of the object represented here.
- How would you describe the 3D form shown in this point cloud?
- What is the nature of the object this point cloud is representing?
- Present a compact account of this 3D object's key features.
- What can you infer about the object from this point cloud?
- Offer a clear and concise description of this point cloud object.
- How would you summarize this 3D data set?
- Give a brief explanation of the object that this cloud of points forms.
- What kind of structure does this 3D point cloud depict?
- Could you delineate the form indicated by this point cloud?
- Express in brief, what this point cloud is representing.
- Give a quick overview of the object represented by this 3D cloud.
- Convey a summary of the 3D structure represented in this point cloud.
- What kind of object is illustrated by this collection of points?
- Describe the object that this point cloud forms.
- How would you interpret this 3D point cloud?
- Can you briefly outline the shape represented by these points?
- Give a concise interpretation of the 3D data presented here.
- Explain the object this point cloud depicts succinctly.
- Offer a summary of the 3D object illustrated by this cloud.

Table 7. **The instruction list for detailed descriptions.** An instruction from the list is randomly selected and coupled with a GPT-4 generated caption to form a detailed-description instruction following sample.

- Can you tell me more about this?
- What does this represent?
- Can you describe this in more detail?
- I'm interested in this, can you explain?
- What is this object made of?
- Could you provide more info about this?
- What exactly am I looking at here?
- What is this?
- Could you describe the detailed structure of this?
- This looks interesting, can you expand on it?
- Can you explain more about this form?
- What can you tell me about the shape of this object?
- Could you delve deeper into this?
- I want to know more about this, can you help?
- Can you walk me through the details of this object?
- Can you provide a comprehensive account of this object?
- Offer a detailed interpretation of this point cloud.
- Please elucidate on the characteristics of this form.
- Could you provide an in-depth description of this structure?
- What does this cloud represent in its entirety?
- Elaborate on the details of this point cloud, please.
- Kindly furnish me with more information about this object.
- Please expand on the intricate structure of this form.
- Provide a meticulous explanation of what these points represent.
- I request a detailed breakdown of this structure.
- Give a thorough rundown of this point cloud.
- Can you offer a complete analysis of this object?
- I would like a comprehensive explanation of this form.
- Please detail the specific features of this point cloud.
- Could you elaborate extensively on what this represents?

Table 8. **An example of data generation with GPT-4.** The input consists of a human-written caption provided by Cap3D [29], guided by a system prompt to analyze the existing caption based on the general knowledge of 3D objects. The outputs include an expanded detailed caption, three single-round conversations, and one multi-round conversation.

Uid	4887da0aab51406dab3c5cb69ec82404

System Prompt	Analyze the 3D object model from the given caption:
	1. Write a new detailed caption by describing it in 50-100 words, identifying its type, appearance, and any other inferred information such as functionalities, usages, daily-life knowledge, etc. Exclude any uncertain details.
	2. Generate 3 single-round Q&As regarding diverse aspects of the object based on the provided captions and your new captions.
	3. Construct 1 set of 3-round Q&As, ensuring logical relevance between the questions and answers. Questions should be different from those in "2".
	
	Format your response as:
	```
	json
	{
	"caption": "description",
	"single_conversation": [{"Q": "Q", "A": "A"} x 3],
	"multi_conversation": [{"Q1": "Q", "A1": "A", "Q2": "Q", "A2": "A", "Q3": "Q", "A3": "A"} x 1]
	}
	```

Input	A female doll with long back cream hair and with black high boots.

Output	{
	"caption": "The 3D object model represents a detailed doll figure of a female character with long, cream-colored hair cascading down her back. It is dressed in fashionable attire with black high boots being one of its highlighted features. The doll, resembling those used in play or as collectible items, exhibits a high degree of intricate detailing, showcasing its fashionable outfit and engaging features, making it appealing for users, especially ones intrigued by doll collection or imaginative play.",
	"single_conversation": [
	{"Q": "What color is the doll's hair?", "A": "The doll's hair is cream-colored."},
	{ "Q": "What is specific about the doll's footwear?", "A": "The specific footwear feature of the doll is its black high boots."},
	{"Q": "What are possible usages of this doll?", "A": "Possible usages for this doll might include being part of a toy collection, used for imaginative play, or as a detailed model for hobbyists or artists."},
	]
	"multi_conversation": [
	{"Q1": "What makes this doll distinguishable?", "A1": "The doll is distinguishable by its long cream-colored hair cascading down its back and its outfit featuring black high boots.", "Q2": "Could this doll be part of a collection or series?", "A2": "Yes, based on the intricate detailing and appearance, it's possible that the doll could be part of a collection or series.", "Q3": "Who would be the likely audience or users of this doll?", "A3": "The likely users of this doll would be individuals who enjoy doll collecting, children or adults for imaginative play, or even hobb yists and artists who may use it as a model."}
	]
	}

Table 9. **The prompt and examples of ChatGPT in close-set zero-shot classification.** ChatGPT post-processes the model output and selects the most probable class index from the available 40 categories, even if the description is vague or ambiguous. The blue placeholders {candidate_lists} and {model_output} refer to the ModelNet40 category list and the model's response, respectively.

Prompt	Given the following free-form description of a 3D object, please determine the most probable class index from the following 40 available categories, even if the description doesn't clearly refer to any one of them. Make your best-educated guess based on the information provided. If the description already contains a valid index, then the index should be selected. If it contains more than one valid index, then randomly select one index (specify your reason). If there is no valid index and it cannot be inferred from the information, return "-1#NA#Cannot infer". Categories: 0: airplane 1: bathtub {candidate_lists} 38: wardrobe 39: xbox Reply with the format of "index#class#short reason (no more than 10 words)". Examples: Input: This is a 3D object model of a cartoon white truck. Output: 7#car#Closest match to "car"in categories. Input: A green leaf in a flower pot. Output: 26#plant#The primary subject "leaf"directly indicates a plant. Input: It's difficult to determine the exact type of this object due to insufficient details. But it seems to be like a piece of furniture. Output: 33#table#Randomly select one kind of furniture from the list. Input: I cannot determine the specific type of the object without additional information or context. Output: -1#NA#Cannot infer. Now analyze the following: Input: {model_output} Output:
Example 1	Input: This 3D model represents a war plane, typically used for combat and military purposes. It is characterized by a large, dominating hue of military gray. The detailing on the model suggests it's a fighter jet, hinting at its usage in air combats. Its posture seems ready for flight, showcasing a typical attribute of war planes - preparedness for battle. This could be used in game design or other virtual reality applications that require detailed, lifelike renderings of objects. Output: 0#airplane#The description explicitly mentions "war plane"and "fighter jet".
Example 2	Input: The 3D model object is a tall, cylindrical-shaped clay container with a flat, smooth top and a slightly wider base. The cylindrical form and the structure suggest that it's possibly used for storing liquids or semi-liquids due to its wide top and flat base allowing easy pouring and stability. Its material and structure indicate it could be a common household item, like a vase or a jug. Output: 37#vase#The description includes key characteristics of a vase (tall, cylindrical shape, wide top, flat base).
Example 3	Input: This is a 3D model of a sleek, black rectangular unit with its sides elegantly curved. On one of its sides, there is a distinctive circle inscribed, adding a unique character to the model. The object's form and design suggest that it can be a piece of technology, like a gadget or a home appliance, with the circle possibly indicating a button or a sensor. Its usage, however, cannot be ascertained without specific context. Output: 22#monitor#The description suggests a sleek, black rectangular unit which can be a monitor. The inscribed circle may indicate a power button or a sensor on the monitor.

Table 10. **The prompt and examples of GPT-4 in open-vocabulary classification.** GPT-4 needs to analyze two sentences to determine if they refer to the same general object or concept, focusing on the type of object, not attributes such as color, size, or shape. The placeholders {ground_truth} and {model_output} refer to the human caption and the model's response, respectively.

Prompt	Analyze two sentences and determine if they're referring to the same general object or concept, focusing on the type of object, not attributes such as color, size, or shape. Respond with `"T"`if they refer to the same thing and `"F"`if not. Also, provide a brief rationale (no more than 20 words) for your judgment. Example: Input: 1. Spiral staircase that goes from a ground floor. 2. This is a 3D model of wooden stairs in light brown Output: T#Both refer to a staircase. Now, analyze the following: Input: 1. {ground_truth} 2. {model_output} Output:
Example 1	Input: 1. A black and brown colored gun. 2. The 3D object is a representation of a futuristic, high-tech gun crafted from a glossy black material. Distinctive features include its metallic handrail, giving an impression of a robust mechanized design. The gun, possibly used in a sci-fi or futuristic setting, denotes advanced technology and might include functionalities such as voice recognition, aiming systems, or biometric triggers. Output: T#Both refer to a gun.
Example 2	Input: 1. A yellow and white fish with black stripes and fins. 2. This is a 3D model of a vibrant, polka-dotted toy fish that is predominantly orange on the body, shifting to white on the belly. The toy has dark brown spots that enhance its appearance, potentially mimicking the natural patterns found on real-life fish. It's an ideal object for educational purposes, helping to introduce children to marine life, as well as serving as a playful item in a playroom or nursery. Output: T#Both refer to a fish.
Example 3	Input: 1. A white cartoon scorpion with eight legs. 2. This is a 3D object model representing a cartoon version of a rare type of spider. The entire model is rendered in white, which highlights its unique and exaggerated characteristics such as multiple legs and a funnel-like body. Its cartoonish appeal makes it more appealing to a younger audience, and it could possibly be used in animations or educational materials to teach children about spiders in a less intimidating way. Output: F#One is a scorpion and the other is a spider.

Table 11. **The prompt and examples of GPT-4 in object captioning.** GPT-4 evaluates the model's response by identifying aspects mentioned in the human caption and calculating the percentage of aspects that are correctly or partially matched in the model's caption. The placeholders {ground_truth} and {model_output} refer to the human caption and the model's response, respectively.

Prompt	Evaluate a model-generated caption against a human-generated caption (ground truth) for a 3D model. Identify the aspects mentioned in the human caption and calculate the percentage of these aspects correctly mentioned or partially matched in the model caption. Score from 0 to 100, where each aspect contributes equally to the score. Consider similar concepts for a partial score. Provide your score (0-100) and a short justification (less than 15 words) in the format of `"score#reason"` Example: Human: A white brown skeleton Model: This is a 3D model of a small, cartoon-like robot. It has a spherical body and is covered in a layer of white dust. Output: 50#mention white; skeleton and robot have similar appearence. Now score the following: Human: {ground_truth} Model: {model_output} Output:
Example 1	Human: A white presentation of a planted trees plantation region. Model: The model is a representation of a table saw with a simple, four-legged design. Output: 0#No aspects from the human caption are mentioned in the model's.
Example 2	Human: Private jet Model: This is a 3D model of a cartoon-style airplane. Output: 50#mentioned airplane, but missed private jet aspect.
Example 3	Human: A cartoon look like a tree in 3d Model: The 3D model is an animated depiction of a tree with gray bark, characterized by its lifelike details and textured surface that mimic the realistic grain of bark. Output: 100#mentioned tree, 3D and artistic/cartoonish aspect.

References

[1] Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, et al. Flamingo: a visual language model for few-shot learning. In *NeurIPS*, 2022. 2, 3

[2] Anas Awadalla, Irena Gao, Josh Gardner, Jack Hessel, Yusuf Hanafy, Wanrong Zhu, Kalyani Marathe, Yonatan Bitton, Samir Gadre, Shiori Sagawa, et al. Openflamingo: An open-source framework for training large autoregressive vision-language models. *arXiv:2308.01390*, 2023. 3

[3] Satanjeev Banerjee and Alon Lavie. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *ACL Workshop*, 2005. 2, 6

[4] 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. In *NeurIPS*, 2020. 2

[5] Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality, 2023. 6

[6] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. *arXiv:2204.02311*, 2022. 2

[7] Wenliang Dai, Junnan Li, Dongxu Li, Anthony Meng Huat Tiong, Junqi Zhao, Weisheng Wang, Boyang Li, Pascale Fung, and Steven Hoi. Instructblip: Towards general-purpose vision-language models with instruction tuning. *arXiv:2305.06500*, 2023. 2, 3, 5, 7, 8

[8] Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. In *NeurIPS*, 2022. 7

[9] 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 *CVPR*, 2023. 2, 3, 5, 6, 7, 9

[10] Danny Driess, Fei Xia, Mehdi SM Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, et al. Palm-e: An embodied multimodal language model. *arXiv:2303.03378*, 2023. 2, 3

[11] Peng Gao, Jiaming Han, Renrui Zhang, Ziyi Lin, Shijie Geng, Aojun Zhou, Wei Zhang, Pan Lu, Conghui He, Xiangyu Yue, Hongsheng Li, and Yu Qiao. Llama-adapter v2: Parameter-efficient visual instruction model. *arXiv:2304.15010*, 2023. 3

[12] Tianyu Gao, Xingcheng Yao, and Danqi Chen. Simcse: Simple contrastive learning of sentence embeddings. *arXiv:2104.08821*, 2021. 2, 6

[13] Tao Gong, Chengqi Lyu, Shilong Zhang, Yudong Wang, Miao Zheng, Qian Zhao, Kuikun Liu, Wenwei Zhang, Ping Luo, and Kai Chen. Multimodal-gpt: A vision and language model for dialogue with humans. *arXiv:2305.04790*, 2023. 3

[14] Tanmay Gupta and Aniruddha Kembhavi. Visual programming: Compositional visual reasoning without training. In *CVPR*, 2023. 3

[15] Yaru Hao, Haoyu Song, Li Dong, Shaohan Huang, Zewen Chi, Wenhui Wang, Shuming Ma, and Furu Wei. Language models are general-purpose interfaces. *arXiv:2206.06336*, 2022. 2

[16] Yining Hong, Haoyu Zhen, Peihao Chen, Shuhong Zheng, Yilun Du, Zhenfang Chen, and Chuang Gan. 3d-llm: Injecting the 3d world into large language models. *arXiv:2307.12981*, 2023. 3

[17] Rongjie Huang, Mingze Li, Dongchao Yang, Jiatong Shi, Xuankai Chang, Zhenhui Ye, Yuning Wu, Zhiqing Hong, Jiawei Huang, Jinglin Liu, et al. Audiogpt: Understanding and generating speech, music, sound, and talking head. *arXiv:2304.12995*, 2023. 2, 3

[18] Shaohan Huang, Li Dong, Wenhui Wang, Yaru Hao, Saksham Singhal, Shuming Ma, Tengchao Lv, Lei Cui, Owais Khan Mohammed, Qiang Liu, et al. Language is not all you need: Aligning perception with language models. *arXiv:2302.14045*, 2023. 2, 3

[19] Tianyu Huang, Bowen Dong, Yunhan Yang, Xiaoshui Huang, Rynson WH Lau, Wanli Ouyang, and Wangmeng Zuo. Clip2point: Transfer clip to point cloud classification with image-depth pre-training. In *ICCV*, 2023. 3

[20] Gabriel Ilharco, Mitchell Wortsman, Ross Wightman, Cade Gordon, Nicholas Carlini, Rohan Taori, Achal Dave, Vaishaal Shankar, Hongseok Namkoong, John Miller, Hannaneh Hajishirzi, Ali Farhadi, and Ludwig Schmidt. Openclip, 2021. 6

[21] Biao Jiang, Xin Chen, Wen Liu, Jingyi Yu, Gang Yu, and Tao Chen. Motiongpt: Human motion as a foreign language. *arXiv:2306.14795*, 2023. 3

[22] Heewoo Jun and Alex Nichol. Shap-e: Generating conditional 3d implicit functions. *arXiv preprint arXiv:2305.02463*, 2023. 2

[23] Bo Li, Yuanhan Zhang, Liangyu Chen, Jinghao Wang, Fanyi Pu, Jingkang Yang, Chunyuan Li, and Ziwei Liu. Mimic-it: Multi-modal in-context instruction tuning. *arXiv:2306.05425*, 2023. 3

[24] Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. *arXiv:2301.12597*, 2023. 2, 3

[25] Junnan Li, Dongxu Li, Caiming Xiong, and Steven Hoi. Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation. In *ICML*, 2022. 3

[26] Chin-Yew Lin. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, 2004. 2, 6

[27] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. *arXiv:2304.08485*, 2023. 2, 3

[28] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2019. 7

[29] Tiange Luo, Chris Rockwell, Honglak Lee, and Justin Johnson. Scalable 3d captioning with pretrained models. *arXiv:2306.07279*, 2023. 2, 3, 5, 6, 11, 12, 14

[30] Alex Nichol, Heewoo Jun, Prafulla Dhariwal, Pamela Mishkin, and Mark Chen. Point-e: A system for generating 3d point clouds from complex prompts. *arXiv:2212.08751*, 2022. 2

[31] OpenAI. Chatgpt. `https://openai.com/blog/chatgpt`, 2022. 2

[32] OpenAI. Gpt-4 technical report. *arXiv:2303.08774*, 2023. 2, 3

[33] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. In *NeurIPS*, 2022. 2

[34] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *ACL*, 2002. 2, 6

[35] Shishir G Patil, Tianjun Zhang, Xin Wang, and Joseph E Gonzalez. Gorilla: Large language model connected with massive apis. *arXiv preprint arXiv:2305.15334*, 2023. 3

[36] Zhiliang Peng, Wenhui Wang, Li Dong, Yaru Hao, Shaohan Huang, Shuming Ma, and Furu Wei. Kosmos-2: Grounding multimodal large language models to the world. *arXiv:2306.14824*, 2023. 3

[37] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *ICML*, 2021. 3, 6

[38] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. In *JMLR*, 2020. 2

[39] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv:1908.10084*, 2019. 2, 6

[40] Yixuan Su, Tian Lan, Huayang Li, Jialu Xu, Yan Wang, and Deng Cai. Pandagpt: One model to instruction-follow them all. *arXiv:2305.16355*, 2023. 3

[41] Qinghong Sun, Yangguang Li, ZeXiang Liu, Xiaoshui Huang, Fenggang Liu, Xihui Liu, Wanli Ouyang, and Jing Shao. Unig3d: A unified 3d object generation dataset. *arXiv:2306.10730*, 2023. 3

[42] Dídac Surís, Sachit Menon, and Carl Vondrick. Vipergpt: Visual inference via python execution for reasoning. *arXiv:2303.08128*, 2023. 3

[43] InternLM Team. Internlm: A multilingual language model with progressively enhanced capabilities. `https://github.com/InternLM/InternLM`, 2023. 2

[44] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv:2302.13971*, 2023. 2, 6

[45] Wenhai Wang, Zhe Chen, Xiaokang Chen, Jiannan Wu, Xizhou Zhu, Gang Zeng, Ping Luo, Tong Lu, Jie Zhou, Yu Qiao, et al. Visionllm: Large language model is also an open-ended decoder for vision-centric tasks. *arXiv:2305.11175*, 2023. 3

[46] Jason Wei, Maarten Bosma, Vincent Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. Finetuned language models are zero-shot learners. In *ICLR*, 2021. 5

[47] Chenfei Wu, Shengming Yin, Weizhen Qi, Xiaodong Wang, Zecheng Tang, and Nan Duan. Visual chatgpt: Talking, drawing and editing with visual foundation models. *arXiv:2303.04671*, 2023. 3

[48] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *CVPR*, 2015. 2, 5, 7, 9, 11

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

[50] Le Xue, Ning Yu, Shu Zhang, Junnan Li, Roberto Martín-Martín, Jiajun Wu, Caiming Xiong, Ran Xu, Juan Carlos Niebles, and Silvio Savarese. Ulip-2: Towards scalable multimodal pre-training for 3d understanding. *arXiv:2305.08275*, 2023. 3, 6

[51] Shukang Yin, Chaoyou Fu, Sirui Zhao, Ke Li, Xing Sun, Tong Xu, and Enhong Chen. A survey on multimodal large language models. *arXiv:2306.13549*, 2023. 3

[52] Xumin Yu, Lulu Tang, Yongming Rao, Tiejun Huang, Jie Zhou, and Jiwen Lu. Point-bert: Pre-training 3d point cloud transformers with masked point modeling. In *CVPR*, 2022. 6

[53] Renrui Zhang, Ziyu Guo, Wei Zhang, Kunchang Li, Xupeng Miao, Bin Cui, Yu Qiao, Peng Gao, and Hongsheng Li. Pointclip: Point cloud understanding by clip. In *CVPR*, 2022. 2, 3

[54] Renrui Zhang, Jiaming Han, Chris Liu, Peng Gao, Aojun Zhou, Xiangfei Hu, Shilin Yan, Pan Lu, Hongsheng Li, and Yu Qiao. Llama-adapter: Efficient fine-tuning of language models with zero-init attention. *arXiv:2303.16199*, 2023. 2, 3

[55] Shilong Zhang, Peize Sun, Shoufa Chen, Min Xiao, Wenqi Shao, Wenwei Zhang, Kai Chen, and Ping Luo. Gpt4roi: Instruction tuning large language model on region-of-interest. *arXiv:2307.03601*, 2023. 3

[56] Deyao Zhu, Jun Chen, Xiaoqian Shen, Xiang Li, and Mohamed Elhoseiny. Minigpt-4: Enhancing vision-language understanding with advanced large language models. *arXiv:2304.10592*, 2023. 2, 3

[57] Xiangyang Zhu, Renrui Zhang, Bowei He, Ziyu Guo, Ziyao Zeng, Zipeng Qin, Shanghang Zhang, and Peng Gao. Pointclip v2: Prompting clip and gpt for powerful 3d open-world learning. In *ICCV*, 2023. 3