Point2Text: Lightweight 3D Point Cloud Captioning via Large Language Models

Yuntian Liu
Department of Computer Science
University of Toronto
Toronto, Canada
yyuntian.liu@mail.utoronto.ca

Oleksii Nakhod
Department of Computer Science
University of Toronto
Toronto, Canada
oleksii@cs.toronto.edu

***Abstract*—Automatic 3D captioning enables searchable 3D repositories, zero-shot cognition, and multimodal AR/VR interfaces. Traditional approaches often rely on projecting 3D meshes into 2D images to utilize powerful vision-language models, losing geometric fidelity. Conversely, recent 3D-native Multimodal Large Language Models (MLLMs) encode rich 3D information but often suffer from excessive parameter counts and complex multi-stage training procedures. We propose Point2Text, a lightweight, 3D-native approach inspired by the ClipCap and PointLLM paradigms. Our architecture bridges a pre-trained Point-BERT encoder with a GPT-2 large language model via a compact Transformer-based mapping network. This design utilizes significantly fewer parameters than state-of-the-art baselines while maintaining semantic coherence. Validated on the Objaverse dataset (660K samples), our preliminary results demonstrate that this lightweight formulation offers a competitive and efficient alternative for 3D dense captioning. Our code is hosted at https://github.com/NilClass246/Point2Txt**

***Index Terms*—Point Cloud, 3D Captioning, Vision-Language Models, MLLM**

Fig. 1. Descriptive captions produced by our proposed Point2Text model by leveraging a pretrained point cloud encoder and LLM. The corresponding colored point clouds are shown here. Additional examples are available in the VI

I. Introduction

With the rapid proliferation of large-scale 3D repositories such as ShapeNet [1], Objaverse [2], and ModelNet [3], there is an increasing demand for scalable, content-aware 3D understanding systems. Specifically, the task of automatic 3D captioning—generating natural language descriptions directly from 3D data—is a critical step toward enabling searchable 3D asset libraries, enhancing zero-shot recognition, and creating conversational agents for Augmented and Virtual Reality (AR/VR).

The core challenge in this domain lies in bridging the semantic gap between geometric structure (point clouds) and linguistic semantics. While Large Language Models (LLMs) have revolutionized Natural Language Processing (NLP), integrating them with 3D percepts remains non-trivial. Existing solutions generally fall into two categories.

The first category employs *2D projection*. By rendering 3D models into multi-view images, these methods leverage mature 2D Vision-Language Models (such as CLIP or BLIP). These include PointCLIP [4] and Diff-3DCap [5]. However, 2D projections often introduce depth ambiguity, occlusion issues, and viewpoint dependency, failing to capture the holistic geometric reality of the object.

The second category is *3D-native MLLMs*. Models such as PointLLM [6] and 3D-LLM [7] extract features directly from point clouds. These models often work to align the 3D representations from some point cloud encoders with a modern LLM. While powerful, these models often rely on massive LLM backbones (e.g., LLaMA-7B/13B), requiring substantial computational resources for training and inference. Furthermore, they often necessitate complex two-stage training pipelines involving latent space alignment followed by instruction tuning.

To address these limitations, we propose **Point2Text**, a lightweight 3D-native framework for 3D captioning. Inspired by the efficiency of ClipCap [8], our model employs a prefix-tuning strategy. We utilize a pre-trained, frozen Point-BERT encoder to extract geometric features, which are then projected into the embedding space of a GPT-2 model via a learnable mapping network. This architecture allows us to keep the heavy LLM backbone frozen (or fine-tuned efficiently) while training only a lightweight mapping network. This has not been done by any prior work.

Our contributions are as follows:

- We present a streamlined 3D-native captioning architecture that eliminates the need for 2D rendering or multi-stage training pipelines.
- We perform a comprehensive ablation study on the mapping network (MLP vs. Transformer) and encoder hyperparameters.
- We demonstrate that a lightweight backbone (GPT-2) can achieve competitive semantic understanding on the

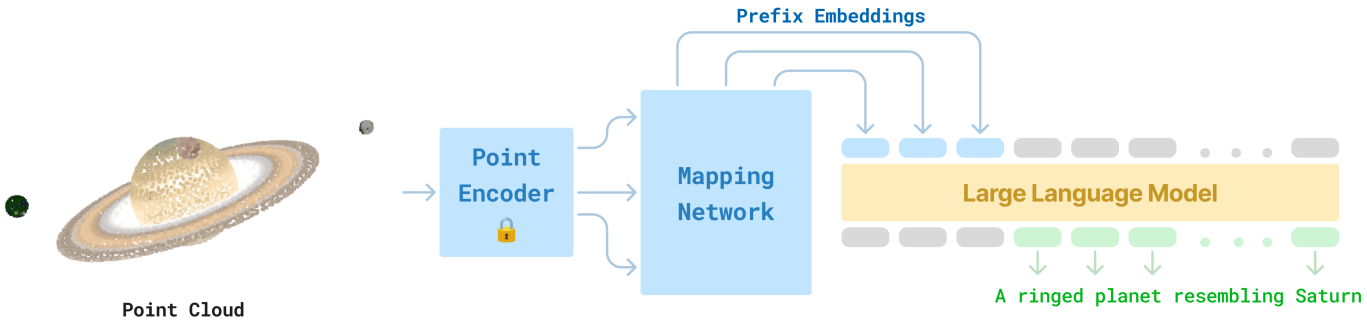


Fig. 2. Overview of the Point2Text architecture. A point cloud ($N = 8192$) is processed by frozen Point-BERT. The resulting features are mapped via a Transformer Mapper into k prefix tokens, which prompt the LLM to generate the caption.

Objaverse dataset compared to heavier baselines.

II. Related Work

A. 3D-Language Learning

Early works in 3D-language learning focused on discriminative tasks such as classification or retrieval. Models like PointCLIP [4] and ULIP [9] align point cloud features with image/text embeddings from CLIP. Aside from the models inspired by CLIP, there are also models focusing on extracting point cloud features solely from the geometric data. These include PointNet++ [10] and Point-BERT [11]. While effective for zero-shot classification, these models are not inherently generative.

Recently, generative 3D MLLMs have emerged. PointLLM [6] integrates a Point-BERT encoder with the LLaMA family of models, achieving state-of-the-art results on object captioning. However, PointLLM requires significant compute to align the 3D encoder with the 7B+ parameter LLM. Our work aims to democratize this capability by demonstrating the efficacy of smaller, more accessible LLMs (GPT-2) in a similar architectural setup.

Another 3D MLLM architecture similar to PointLLM is MiniGPT3D [12], which facilitates the alignment of the point cloud encoder by incorporating 2D priors. This model achieves state-of-the-art results in object captioning with fewer parameters and less training time. However, it has a four-stage training procedure that is complicated to reproduce. Our model aims to provide a one-stage, end-to-end training procedure that makes it easier for other researchers to replicate our work or to deploy it to production.

B. Prefix Tuning for Captioning

Our approach draws significant inspiration from ClipCap [8], which demonstrated that image captioning could be solved by training a lightweight mapping network to translate CLIP embeddings into a sequence of ”prefix” embeddings for a frozen GPT-2. This method treats the visual data as a continuous soft prompt. We extend this paradigm to the 3D domain, replacing the 2D CLIP encoder with a 3D Point-BERT encoder, thereby validating the effectiveness of prefix tuning for geometric data.

The approach proposed by ClipCap is widely adopted in modern image-captioning architectures. The Vision-Language-Vision Auto-Encoder [13] is a notable example: it employs an MLP mapping layer aligned with pretrained diffusion models and derives captions using the resulting aligned image encoder. The success of this model strengthens our confidence in the mapping-network paradigm and motivates the development of this project.

III. Methodology

Our model, Point2Text, consists of three primary components: a 3D Point Cloud Encoder, a Mapping Network, and a Generative LLM. The main objective of our training is to align the point representation from the point cloud encoder with the input embedding of the LLM. The overall architecture is illustrated in Fig. 2.

A. Preprocessing

We represent the input 3D object as a colored point cloud $P \in \mathbb{R}^{N\times 6}$, where $N = 8192$ is the number of points, and 6 represents the (x, y, z) coordinates and (r, g, b) color values. The coordinates are normalized to the unit sphere, and the color values are within the range of $[0, 1]$.

B. 3D Point Cloud Encoder

We utilize pretrained **Point-BERT** [11] as our backbone. We chose this model because of its success in PointLLM. Point-BERT is a Transformer-based encoder pre-trained via Masked Point Modeling. It processes the point cloud into groups using Farthest Point Sampling (FPS) and k-Nearest Neighbors (kNN), generating a sequence of point tokens. The encoder outputs a feature representation $F \in \mathbb{R}^{M\times D_{enc}}$, where $D_{enc} = 384$ is the feature dimension. We extract the global feature representation either via the classification token (CLS) or max-pooling over the sequence, depending on the configuration. The expected final output is a feature representation tensor of the shape $\mathbb{R}^{1\times D_{enc}}$.

Crucially, this part of our model is frozen during training.

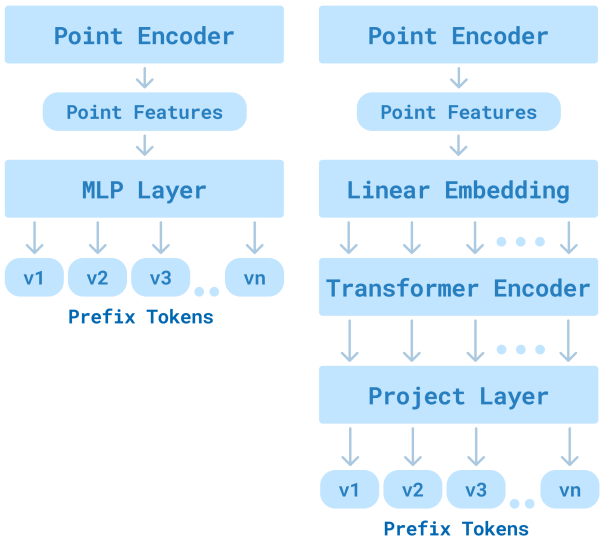


Fig. 3. Our MLP mapper (left) and transformer mapper (right). A point feature vector is fed into the mapper which outputs prefix tokens for the LLM.

C. Mapping Network

To bridge the modality gap between the 3D encoder and the LLM, we employ a Mapping Network that translates the 3D features F into the embedding space of the LLM. We investigate two architectures for this mapper:

1. MLP Mapper: A simple multi-layer perceptron that projects the flattened 3D features directly to the shape $(K \times D_{llm})$, where K is the prefix length and D_{llm} is the LLM embedding size.

2. Transformer Mapper: A more expressive approach where the 3D features serve as input tokens to a Transformer Encoder. This allows the model to learn complex relationships between different local geometric regions before projecting them to the language space. The output is a sequence of K continuous embeddings $P_{prefix} = \{v_1, \ldots, v_K\}$, acting as a "soft prompt" for the LLM.

D. Large Language Model (LLM)

We primarily employ **GPT-2** (124M parameters) as our decoder. We chose this model because of its success in ClipCap and its relatively lightweight design. The model receives the concatenated sequence of the prefix embeddings and the text token embeddings:

$$[v_1, \ldots, v_K, w_1, \ldots, w_L]$$

where w_i are the embeddings of the caption tokens. The training objective is standard Causal Language Modeling (CLM):

$$\mathcal{L} = -\sum_{i=1}^{L} \log P(w_i | v_{1:K}, w_{<i})$$

Crucially, we compute the loss only on the caption tokens w_i, masking the gradients for the prefix tokens. The computed loss is backpropagated through the entire pipeline to train the mapping network in an end-to-end manner. Given the configuration, the LLM may also be fine-tuned, but it is expected to be frozen by default.

For inference, we only fill the prefix embeddings and leave the text token embeddings blank.

IV. Experiments and Results

A. Experimental Setup

Datasets: We utilize the **Objaverse** dataset [2], specifically the subset processed by Cap3D [14], containing 660K 3D objects paired with descriptive captions. Each of these 3D objects contains 8192 normalized, colored points, and all other metadata is disregarded. The point clouds are collected over a wide range of sources, including GitHub, Thingiverse, Sketchfab, Polycam, and Smithsonian 3D Digitization. With this dataset, we ensure that the training and evaluation of our model covers as many topics as possible.

For the purpose of evaluation, we extracted 200 point clouds from the Objaverse dataset as the evaluation set. All evaluation results listed below are obtained over our evaluation set.

Implementation: We use a prefix length of $K = 10$. The model is trained using the AdamW optimizer with a learning rate of $2e{-}5$ and a cosine annealing schedule. The coordinates of the input point clouds are normalized to the unit sphere. Training is performed on NVIDIA RTX 4090 GPUs with a batch size of 128 and epochs of 10.

B. Metrics

We evaluate the quality of generated captions using standard NLP metrics: BLEU-1, ROUGE-L, and METEOR. Additionally, to capture semantic alignment beyond n-gram overlap, we employ **S-BERT** (Sentence-BERT) and **SimCSE** cosine similarity between the generated and ground-truth captions.

C. Ablation Studies

We conducted extensive ablations to determine the optimal hyperparameters and architectures for the 3D-to-Text mapping.

1) Encoder and Sequence Length: We first analyzed the impact of the Point-BERT configuration, specifically the maximum sequence length (ML) processed by the encoder. We hypothesized that a longer sequence ($ML = 64$) would capture finer geometric details. However, as shown in Table I, increasing the sequence length from 32 to 64 yields negligible improvements in n-gram metrics and actually slightly degrades semantic similarity scores (S-BERT/SimCSE). This suggests that the semantic content required for captioning Objaverse objects is sufficiently captured by a more compressed representation ($ML = 32$). We posit that increasing the token count introduces high-frequency geometric noise that complicates the mapping process without adding descriptive value, making the compact $ML = 32$ configuration the more efficient choice.

2) Mapping Network Architecture: We compared the efficacy of a simple MLP versus a Transformer-based mapper. Surprisingly, the MLP mapper consistently outperforms the Transformer variant. As shown in Table II, while the

TABLE I
ABLATION ON POINT-BERT MAX LENGTH

Config	S-BERT	SimCSE	B-1	R-L	MET
$ML = 32$	**0.5658**	**0.7729**	**0.0973**	0.1106	0.2557
$ML = 64$	0.5641	0.7712	0.0971	**0.1117**	**0.2567**

Transformer theoretically offers better context mixing via self-attention, we observe that the direct projection of the MLP is easier to optimize given the frozen encoder. It is worth noting that our MLP implementation actually possesses a higher parameter count than the Transformer mapper due to the large projection layers required to map 384-dimensional point features to the concatenated prefix shape ($K \times 768$). However, to adhere to our "lightweight" design philosophy and minimize inference latency, avoiding the $O(N^2)$ attention overhead of the Transformer, we selected the 2-layer MLP for our final configuration, as it provides the best balance of performance and inference speed.

TABLE II
COMPARISON OF MAPPING ARCHITECTURES

Mapper Type	S-BERT	SimCSE	B-1	R-L	MET
MLP-2L	0.5658	0.7729	0.0973	0.1106	0.2557
MLP-3L	**0.5678**	**0.7739**	**0.0998**	0.1147	**0.2648**
Transformer-2L-8H	0.5183	0.7520	0.0983	**0.1161**	0.2535
Transformer-4L-8H	0.5515	0.7657	0.0974	0.1091	0.2542

3) Prefix Length: We further explored the impact of the prefix length (K) when using the MLP mapper. The prefix length determines the number of virtual tokens injected into the LLM's context window. As shown in Table III, we observe a trade-off. While increasing to $K = 20$ improves strict n-gram overlap (BLEU/ROUGE), it results in a degradation of semantic similarity (S-BERT/SimCSE) compared to $K = 10$. This indicates that longer prefixes may over-condition the LLM on specific geometric tokens at the cost of natural language fluency and semantic coherence. Consequently, we identify $K = 10$ as the optimal middle ground, balancing sufficient geometric conditioning with the generative freedom required for high-quality captions.

TABLE III
ABLATION ON MLP PREFIX LENGTH

Prefix Len. (K)	S-BERT	SimCSE	B-1	R-L	MET
$K = 5$	0.5163	0.7548	0.0947	0.1124	0.2464
$K = 10$	**0.5658**	**0.7729**	0.0973	0.1106	0.2557
$K = 20$	0.5403	0.7570	**0.0991**	**0.1172**	**0.2646**

4) LLM Backbone and Fine-tuning: Finally, we investigated the impact of the LLM choice. We compared a frozen GPT-2 (training only the mapper) against a fully fine-tuned GPT-2, as well as a frozen Qwen3-4B to test scaling laws. As expected, in Table IV, the modern Qwen3-4B (Frozen) outperforms the older GPT-2 (Frozen), illustrating the benefit of a stronger base language model. However, our fine-tuned GPT-2 significantly outperforms both. While fine-tuning Qwen would likely yield superior results, the computational cost would violate our project's core constraint of maintaining a lightweight framework trainable on consumer hardware. The fine-tuned GPT-2 represents the sweet spot between model size (124M) and task-specific adaptation, achieving the best performance while remaining highly efficient.

TABLE IV
IMPACT OF LLM CHOICE AND FREEZING

Model	S-BERT	SimCSE	B-1	R-L	MET
GPT-2 (Frozen)	0.5111	0.7420	0.0907	**0.1110**	0.2446
GPT-2 (Finetune)	**0.5658**	**0.7729**	**0.0973**	0.1106	**0.2557**
Qwen3 (Frozen)	0.5341	0.7452	0.0907	0.1058	0.2523

D. Comparison with State-of-the-Art

As shown in Table V, our lightweight approach (208M parameters: 124M for GPT-2, 62M for MLP, 22M for Point-BERT) achieves competitive performance with the significantly larger PointLLM (7B parameters), particularly in semantic similarity metrics (S-BERT, SimCSE). This validates that for specific dense captioning tasks, a specialized lightweight model can rival much larger generalist models. We note the recent release of PointLLM-v2 [15], which reportedly achieves higher performance on similar benchmarks. However, as the official code and checkpoints for v2 were not publicly available at the time of this work, we were unable to perform a direct comparison. Consequently, we benchmark against the accessible v1 baseline.

TABLE V
COMPARISON WITH STATE-OF-THE-ART (EVALUATED ON CAP3D GROUND TRUTH). "*" INDICATES POINT2TEXT WAS PROMPTED FOR SHORTER CAPTIONS WITH NO MORE THAN 20 TOKENS.

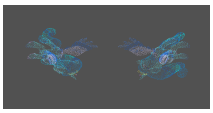
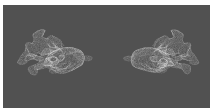
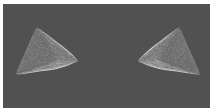
Model	S-BERT	SimCSE	B-1	R-L	MET
PointLLM-7B	0.4919	0.4927	0.0527	0.0859	0.1341
Point2Text-0.2B	0.5658	0.7729	0.0973	0.1106	0.2557
Point2Text-0.2B*	**0.5784**	**0.7958**	**0.3572**	**0.3001**	**0.3141**

E. Discussion on Dataset Limitations

It is worth noting that the performance gap between our varying mapping architectures (MLP vs. Transformer) is relatively narrow. We hypothesize that this saturation is largely attributable to the quality of the ground truth captions. The Cap3D dataset, while large, relies on captions generated by 2D vision-language models, which can sometimes be generic or hallucinatory, failing to capture fine-grained 3D details. Consequently, both Point2Text and PointLLM frequently generate descriptions that are arguably more visually accurate than the "ground truth" labels, yet are penalized by standard n-gram metrics. This suggests that future advancements in 3D captioning may require higher-fidelity, human-verified 3D-native datasets to fully distinguish model capabilities.

TABLE VI

QUALITATIVE RESULTS ON OBJAVERSE. WE COMPARE THE GROUND TRUTH CAPTIONS WITH THE PREDICTIONS GENERATED BY OUR POINT2TEXT MODEL, PROMPTED FOR SHORTER CAPTIONS WITH NO MORE THAN 20 TOKENS. THE MODEL SUCCESSFULLY CAPTURES FINE-GRAINED ATTRIBUTES SUCH AS COLOR AND MATERIAL.

3D Object	Ground Truth	Prediction
	A blue-winged pony with a crown.	A blue and white pony with wings. It has a long tail, resembling a Pokémon character or fairy
	A brown, angular container-like object with small mechanical components and two visible wheels underneath.	A vintage computer setup with a monitor, keyboard on the left side of one panel and various control kn
	A solid, irregularly shaped object with a rectangular base and a semi-circular notch on one side.	A solid purple block with a stepped notch on one side and an irregular cutout near the top.
	Matte white figurine resembling a cartoon bear in a seated position with a content facial expression, wearing a beanie hat.	A white teddy bear. It has a rounded body, large ears with small protrusions on
	White triangular object resembling a diamond.	A solid, irregularly shaped triangular prism with a pointed tip and flat base. The object has a

F. Qualitative Results

Table VI demonstrates the captioning capabilities of Point2Text.

The model successfully captures both high-level semantics (e.g., identifying a ”pony” or ”bear”) and low-level attributes (e.g., ”blue”, ”solid”). Compared to the ground truth Cap3D captions, our model tends to be more verbose in details, leading to longer sequences and later generation of the EOS token. Failure cases primarily occur in abstract shapes or highly occluded objects where the point cloud resolution ($N = 8192$) may be insufficient to resolve fine details.

V. CONCLUSION

In this work, we presented Point2Text, a lightweight framework for 3D point cloud captioning. By adapting the prefix-tuning paradigm to 3D data, we demonstrated that expensive end-to-end training of massive MLLMs is not strictly necessary for generating coherent 3D descriptions. Our results on the Objaverse dataset show that an MLP-based mapper combined with a standard GPT-2 backbone achieves a strong balance between computational efficiency and caption quality. Future work will focus on integrating more advanced 3D backbones and exploring retrieval-augmented generation.

CONTRIBUTIONS

Yuntian Liu: Conceptualized the Point2Text framework and led the core architectural implementation, including the Point-BERT encoder integration and Adapter modules. Developed the primary end-to-end training pipeline and optimized the large-scale dataloading strategies for the Objaverse dataset. Co-authored the Abstract, Introduction, and Methodology sections of the report.

Oleksii Nakhod: Designed and implemented the inference and evaluation suites, integrating traditional and semantic metrics. Developed the visualization tools and engineered the streaming evaluation pipeline for the Objaverse dataset. Conducted the ablation studies regarding mapper architectures and sequence lengths. Co-authored the Experiments, Results, and Related Work sections of the report.

REFERENCES

[1] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, and F. Yu, “Shapenet: An information-rich 3d model repository,” 2015.

[2] M. Deitke, D. Schwenk, J. Salvador, L. Weihs, O. Michel, E. VanderBilt, L. Schmidt, K. Ehsani, A. Kembhavi, and A. Farhadi, “Objaverse: A universe of annotated 3d objects,” 2022.

[3] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao, “3d shapenets: A deep representation for volumetric shapes,” 2015.

[4] R. Zhang, Z. Guo, W. Zhang, K. Li, X. Miao, B. Cui, Y. Qiao, P. Gao, and H. Li, “Pointclip: Point cloud understanding by clip,” 2021.

[5] Z. Shu, J. Wen, S. Li, S. Xin, and L. Liu, "Diff-3dcap: Shape captioning with diffusion models," *IEEE Transactions on Visualization and Computer Graphics*, vol. 31, p. 8129–8142, Oct. 2025.
[6] R. Xu, X. Wang, T. Wang, Y. Chen, J. Pang, and D. Lin, "Pointllm: Empowering large language models to understand point clouds," 2024.
[7] Y. Hong, H. Zhen, P. Chen, S. Zheng, Y. Du, Z. Chen, and C. Gan, "3d-llm: Injecting the 3d world into large language models," 2023.
[8] R. Mokady, A. Hertz, and A. H. Bermano, "Clipcap: Clip prefix for image captioning," 2021.
[9] L. Xue, M. Gao, C. Xing, R. Martín-Martín, J. Wu, C. Xiong, R. Xu, J. C. Niebles, and S. Savarese, "Ulip: Learning a unified representation of language, images, and point clouds for 3d understanding," 2023.
[10] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," 2017.
[11] X. Yu, L. Tang, Y. Rao, T. Huang, J. Zhou, and J. Lu, "Point-bert: Pre-training 3d point cloud transformers with masked point modeling," 2022.
[12] Y. Tang, X. Han, X. Li, Q. Yu, Y. Hao, L. Hu, and M. Chen, "Minigpt-3d: Efficiently aligning 3d point clouds with large language models using 2d priors," 2024.
[13] T. Zhang, Y. Li, Y. cheng Chou, J. Chen, A. Yuille, C. Wei, and J. Xiao, "Vision-language-vision auto-encoder: Scalable knowledge distillation from diffusion models," 2025.
[14] T. Luo, C. Rockwell, H. Lee, and J. Johnson, "Scalable 3d captioning with pretrained models," 2023.
[15] R. Xu, S. Yang, X. Wang, T. Wang, Y. Chen, J. Pang, and D. Lin, "Pointllm-v2: Empowering large language models to better understand point clouds," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–15, 2025.