BlenderLLM: Training Large Language Models for Computer-Aided Design with Self-improvement

Yuhao Du[α], Shunian Chen[α], Wenbo Zan[α], Peizhao Li[α], Mingxuan Wang[α],
Dingjie Song[α], Bo Li[γ], Yan Hu[α], Benyou Wang[α]*
[α]The Chinese University of Hong Kong, Shenzhen
[γ]Information Technology Research Institute, China Tower
[*]**Correspondence:** wangbenyou@cuhk.edu.cn

Abstract

The application of Large Language Models (LLMs) in Computer-Aided Design (CAD) remains an underexplored area, despite their remarkable advancements in other domains. In this paper, we present **BlenderLLM**, a novel framework for training LLMs specifically for CAD tasks leveraging a self-improvement methodology. To support this, we developed a bespoke training dataset, **BlendNet**, and introduced a comprehensive evaluation suite, **CAD-Bench**. Our results reveal that existing models demonstrate significant limitations in generating accurate CAD scripts. However, through minimal instruction-based fine-tuning and iterative self-improvement, BlenderLLM significantly surpasses these models in both functionality and accuracy of CAD script generation. This research establishes a strong foundation for the application of LLMs in CAD while demonstrating the transformative potential of self-improving models in advancing CAD automation. We encourage further exploration and adoption of these methodologies to drive innovation in the field. The dataset, model, benchmark, and source code are publicly available at https://github.com/FreedomIntelligence/BlenderLLM

1 Introduction

CAD is extensively used in industries such as automotive, aerospace, manufacturing, and architecture for 3D design (Heesom and Mahdjoubi, 2004; Pottmann et al., 2005; Susic et al., 2017). Despite its widespread application, the effective use of CAD often demands specialized skills and substantial training, making the design process both labor-intensive and time-consuming. Tasks like parameter adjustments and model validation require considerable human effort, leading to increased project costs and slowing down rapid iteration and innovation (Kreis et al., 2020).

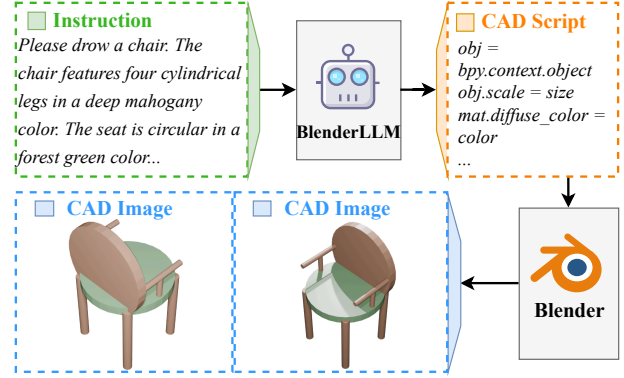Large language models (LLMs) have experienced rapid advancements in recent years, par-

Figure 1: Illustrative Instances

ticularly in architecture and training methodologies. Sophisticated models such as GPT-4 (OpenAI, 2023) have demonstrated human-like performance on a variety of tasks. Their ability to generate coherent and contextually relevant text has made them valuable across numerous applications, including potentially transforming the way CAD tasks are approached.

Problem Definition This paper addresses the challenge of reducing the manual workload associated with CAD design by leveraging the capabilities of LLMs. As illustrated in Figure 1, we utilize LLMs to automate the generation of CAD scripts from natural language inputs. These scripts can be executed in Blender to create precise 3D models. By converting user instructions into executable CAD scripts, our approach streamlines the CAD process, thereby alleviating the manual workload for engineers and designers.

Challenges Although recent work (Kreis et al., 2020; Aarya, 2023; Wu et al., 2023; Zhang et al., 2024) has explored the application of LLM in the CAD field, several significant challenges still hinder its widespread adoption. Firstly, some work is limited by the complexity of input forms, resulting in a high threshold for use. Secondly, there is a notable shortage of high-quality, domain-specific

	BlenderLLM	o1-Preview	GPT-4o	GPT-4-Turbo	Claude-3.5-Sonnet	Gemini-1.5-Pro	BlenderGPT
Burger					✗		
Desk Lamp							
Celtic Knot				✗	✗	✗	✗

Table 1: Examples of the performance of different LLMs. Note: ✗ means that the CAD script generated by the model result in an error during execution, thus no corresponding 3D model is generated.

datasets required to train models capable of capturing the intricate nuances of CAD design. Thirdly, the lack of open-source models limits accessibility, local deployment, and privacy preservation. Finally, the absence of a comprehensive evaluation framework hampers the ability to rigorously assess LLM performance in CAD applications. Addressing these challenges is critical for advancing CAD-oriented LLMs and ensuring robust, secure, and on-premises solutions.

Methodology To address the aforementioned challenges, we present a novel framework consisting of three key components that allow users to generate CAD models with natural language: **Blend-Net**, a high-quality dataset comprising $8k$ samples; **BlenderLLM**, a CAD script generation model; and **CADBench**, a comprehensive benchmarking suite. First, we construct a multi-module data generation pipeline to create BlendNet, whose samples map natural language instructions to bpy scripts. Then, we use BlendNet to fine-tune a model, obtaining the BlenderLLM-base. To further address the issue of data scarcity, we employ a self-improvement approach, utilizing data generated by the model itself to enhance its performance through an iterative process. Furthermore, we introduce a specialized benchmark, CADBench, an evaluation framework employing MLLM-as-judge (Ge et al., 2024) for assessing a model's capacity to generate 3D models from open-ended instructions.

Empirical evaluations demonstrate that Blender-LLM outperforms all baseline models across multiple dimensions on CADBench. Examples are shown in Table 1. Contributions of this paper are summarized as follows:

- We introduce a high-quality dataset, **Blend-Net**, comprising $8k$ diverse CAD samples, along with its data generation pipeline.

- We train a novel bpy script generation model, **BlenderLLM**, which undergoes Supervised Fine-tuning and iterative self-improvement process to achieve state-of-the-art performance.

- We develop a benchmarking framework, **CADBench**, to evaluate the model's ability to generate CAD scripts from user-provided instructions, enabling a systematic assessment of CAD generation capabilities.

2 Related Work

2.1 Computer-Aided Design (CAD)

CAD is a widely used technology in various industries, enabling engineers and designers to create precise digital representations of objects, offering significant advantages in precision, flexibility, and speed. Early efforts leveraged rule-based systems and simple machine learning algorithms to assist in CAD tasks (Chavali et al., 2008). Later, convolutional neural networks were used to convert 2D sketches into 3D models (Li et al., 2020). However, these methods have limitations. Rule-based systems lack flexibility, while machine learning require extensive labeled data and are constrained by their training data's scope (Rapp et al., 2021).

2.2 Large Language Models for CAD

Recent work has begun to explore how LLMs can be adapted for CAD tasks. For instance, CADGPT (Kapsalis, 2024) directly parses natural language inputs into executable commands for CAD software. BlenderGPT (Aarya, 2023) and 3D-PREMISE (Ze-qing et al., 2024) have utilized LLMs like GPT-4 to generate CAD scripts based on natural language prompts. Additionally, CAD-LLM (Wu et al.,

2023) has successfully trained a T5 model for CAD sketch completion. Moreover, CadVLM (Zhang et al., 2024) introduces a multimodal approach that bridges language and vision, enabling the generation of parametric CAD sketches from both textual and visual inputs. Appendix A outlines the key differences between BlenderLLM and existing LLMs designed for CAD-related tasks.

2.3 Blender

Blender is an open-source 3D creation suite widely used in film, game development, and architectural visualization, offering a comprehensive toolset for modeling, animation, and rendering, with flexibility enhanced by its Python API (bpy scripts). Its advantages over other CAD software, including a lower learning curve and broader user base (Hosen and Ahmmed, 2019; Tuori, 2022), make it the ideal platform for CAD tasks. In our work, Blender is used for rendering CAD scripts, acting as an intermediary between the large language model outputs and the visual results.

3 Methodology

3.1 Data Construction

We design and implement a multi-module pipeline for generating high-quality training data for SFT. The pipeline for data construction is illustrated in Figure 2. The multi-module pipeline is composed of three primary components: the **Text Module**, the **Image Module**, and the **Verification Module**. The Text Module generates instructions and their corresponding bpy scripts. The Image Module executes these bpy scripts within Blender to produce images. The Verification Module ensures that the images align with the instructions, thereby validating the data quality.

3.1.1 Text Module

The objective of the text module is to develop diverse instructions and corresponding bpy scripts.

Instruction Generation To encompass a broad range of item types, emulate various communication styles (Sims, 2017), and craft instructions with differing levels of complexity, the diversity of the instructions is categorized along three dimensions:

- **Object Categories:** Objects are classified into 16 categories following the Locarno classification system (Organization, 2013), as detailed in Appendix B.1.1.

- **Instruction Types:** We employ the Myers-Briggs Type Indicator (MBTI) (Myers, 1985) to create eight distinct tones for instructions, as detailed in Appendix B.1.2.

- **Complexity:** To manage the complexity of instructions, we vary their length, classifying them into five categories, as detailed in Appendix B.1.3.

Based on these dimensions, we manually create a set of 135 diverse seed instructions, denoted as $L_{\text{seed}} = \{l_1, l_2, \ldots, l_{135}\}$, where l_i denotes the ith natural language instruction. Next, we employ Self-Instruct data distillation techniques (Wang et al., 2022) to expand these seed instructions into a larger dataset. In each iteration of instruction generation, we randomly sample instances from the L_{seed}. These sampled instances are used to generate new instructions. Through multiple iterations, this process results in a comprehensive dataset of approximately $50k$ instructions, denoted as L_{gen}.

The distribution of both seed instructions L_{seed} and generated instructions L_{gen} by category, type, and length is illustrated in Figure 3. The detailed process is outlined in Appendix B.2.

Script Generation We then utilize GPT-4o[1] to generate pairs $\langle l_j, s_j \rangle$ based on given instructions l_j. For each instruction $l_j \in L_{\text{gen}}$, GPT-4o produces a corresponding script s_j. The generation process ensures that each script is derived from its instruction, as detailed in Appendix B.4.

3.1.2 Image Module

We render the scripts using Blender to generate corresponding images. For each generated 3D object, four images are captured from different angles to better capture the full view of 3D objects, resulting in $\langle l_j, I_j \rangle$ pairs, where $I_j = \{i_{j,1}, i_{j,2}, i_{j,3}, i_{j,4}\}$ is the set of images.

3.1.3 Verification Module

We use GPT-4o as the validator. The model is required to determine whether the images match the instruction based on the given $\langle l_j, I_j \rangle$ pairs, detailed instruction can be found in Appendix B.5.

To verify the reliability of GPT-4o as the validator, we perform manual cross-validation on a portion of the data. We manually validate $10k$ data points, of which 89.7% produce consistent results with the GPT-4o verification, demonstrating
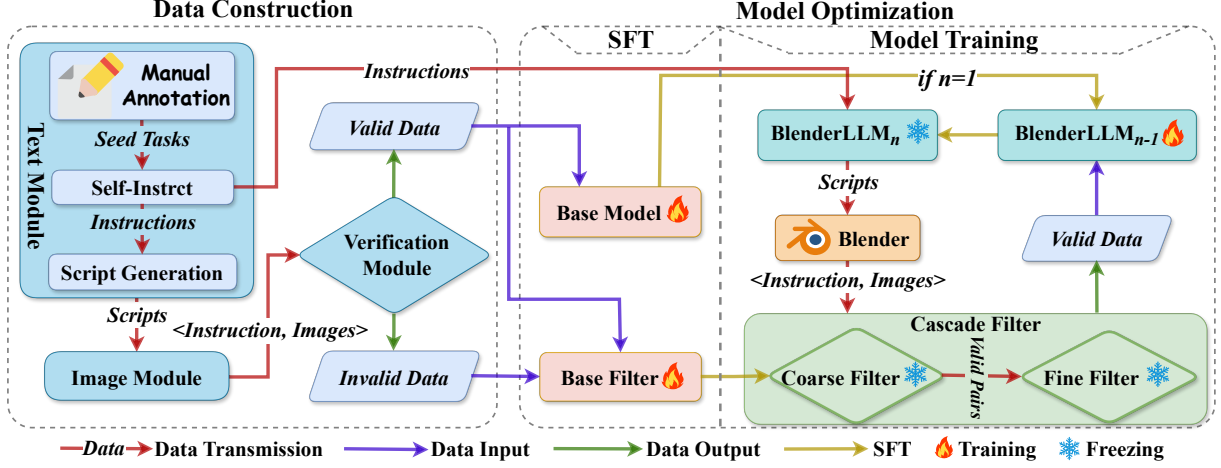
---

[1] Model id gpt-4o-2024-08-06

Figure 2: The Pipeline of the Methodology. In **Step I**, we utilize a multi-module pipeline to construct a high-quality training dataset and fine-tune the Base Model and Base Filter on it, establishing a foundation for the next phase. In **Step II**, the model is fine-tuned by Self-improvement until achieving the optimal model.

the reliability of `GPT-4o` as a validator. Detailed cross-validation result is shown in Appendix B.6.

As a result, we obtain $2k$ accurate $\langle l_j, s_j \rangle$ pairs through manual verification, referred to as `BlendNet-Human`, and $6k$ $\langle l_j, s_j \rangle$ pairs validated solely by GPT-4o, referred to as `BlendNet-GPT`. Combining these two parts, we obtain `BlendNet`.

The diversity of `BlendNet` is illustrated in Figure 3. Additionally, we quantify the complexity of `BlendNet` tasks using three metrics: **Unit Number**, **Parameter Density**, and **Entropy** (Contero et al., 2023). More details about these metrics can be found in Appendix B.7, and sample data is provided in Appendix B.8.

3.2 Model Optimization

The development of BlenderLLM involves a two-phase optimization process: Supervised Fine-tuning (SFT) and Self-improvement.

3.2.1 Step I: Supervised Fine-tuning

We utilize the aforementioned data to fine-tune the `Qwen2.5-Coder-7B-Instruct` model, thereby obtaining the BlenderLLM-base, which serves as the base model for the next step's optimization, denoted as M_0.

3.2.2 Step II: Self-improvement

Due to the limited data, we employed a self-improvement approach, allowing the model to further optimize itself using data it generates. Specifically, we trained a filter with previous data to select high-quality data generated by the model, and then iteratively optimized the model through a cycle of data generation and model training.

Cascade Filter We utilize `BlendNet-Human` and `BlendNet-GPT` as positive examples. $8k$ samples are selected as negative examples from the remaining $\langle l_j, s_j \rangle$ pairs. These data are employed to fine-tune the `Qwen2-VL-7B-Instruct` model, resulting in the Coarse Filter. Combined with `GPT-4o`, which functions as the Fine Filter, they form a Cascade Filter through a cascaded mechanism. Appendix C.2 summarizes the precision of each filter.

Data Generation: In the i-th iteration, we generate training data using the model from the previous iteration M_{i-1}. Specifically, for each instruction l_j, we obtain a script s_j through inference with M_{i-1}. We denote the generated dataset for iteration i as $D_i = \{\langle l_j, s_j \rangle_i\}$. These pairs are rigorously filtered using the Cascade Filter $F(l_j, s_j) \rightarrow \{0, 1\}$ to ensure high-quality data selection, retaining only those pairs for which $F(l_j, s_j) = 1$.

Model Training: The selected high-quality data from the data generation phase is used to fine-tune the model M_{i-1}. This process uses the filtered data to update M_{i-1}, thereby resulting M_i.

The process alternates between data generation and model training, creating an iterative approach to model refinement through Self-improvement, until the loss doesn't drop on the validation set. More details can be found in Appendix C.

4 Benchmarking CAD

In response to the lack of a benchmark for assessing CAD script generation, we develop `CADBench`, a system designed to quantitatively evaluate this capability utilizing the method of MLLM-as-a-judge (Ge et al., 2024). `CADBench` comprises 700 meticulously designed instructions, offering a com-
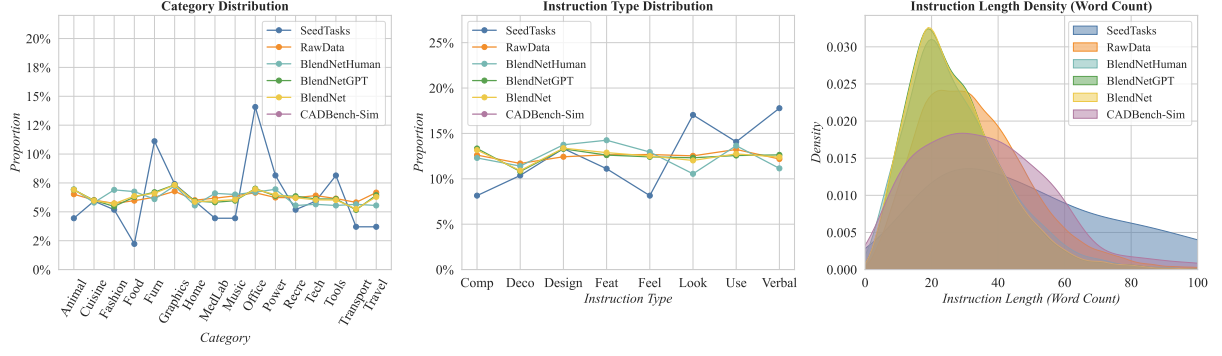
Figure 3: Diversity in Training and Evaluation Datasets. Each dataset is designed to ensure a uniform distribution across *Category* and *Instruction Type*, while maintaining a broad-ranging density in *Instruction Length*.

prehensive dataset for evaluation. Given the open-ended nature of the task, no fixed ground truth is established. Instead, the evaluation process employs a flexible and realistic framework that make the evaluation through predefined criteria.

4.1 Design Principles

`CADBench` is developed by the principles of user-centric, comprehensiveness, granularity and reliability.

User-Centric To simultaneously meet the diversity of test cases and align with practical applications, we constructed `CADBench-Sim` and `CADBench-Wild` through synthesized data and the collection of real data, respectively. `CADBench-Sim` provides controlled synthetic data for baseline testing, covering multiple scenarios, while `CADBench-Wild` offers real-world internet-sourced data to assess the model's practical performance and adaptability.

Comprehensiveness The comprehensive nature of `CADBench` is driven by the necessity to rigorously evaluate 3D generative models across a wide array of object categories, instruction types, and complexities. By systematically covering all categories defined in Appendices B.1, the benchmark provides a robust and inclusive assessment of model performance and generalizability.

Granularity The fine-grained evaluation approach of `CADBench` significantly enhances the benchmark's ability to provide detailed insights into model performance. By incorporating evaluation criteria across three dimensions, as show in Figure 4`CADBench` ensures that models are thoroughly evaluated on diverse aspects, leading to a deeper understanding of their strengths and weaknesses. Detailed explanations and examples of each

Figure 4: Dimensions of Criteria. Numbers represent the average count of criteria in that dimension.

evaluation dimension are available in Appendix D.

Reliability Ensuring the reliability of `CADBench` is paramount, and this is achieved through manual annotation of grading criteria for each sample in `CADBench`. It is also ensured by consistent evaluation and alignment with human preferences. This meticulous approach provides a dependable framework for assessing model performance, fostering trust in the results. For detailed insights into the annotation process, please refer to Appendix G.2.

4.2 CADBench Construction

4.2.1 Part I: CADBench-Sim

`CADBench-Sim` comprises 500 synthetic samples. To ensure the comprehensiveness of `CADBench-Sim`, we employed the **Text Module** from Section 3.1 to generate the instruction data for `CADBench-Sim`. The resulting distribution is shown in Figure 3.

4.2.2 Part II: CADBench-Wild

`CADBench-Wild` incorporates 200 real-world 3D modeling questions, sourced from various CAD-related online forums[2]. These questions represent

---

[2] https://blenderartists.org/c/general-forums/5
https://www.reddit.com/r/blender/

5

complex, real-world scenarios that are substantially more challenging than synthetic tasks, positioning them as out-of-distribution (OOD) data relative to the training data of BlenderLLM. By reflecting actual user requirements, CADBench-Wild offers a critical opportunity to evaluate the generalization capacity of BlenderLLM beyond synthetic environments. The integration of these tasks ensures that CADBench encompasses both synthetic scenarios and real-world applications, providing a comprehensive assessment for the LLMs.

4.3 Criteria

Given the open-ended evaluation characteristics of CAD model assessment, we assist GPT-4o in evaluation by providing customized criteria, instead of ground truth, for each test sample. To achieve a comprehensive and detailed assessment, we designed the criteria from top to bottom into 3 major dimensions and 8 minor dimensions, as shown in the Figure 4. After determining the criteria dimensions, we employ GPT-4o to generate a draft criteria for each sample, and thenmanually verify the criteria following the instruction in Appendix G.2, with criteria examples available in the Appendix D.2. The introduction of criteria not only enhances the comprehensiveness of the evaluation but also improves the consistency between model assessment and human evaluation, as mentioned in the next section.

4.4 Evaluation Protocol

Evaluation Procedure CADBench operates through three distinct stages.

The first stage is script generation. Let e represent the one-shot example used to guide the LLM. The LLM generates a bpy script $s = f(l, e)$ based on these instructions and the context. This ensures improved responses and maintains comparability with BlenderLLM's results.

Second, the generated script s is executed in Blender to produce a set of rendered images $I = \{i_1, i_2, i_3, i_4\}$, where each i_k is a screenshot captured from different angles.

Finally, these images I along with the script are evaluated by GPT-4o using predefined scoring criteria. For each criterion c_i, we define the evaluation function $E(l, I, s, c_i) \rightarrow \{0, 1\}$, where $E(l, I, s, c_i) = 1$ if the criterion is satisfied and 0 otherwise.

Evaluation Methodology To accurately assess the generated CAD outputs from different aspects, we employ GPT-4o for two complementary evaluation approaches:

- **Image-Based Evaluation:** This approach targets the spatial aspects of the CAD scripts which are hard to evaluate without image. Each criterion c_i is assessed for visual fidelity using the evaluation function $E_I(l, I, c_i)$.

- **Script-Based Evaluation:** To accurately assess objective attributes such as size, color, and material, which are challenging to evaluate visually, we evaluate directly using the bpy script s. The evaluation function $E_S(l, s, c_i)$ ensures precise scoring of these attributes.

The detailed evaluation process is provided in Appendix E.

Evaluation Reliability To verify the reliability of the LLM-as-a-Judge framework, two human evaluators independently review a sample of 200 outputs from different models. Appendix G.3 presents the details of the manual annotation for evaluation. And the human evaluation resulted in a *kappa* value of 0.883. The inter-rater reliability between LLM and the human evaluators is calculated using Cohen's kappa coefficient, yielding a *kappa* value of 0.791, which signifies a high level of agreement.

4.5 Evaluation Metrics

For each model, the final score is calculated by averaging the outputs across all criteria:

$$Score = \frac{1}{|C|} \sum_{c_i \in C} E(l, I, s, c_i)$$

Note that for some of the criteria, the image input I is empty, while for others, script input s is empty. See Appendix D.4 for more details.

5 Experiments

5.1 Training Details

We use Qwen2.5-Coder-7B-Instruct as the base model and fine-tune it on BlendNet-Human to obtain the BlenderLLM-base. For subsequent rounds, the input data size is fixed at $2k$ samples to prevent training data saturation and overfitting. During the SFT, full parameter fine-tuning is applied. Each model training session is conducted on four A800

6

Models	CADBench-Sim					CADBench-Wild				
	Attr.↑	Spat.↑	Inst.↑	Avg.↑	E_{syntax}↓	Attr.↑	Spat.↑	Inst.↑	Avg.↑	E_{syntax}↓
Closed-source Models										
o1-Preview	0.729	0.707	0.624	0.687 ± 0.045	15.6%	0.595	0.612	0.542	0.583 ± 0.030	17.5%
GPT-4-Turbo	0.658	0.621	0.488	0.589 ± 0.073	18.2%	0.526	0.541	0.478	0.515 ± 0.027	24.5%
Claude-3.5-Sonnet	0.687	0.608	0.482	0.593 ± 0.084	15.6%	0.529	0.508	0.43	0.489 ± 0.043	26.5%
GPT-4o	0.623	0.593	0.479	0.565 ± 0.062	21.4%	0.460	0.466	0.408	0.444 ± 0.026	28.5%
BlenderGPT	0.574	0.540	0.444	0.519 ± 0.055	25.2%	0.402	0.425	0.368	0.398 ± 0.023	35.0%
Gemini-1.5-Pro	0.535	0.483	0.387	0.468 ± 0.061	30.2%	0.375	0.404	0.361	0.380 ± 0.018	38.0%
Open-source Models										
DeepSeek-V2.5	0.569	0.497	0.372	0.479 ± 0.081	25.2%	0.422	0.394	0.345	0.387 ± 0.032	34.0%
Qwen2.5-Coder-7B-Instruct	0.457	0.352	0.251	0.353 ± 0.084	31.4%	0.354	0.327	0.250	0.310 ± 0.044	37.0%
Qwen2.5	0.367	0.274	0.193	0.278 ± 0.071	44.8%	0.220	0.219	0.170	0.203 ± 0.023	58.5%
LLaMA-3.1-8B-Instruct	0.125	0.087	0.071	0.094 ± 0.023	76.0%	0.130	0.127	0.105	0.120 ± 0.011	65.5%
Mistral-7B-Instruct-V0.3	0.015	0.018	0.015	0.016 ± **0.001**	96.8%	0.023	0.031	0.030	0.028 ± **0.004**	93.0%
CodeLLaMA-7B-Instruct	0.005	0.004	0	0.003 ± 0.002	98.8%	0.009	0.019	0.015	0.014 ± **0.004**	96.5%
BlenderLLMs (Ours)										
Iteration 1	0.784	0.689	0.517	0.663 ± 0.111	5.8%	0.673	0.569	0.444	0.562 ± 0.094	6.0%
Iteration 2	0.822	0.743	0.597	0.721 ± 0.093	5.2%	0.689	0.608	0.473	0.590 ± 0.089	6.0%
Iteration 3	**0.846**	0.760	**0.638**	**0.748** ± 0.085	3.4%	**0.739**	**0.675**	**0.578**	**0.664** ± 0.066	**3.5%**
Iteration 4	**0.846**	**0.767**	0.626	0.747 ± 0.091	**3.2%**	0.717	0.614	0.493	0.608 ± 0.092	5.0%

Table 2: Quantitative Assessment for Instruction-to-Script Generation. This table compares the performance of 12 LLMs and BlenderLLM in assisting CAD script generation on CADBench across three dimensions: *Attr.*, *Spat.*, and *Inst.*. Additionally, *Avg.* and E_{syntax} are provided. A higher score indicates better performance in a given dimension. The results show that BlenderLLM outperforms all other models and effectively handles the task of Instruction-to-CAD script generation.

GPUs with 80GB of memory, with a training time of approximately 21 minutes per SFT round. The batch size, gradient steps, learning rate, epochs, and warmup ratio are set to 1, 2, 1×10^{-5}, 1, and 0.1, respectively. The validation dataset constitutes 10% of the total dataset, with a batch size of 1 and 50 evaluation steps.

5.2 Baselines

To evaluate the performance of BlenderLLM, we compare it against several existing models using a one-shot context approach for all comparisons. The models used for comparison include:**o1-Preview** (Team, 2024), **GPT-4 turbo** (OpenAI, 2023), **Claude3.5-sonnet** (Anthropic, 2024), **GPT-4o** (OpenAI, 2024), **BlenderGPT** (Aarya, 2023), **Gemini-1.5-pro** (Gemini Team, 2024), **DeepSeek-V2.5** (Liu et al., 2024), **Qwen2.5-Coder-7B-Instruct** (Hui et al., 2024), **Qwen-2.5** (Hui et al., 2024), **LLaMA3.1-8B-Instruct** (Touvron et al., 2023), **Mistral-7B-Instruct-V0.3** (Jiang et al., 2023), and **CodeLLaMa-7B-Instruct** (Rozière et al., 2024). Details about these models can be found in Appendix H.

5.3 Main Results

Overall Performance As shown in Table 2, BlenderLLM achieves SOTA performance across all dimensions in both CADBench-Sim and CADBench-Wild, significantly outperforming the second-place model, o1-Preview. A visual comparison of the performance of different models across the dimensions of *attr.*, *spat.*, and *inst.* is provided in Appendix J, where it is evident that BlenderLLM demonstrates substantial improvements in all three dimensions. Furthermore, the comparison shows that BlenderLLM not only adheres more closely to the specified requirements but also offers more reasonable solutions for unmentioned aspects. Its strong performance on CADBench-Wild further highlights BlenderLLM's exceptional generalization capabilities.

Syntax Error Rate As BlenderLLM fine-tuned with high-quality specialized data, its syntax error rate is significantly lower than that of other models. Moreover, the syntax error rate on CADBench-Wild has barely increased, further demonstrating that BlenderLLM has achieved a high level of proficiency in understanding CAD script syntax.

Self-improvement As shown in the examples in Table 3, during the Self-improvement process, BlenderLLM evolves from initially having limited ability to follow instructions, to gradually understanding the instructions and developing spatial reasoning capabilities, ultimately succeeding in modeling the specified object.
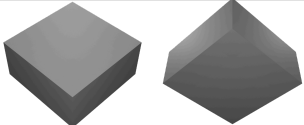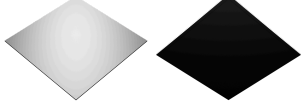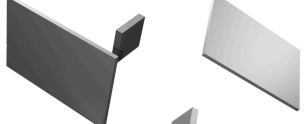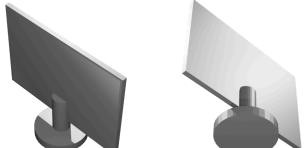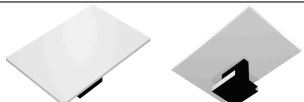
Instruction: *Create a desktop monitor. It should have a 24-inch screen with a thin bezel.*	
Iteration	**Images**
Base Model	
Iteration 1	
Iteration 2	
Iteration 3	
Iteration 4	

Table 3: Visual Process of Self-improvement

5.4 Analysis and Discussion

The experimental results demonstrate that Blender-LLM exhibits significant advantages in *attr.*, *spat.*, *inst.*, and E_{syntax}. Combining the performance of different models on sub-dimensions, as shown in Appendix I, with the comparison of visualization results presented in Appendix J and Table C, these achievements can be attributed to two key factors. First, the BlendNet enables BlenderLLM to learn a variety of instructions. Also, This comprehensive training helped BlenderLLM develop a deeper understanding of the rationality of object attributes, such as the relative size and position of components, as well as the matching of colors and materials. Second, the Self-improvement training strategy allowed BlenderLLM to continuously learn and adapt, progressively enhancing its spatial reasoning capabilities over iteration.

6 Ablation

To demonstrate that Self-improvement Training strategy is more effective than conventional iterative training strategy with similar computational resources, we conducte two comparative experiments:

Methods	CADBench-Sim		CADBench-Wild	
	Avg.	E_{syntax}	Avg.	E_{syntax}
Epoch Accumulation Training				
+ 1 epoch	0.663 ± 0.111	5.8%	0.562 ± 0.094	6.0%
+ 2 epoch	0.685 ± 0.105	5.6%	0.578 ± 0.086	5.0%
+ 3 epoch	0.721 ± 0.099	3.6%	0.568 ± 0.089	6.5%
+ 4 epoch	0.705 ± 0.103	**3.2%**	0.595 ± 0.082	6.0%
Predefined Incremental Training				
+ 1 increment	0.663 ± 0.111	5.8%	0.562 ± 0.094	6.0%
+ 2 increment	0.716 ± 0.098	4.8%	0.559 ± 0.088	5.5%
+ 3 increment	0.722 ± 0.099	3.6%	0.593 ± 0.080	6.5%
+ 4 increment	0.721 ± 0.098	3.8%	0.606 ± 0.087	5.0%
Self-improvement Training				
+ 1 iteration	0.663 ± 0.111	5.8%	0.562 ± 0.094	6.0%
+ 2 iteration	0.721 ± 0.093	5.2%	0.590 ± 0.089	6.0%
+ 3 iteration	$\mathbf{0.748 \pm 0.085}$	3.4%	$\mathbf{0.664 \pm 0.066}$	**3.5%**
+ 4 iteration	0.747 ± 0.091	**3.2%**	0.608 ± 0.092	5.0%

Table 4: Comparison between different SFT strategy.

Epoch Accumulation Training We fine-tune `Qwen2.5-Coder-7B-Instruct`, using the fixed dataset `BlendNet-Human`. The training process begin with one epoch and is incrementally extended by adding an additional epoch in each iteration.

Predefined Incremental Training We fine-tuned the base model, `Qwen2.5-Coder-7B-Instruct`, using a predefined incremental strategy. The process began with the initial dataset, `BlendNet-Human`. In subsequent iterations, $2k$ unused examples from `BlendNet-GPT` were added for further fine-tuning.

Table 4 demonstrates that, after the same number of training iterations, models trained using the Self-improvement Training strategy consistently outperform those trained with the other two approaches on both `CADBench-Sim` and `CADBench-Wild`. Furthermore, Appendix K presents the visualization results of the three different training strategies. It can be observed that, compared to the other two strategies, the Self-improvement Training strategy exhibits superior performance in both instruction-following and spatial reasoning capabilities.

7 Conclusion

In this paper, we propose a comprehensive framework that spans from data construction to self-improvement-based SFT model training and benchmark testing. Through this framework, Blender-LLM, has demonstrated superior performance across various metrics compared to mainstream models. Our results highlight the effectiveness of combining Self-improvement with high-quality dataset, leading to significant advancements in model capabilities.

Limitation

This study has several limitations. First, the data construction and model training primarily focused on basic CAD modeling aspects and did not address more intricate elements, such as material properties, surface treatments, or internal complexity. These factors could influence the model's performance in handling more advanced CAD tasks. Second, our work focused solely on generating CAD scripts from user instructions, without exploring the potential for direct CAD model generation or the integration of multimodal inputs, such as combining user instructions with images. Future research could investigate these avenues to enhance model versatility. Lastly, the model has not been trained for multi-turn dialogues, limiting its ability to engage in more complex, interactive conversations. These limitations highlight key areas for future improvement and expansion of the model's capabilities.

Ethics Statement

This research involves the development and evaluation of a novel dataset and methodology for applying Large Language Models (LLMs) to Computer-Aided Design (CAD). The study does not involve human subjects, nor does it utilize any personally identifiable information. The research adhere to ethical guidelines regarding data privacy and intellectual property. The authors declare no conflicts of interest related to this work. The datasets and models we provide follow the CC-BY 4.0 License.

References

Flip Phillips Aarya. 2023. BlenderGPT. https://github.com/gd3kr/BlenderGPT.git. Accessed: 2024-12-01.

Anthropic. 2024. Claude 3.5 Sonnet. *Anthropic Technical Reports*.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language Models are Few-Shot Learners. *https://arxiv.org/abs/2005.14165*.

Siva R Chavali, Chiradeep Sen, Gregory M Mocko, and Joshua D Summers. 2008. Using rule based design in engineer to order industry: an SME case study. *Computer-Aided Design and Applications*, 5(1-4):178–193.

Manuel Contero, David Pérez-López, Pedro Company, and Jorge D. Camba. 2023. A quantitative analysis of parametric cad model complexity and its relationship to perceived modeling complexity. *Advanced Engineering Informatics*, 56:101970.

Wentao Ge, Shunian Chen, Guiming Hardy Chen, Zhihong Chen, Junying Chen, Shuo Yan, Chenghao Zhu, Ziyue Lin, Wenya Xie, Xinyi Zhang, Yichen Chai, Xiaoyu Liu, Nuo Chen, Dingjie Song, Xidong Wang, Anningzhe Gao, Zhiyi Zhang, Jianquan Li, Xiang Wan, and Benyou Wang. 2024. Mllm-bench: Evaluating multimodal llms with per-sample criteria. *Preprint*, arXiv:2311.13951.

Google Gemini Team. 2024. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *Gemini Technical Reports*.

David Heesom and Lamine Mahdjoubi. 2004. Trends of 4D CAD applications for construction planning. *Construction management and economics*, 22(2):171–182.

Md Hosen and Shahed Ahmmed. 2019. Mastering 3d modeling in blender: From novice to pro. *ABC Research Alert*, 7:169–180.

Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Kai Dang, An Yang, Rui Men, Fei Huang, Xingzhang Ren, Xuancheng Ren, Jingren Zhou, and Junyang Lin. 2024. Qwen2.5-Coder Technical Report. *arXiv preprint arXiv:2409.12186*.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego

de las Casas, Florian Bressand, Guillaume Lample Gianna Lengyel, Lucile Saulnier, Lélio Renard Lavaud, Pierre Stock Marie-Anne Lachaux, Teven Le Scao, Thibaut Lavril, Thomas Wang, and William El Sayed Timothée Lacroix. 2023. Mixtral of Experts. *arXiv preprint arXiv:2401.04088*.

Timo Kapsalis. 2024. Cadgpt: Harnessing natural language processing for 3d modelling to enhance computer-aided design workflows. *Preprint*, arXiv:2401.05476.

Alexander Kreis, Mario Hirz, and Patrick Rossbacher. 2020. CAD-automation in automotive development–potentials, limits and challenges. *Comput.-Aided Des. Appl*, 18:849–863.

Jun Li, Ryan Finnigan, Tom Fuhlbrigge, and Yixin Liu. 2020. Sketch2CAD: Sequential CAD modeling by sketching in context. *Computer-Aided Design*, 125:102831.

Aixin Liu, Bei Feng, and Bin Wang et al. 2024. DeepSeek-V2: A Strong, Economical, and Efficient Mixture-of-Experts Language Model. *arXiv preprint arXiv:2405.04434*.

Isabel Briggs Myers. 1985. *A guide to the development and use of the Myers-Briggs type indicator: Manual*. Consulting Psychologists Press.

OpenAI. 2023. GPT-4 Technical Report. *OpenAI Technical Reports*.

OpenAI. 2024. GPT-4o System Card. *OpenAI Technical Reports*.

World Intellectual Property Organization. 2013. INTERNATIONAL CLASSIFICATION FOR INDUSTRIAL DESIGNS(LOCARNO CLASSIFICATION). *ISBN 978-92-805-2323-2*.

Helmut Pottmann, Stefan Leopoldseder, Michael Hofer, Tibor Steiner, and Wenping Wang. 2005. Industrial geometry: recent advances and applications in CAD. *Computer-Aided Design*, 37(7):751–766.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2019. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *arXiv preprint arXiv:1910.10683*.

Martin Rapp, Hussam Amrouch, Yibo Lin, Bei Yu, David Z Pan, Marilyn Wolf, and Jörg Henkel. 2021. MLCAD: A survey of research in machine learning for CAD keynote paper. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 41(10):3162–3181.

Baptiste Rozière, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Romain Sauvestre, Tal Remez, Jérémy Rapin, Artyom Kozhevnikov, Ivan Evtimov, Joanna Bitton, Manish Bhatt, Cristian Canton Ferrer, Aaron Grattafiori, Wenhan Xiong, Alexandre Défossez,

Jade Copet, Faisal Azhar, Hugo Touvron, Louis Martin, Nicolas Usunier, Thomas Scialom, and Gabriel Synnaeve. 2024. Code llama: Open foundation models for code. *arXiv preprint arXiv:2308.12950*.

Ceri M. Sims. 2017. Do the big-five personality traits predict empathic listening and assertive communication? *International Journal of Listening*, 31(3):163–188.

I Susic, M Travar, and M Susic. 2017. The application of CAD / CAM technology in Dentistry. In *IOP Conference Series: Materials Science and Engineering*, volume 200, page 012020. IOP Publishing.

O1 Team. 2024. Learning to reason with llms.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Roziere, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. LLaMA: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Vili Tuori. 2022. Advantages of Photogrammetry in Creating Photorealistic 3D Assets Using Blender. *Theseus*.

Yizhong Wang et al. 2022. Self-Instruct: Aligning Language Models with Self-Generated Instructions. *arXiv preprint arXiv:2212.10560*.

Sifan Wu, Amir Khasahmadi, Mor Katz, Pradeep Kumar Jayaraman, Yewen Pu, Karl Willis, and Bang Liu. 2023. CAD-LLM: Large Language Model for CAD Generation. *Conference on Neural Information Processing Systems 2023*.

Yuan Zeqing, Lan Haoxuan, Zou Qiang, and Zhao Junbo. 2024. 3D-PREMISE: CAN LARGE LANGUAGE MODELS GENERATE 3D SHAPES WITH SHARP FEATURES AND PARAMETRIC CONTROL? *arXiv:2401.06437*.

Haoran Zhang, Yue Wang, Wei Wang, Hao Dong, Yongjing Liu, and Pan Pan. 2024. CadVLM: Bridging Language and Vision in the Generation of Parametric CAD Sketches. *arXiv preprint arXiv:2409.17457*.

A Comparison of BlenderLLM and Recent works

The comparison of BlenderLLM and recent works is shown in Table 5.

B Data Construction

B.1 Categories, Instruction Types and Instruction Length

B.1.1 Categories

We based on the Locarno Classification System to generate our own classification method and concluded all objects into 16 categories $C = \{Tech, Music, \ldots, Home\}$, with their names listed below:

- *Tech*: Recording, telecommunication, or data processing equipment

- *Music*: Musical instruments

- *Animal*: Articles for the care and handling of animals

- *Furn*: Furnishing

- *Transport*: Means of transport or hoisting

- *Office*: Stationery and office equipment, artists' and teaching materials

- *Food*: Foodstuffs

- *MedLab*: Medical and laboratory equipment

- *Fashion*: Articles of clothing and haberdashery

- *Graphics*: Graphic symbols, logos, surface patterns, ornamentation, arrangement of interiors and exteriors

- *Recre*: Recreational goods (Games, toys, tents, and sports goods)

- *Tools*: Tools and hardware

- *Travel*: Travel goods, cases, parasols, and personal belongings, not elsewhere specified

- *Power*: Electrical systems (Equipment for production, distribution, or transformation of electricity)

- *Cuisine*: Culinary machines (Machines and appliances for preparing food or drink, not elsewhere specified)

- *Home*: Household goods, not elsewhere specified

B.1.2 Instruction Types

We notice the difference between styles of prompting. In order to make input data more diverse, we specified them into 8 types, denoted as $T = \{Verbal, Look, \ldots, Design\}$, with their names listed below:

- *Verbal*: Verbal Question
  Direct and conversational requests for creating dynamic or specific action images, focusing on movement and behavior.

- *Look*: Outlook Question
  Focuses on the physical appearance of objects, emphasizing visual attributes like color and shape.

- *Use*: Specific Usage Question
  Emphasizes the practicality or functionality of objects, highlighting how they can be used or their intended purpose.

- *Deco*: Decoration Question
  Concentrates on the aesthetic or decorative aspects of objects, underlining their decorative value and appearance.

- *Feel*: Feeling Question
  Involves sensory experiences or the tactile quality of objects, aiming to capture the feel or sensory impression they convey.

- *Comp*: Comparing Question
  Entails making distinctions based on comparison, often with a focus on historical or time-specific characteristics to capture a specific style.

- *Feat*: Feature Question
  Centers around exploring and describing specific features of objects, requiring creativity based on given characteristics.

- *Design*: Design Question
  Revolves around creative construction or conceptualization based on specific shapes or ideas, emphasizing innovative design solutions.

B.1.3 Instruction Length

We set the length of the instruction to enhance the variety. We place instruction into 5 classes regarding to their words count, as $L = \{VS, S, \ldots, E\}$.

Models	Open Source	Self-improvement	Methodology	LM Backbone	Size	Task
BlenderGPT (Aarya, 2023)	✗	✗	Prompt Engineering	GPT-4 (OpenAI, 2023)	/	Text-to-Code
CADGPT (Kapsalis, 2024)	✗	✗	Prompt Engineering	GPT-4 (OpenAI, 2023)	/	Text-to-API
CAD-LLM (Wu et al., 2023)	✗	✗	Training	T5 (Raffel et al., 2019)	770M	CAD-to-CAD
CADVLM (Zhang et al., 2024)	✗	✗	Training	/	/	Multimodal-to-CAD
BlenderLLM	✓	✓	Training	Qwen2.5-Coder (Hui et al., 2024)	7B	Text-to-Code

Table 5: Comparison of BlenderLLM and Recent Works

- *VS*: Very Short

- *S*: Short

- *M*: Medium

- *L*: Long

- *E*: Extended

B.2 Instruction Generation Process

the generation process for instructions is shown in Algorithm 1

Algorithm 1 Instruction Generation Process

1: **Input:**
2:   I: Set of instructions
3:   I_{prev}: Set of previous instructions
4:   I_{seed}: Set of seed instructions
5:   C: Set of categories
6:   T: Set of types
7:   L: Set of lengths
8:   D_{old}: Dataset of old instructions
9:   threshold: Threshold for name counts
10:   S: Similarity score function
11: **Output:**
12:   I_{new}: Set of new instructions
13:   C_{new}: Set of new categories
14:   T_{new}: Set of new types
15:   L_{new}: Set of new lengths
16:   $N_{\text{normalized}}$: Normalized names set
17:   N_{throwed}: Filtered names (names to avoid)
18:   $N_{\text{remaining}}$: Remaining names
19: **Iterative Generation:**
20:   $I_{\text{new}} \leftarrow \{i \in I \mid S(i,j) < 0.8, \forall j \in I_{\text{prev}} \cup I_{\text{seed}}\}$
21:   $|I_{\text{new}}| \leftarrow 10$
22: **Constraints:**
23:   $C_{\text{new}} \leftarrow \{C_i \mid C_i \in C, |C_{\text{new}}| = 16\}$
24:   $T_{\text{new}} \leftarrow \{T_i \mid T_i \in T, |T_{\text{new}}| = 8\}$
25:   $L_{\text{new}} \leftarrow \{L_i \mid L_i \in L, |L_{\text{new}}| = 5\}$
26: **Normalization and Filtering:**
27:   **1. Normalized Names Set:**
28:   $N_{\text{normalized}} \leftarrow \{\text{normalize}(d_{\text{name}}) \mid d \in D_{\text{old}}, d_{\text{category}} \in C_{\text{new}}\}$
29:   **2. Name Counts:**
30:   $N_{\text{counts}} \leftarrow \text{Counter}(\text{normalize}(d_{\text{name}}) \mid d \in D_{\text{old}})$
31:   **3. Filtered Names (Names to Avoid):**
32:   $N_{\text{throwed}} \leftarrow \{n \mid n \in N_{\text{normalized}}, N_{\text{counts}}[n] > \text{threshold}\}$
33:   $N_{\text{remaining}} \leftarrow N_{\text{normalized}} - N_{\text{throwed}}$
34: **Output:**
35:   $I_{\text{new}}, C_{\text{new}}, T_{\text{new}}, L_{\text{new}}, N_{\text{normalized}}, N_{\text{throwed}}, N_{\text{remaining}}$

B.3 Validation

B.4 Script Generation

The process for script generation is shown in Figure 5.

GPT \ Human	Pass	Fail
Pass	21.61%	7.20%
Fail	3.13%	68.06%

Table 6: Cross Validation

B.5 Validation Process

The process for validation is shown in Figure 6.

B.6 Cross Validation

Table 6 shows the details about the cross validation result. The proportion of samples where humans and models consistently judge passed is 21.6%, the proportion of samples where humans and models consistently judge not passed is 68.1%, and the proportion of samples where human and model judgments differ is only 10.3%, which demonstrates a high degree of consistency between human and model assessments. The instruction for human validation can be found in Appendix G.1.

B.7 The Complexity of BlendNet

we define three key metrics to quantify the complexity of BlendNet:

- **Unit Number**: This metric represents the number of basic shapes within the 3D Model. It serves as an indicator of geometric complexity, where higher values imply a greater number of components and higher structural complexity.

- **Parameter Density**: This metric calculates the average complexity per shape, defined as:

$$Parameter\ Density = \frac{Parameter\ \#}{Unit\ \#} \quad (1)$$

A higher parameter density indicates that each shape is more parameterized, implying greater irregularity and higher computational complexity. This value reflects how intricately the shapes are defined and how complex the relationships between the parameters are within the 3D model.

- **Entropy**: Entropy measures the spatial diversity of the shapes in the 3D space. It is defined
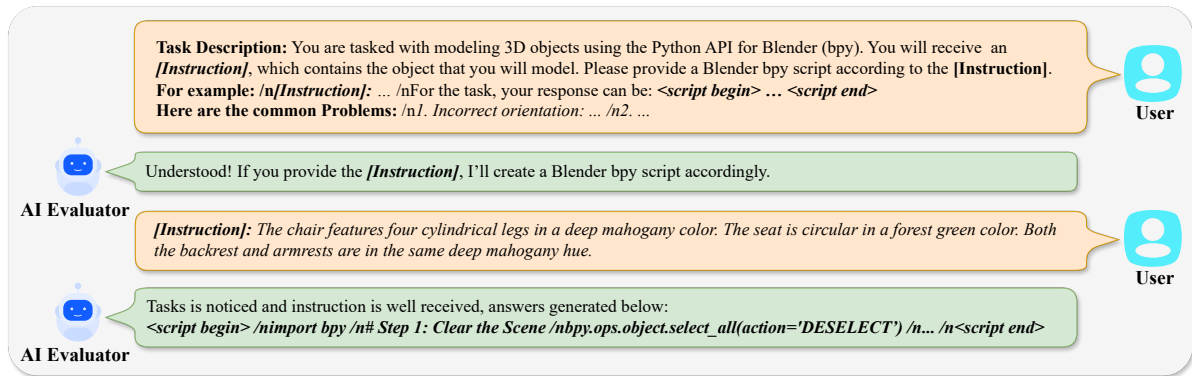
Figure 5: Process for Script Generation. We carefully designed the prompt to maximize the responsiveness and effectiveness of GPT-4o, ensuring that it generates high-quality and contextually accurate CAD scripts.
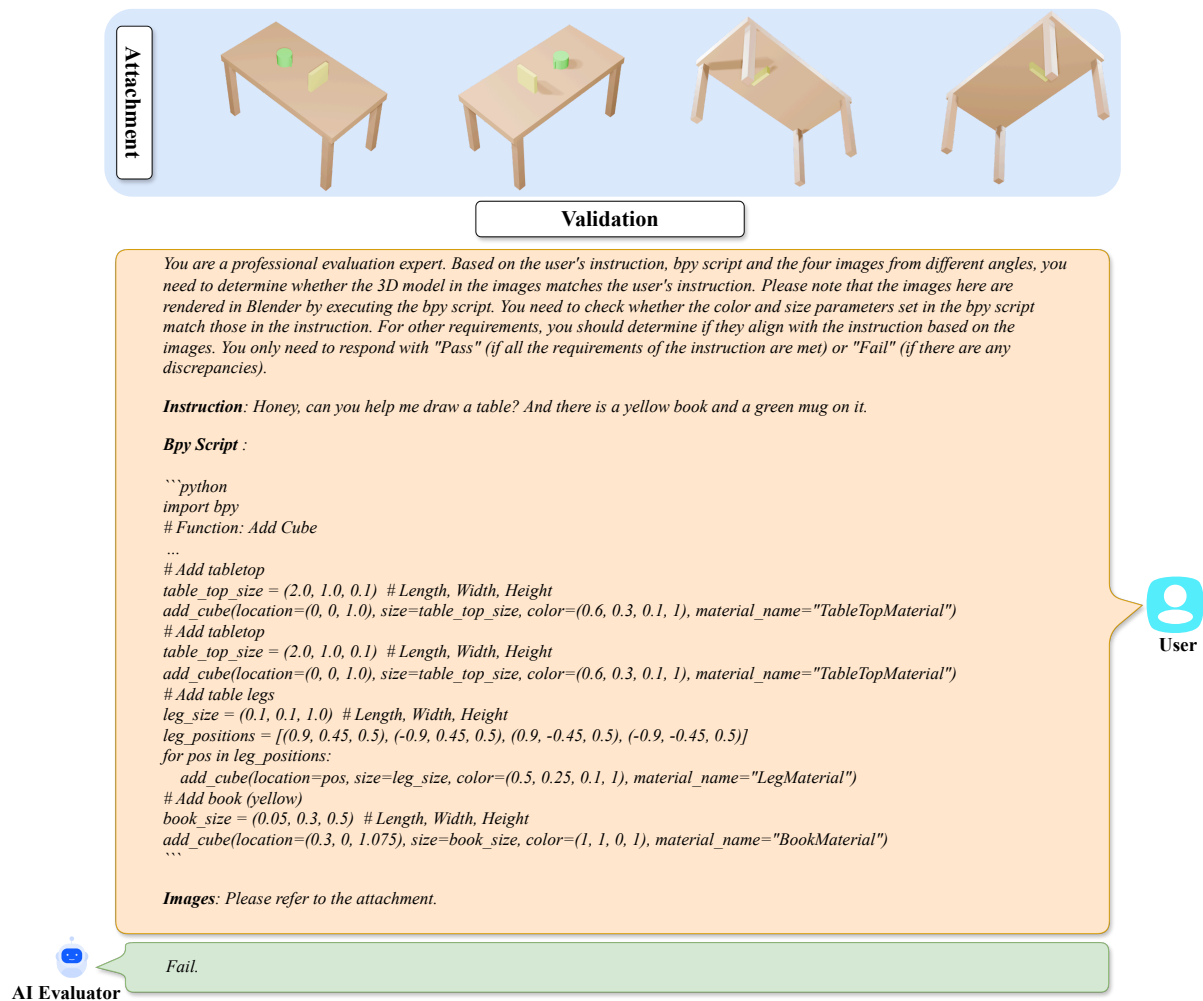

Figure 6: Validation Process

as:
$$H = -\sum p_i \log(p_i) \qquad (2)$$

where p_i is the probability density in 3D voxels. Higher entropy values indicate greater spatial diversity, which implies more irregular and unpredictable configurations. This metric helps capture the distribution and variation of shapes across the 3D space, with larger values corresponding to more complex and diverse spatial arrangements.

The distribution of `BlendNet-Human`, `BlendNet-GPT`, and `BlendNet` across these three metrics is shown in Figure 7.

B.8 Samples of BlendNet

The Samples of `BlendNet` is shown in Table 7.

C Self-improvement Process

C.1 Self-improvement Algorithm

The algorithm for the Self-improvement process is referenced in Algorithm 2.

C.2 Cascade Filter

The classification accuracy of cascade filter is shown in Table 8. Result shows that cascade filter outperforms both single filter.

D Benchmark

D.1 Dimensions for Criteria

D.1.1 Object Attributes (*Attr.*)

Definition: This section focuses on evaluating the visual and physical properties of objects, such as shape, color, size, proportion and material characteristics.

- *Shape*: **Shape Accuracy**
  Ensure that the objects' shapes align with the instructions, including basic geometries like cubes, spheres, and cylinders.

- *Color*: **Color Representation**
  Confirm that the objects' colors precisely match the instructions, including shades, gradients, and lighting effects.

- *Size*: **Size Accuracy**
  Check that objects' absolute sizes, such as height, width, and depth, are consistent with the instructions.

- *Proportion*: **Proportion Accuracy**
  Ensure the size relationships between different parts of the objects are correct relative to each other.

- *Texture*: **Texture and Surface Detail**
  Verify that surface materials like metal, wood, or glass are accurately represented through texture, gloss, or transparency.

D.1.2 Spatial Understanding and Structure (*Spat.*)

Definition: This section evaluates how well the model comprehends and represents the position, relationships, and structure of objects within 3D space.

- *Space*: **Spatial Awareness**
  Assess whether the objects' positions and relative relationships within the 3D coordinate system are accurate and logical.

- *Contact*: **Object Contact and Distance**
  Verify if the relative distances between objects are reasonable, and whether physical interactions like contact, stacking, or collision are handled correctly.

D.1.3 User Instruction Understanding and Execution (*Inst.*)

Definition: This dimension evaluates how accurately the model interprets and executes the user's instructions.

- *Execute*: **Execution Accuracy**
  Ensure that the objects fully conform to user instructions, including shape, color, size, and material, with no deviations.

D.2 Example for Criteria

Instruction: *The chair features four cylindrical legs in a deep mahogany color. The seat is circular in a forest green color. Both the backrest and armrests are in the same deep mahogany hue. The height of the legs is 35cm. The height of the armrests is 10cm.*

For this instruction, the **Evaluation Criteria** is:

- **Object Attributes:**

  - **Shape accuracy:**
    * *The object in the images is a chair.*
    * *The chair has four cylindrical legs.*
    * *The seat is circular.*
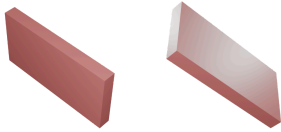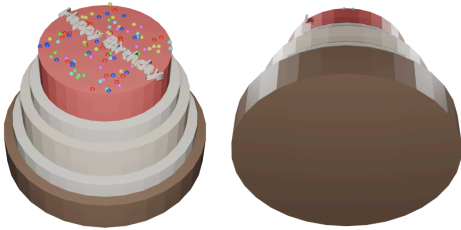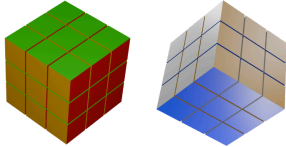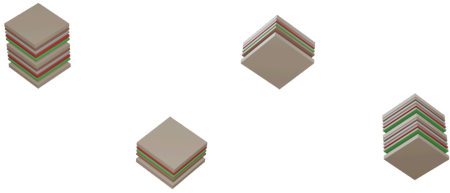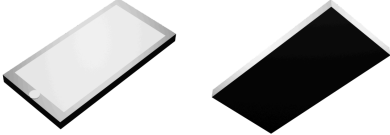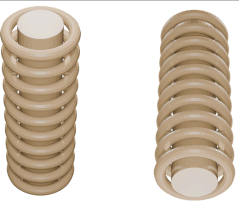
Instruction	Images	Unit Number	Parameter Density	Entropy
Design an eraser.		<u>1</u>	9.00	2.08
Let's create a birthday cake with three layers. The bottom layer should be chocolate, the middle layer vanilla, and the top layer red velvet. Each layer should be separated by a thick layer of buttercream frosting. Add a decorative border of frosting around the top edge, and place colorful sprinkles all over the surface. Finally, add a Happy Birthday message on top.		**107**	0.50	3.66
How does solving a puzzle cube make you feel? Can you create a 3D model of a standard 3x3 puzzle cube?		27	<u>1.41</u>	3.99
Compare the appearance of a club sandwich and a BLT sandwich. Create both sandwiches with the classic ingredients stacked between slices of bread.		2	**13.50**	4.02
Design a 3D model of a smartphone with a screen and a single button on the front.		3	7.67	<u>1.34</u>
Could you design a 3D model of a transformer coil? It should be cylindrical with multiple copper windings.		11	1.37	**6.31**

Table 7: Samples of BlendNet

Filters	Cascade Filter	Coarse Filter	Fine Filter
Precision	**81.8%**	61.9%	73.3%

Table 8: Precision of different Filters. Data deemed acceptable by the Coarse Filter is subsequently processed by the Fine Filter for further verification. This cascaded approach achieves both cost savings and high accuracy.

* The backrest is rectangular.
* The armrests are also cylindrical.

– **Color representation:**

* The color of the legs is deep mahogany.
* The seat color is forest green.
* The backrest color is deep mahogany.
* The color of the armrests is deep mahogany.

– **Size:**

* The height of the legs is 35 cm.
* The height of the armrests is 10 cm.

Algorithm 2 Self-improvement Process

1: **Definitions:**
2:    i: Iteration number   ▷ Counter for optimization iterations, starting from 1
3:    M_i: Model obtained at the i-th iteration   ▷ e.g., M_1 is the first iteration model
4:    M_{final}: Optimal model   ▷ The final model with the best evaluation score
5:    I_j: Instruction for the j-th task   ▷ j-th Task description in natural language
6:    S_j: Script generated for I_j ▷ generated script based on I_j
7:    R_j: Rendered images for S_j ▷ Images by rendering S_j
8:    P_j: Data pair (I_j, R_j) ▷ Combination of instruction and rendered images
9:    CF: Cascade filter for data pair evaluation   ▷ Filters data pairs to ensure quality
10:    T_i: Training dataset at iteration i ▷ Dataset used to train M_i
11:    $Loss_i$: Evaluation score for model M_i on validation Set   ▷ Performance on validation Set
12: **Initialization:**
13: $i \leftarrow 1, M_0 \leftarrow \text{BaseBlenderLLM}, S_0 \leftarrow 0$
14: **while** true **do**   ▷ Main iterative process
15:    $T_i \leftarrow \emptyset$ ▷ Initialize training data for iteration i
16:    **while** true **do**
17:       $S_j \sim M_{i-1}(I_j)$ ▷ Generate script S_j from M_{i-1} using I_j
18:       $R_j = \text{Render}(S_j)$ ▷ Render images R_j using script S_j
19:       $P_j = (I_j, R_j)$
20:       $CF(P_j) = \begin{cases} \text{Match,} & \text{if } P_j \text{ satisfies filter criteria} \\ \text{No Match,} & \text{otherwise} \end{cases}$ ▷ Evaluate the data pair using cascade filter
21:       **if** $CF(P_j) = \text{Match}$ **then**
22:         $T_i \leftarrow T_i \cup \{P_j\}$ ▷ Add valid pair to training dataset
23:       **else**
24:         Discard P_j ▷ Ignore invalid data pairs
25:       **end if**
26:       **if** $|T_i| \geq 2000$ **then**
27:         **Break**   ▷ Stop collecting data if threshold is met
28:       **end if**
29:    **end while**
30:    $M_i = \text{Train}(M_{i-1}, T_i)$ ▷ Train model M_i using M_{i-1} and T_i
31:    $Loss_i = \text{Evaluate}(M_i, \text{Validation Set})$ ▷ Evaluate M_i on Validation Set
32:    **if** $Loss_i > Loss_{i-1}$ **then**
33:       $M_{\text{final}} \leftarrow M_{i-1}$ ▷ Save previous model if score degrades
34:       **Break**
35:    **else**
36:       $M_{i-1} \leftarrow M_i$ ▷ Update base model for next iteration
37:    **end if**
38:    $i \leftarrow i + 1$ ▷ Increment iteration counter
39: **end while**
40: **Output:** M_{final}

- **Proportion:**
  - *The seat is proportionate to the legs.*
  - *The backrest is at a reasonable height relative to the seat.*

- **Texture and surface detail:**
  - *The legs have a smooth wooden texture.*
  - *The seat may have a fabric texture suitable for upholstery.*

- **Spatial Understanding and Structure:**

  - **Three-dimensional spatial awareness:**
    - *The legs are positioned correctly for stability.*
    - *The seat is properly supported by the legs.*

    - *The backrest is properly supported by the seat.*
    - *The two armrests are symmetrical.*

  - **Object distance and contact:**
    - *The legs do not overlap with the seat.*
    - *There is no gap between the seat and the legs.*
    - *The backrest connects with the seat at the edge.*
    - *The armrests are fixed to the backrest and seat.*

- **User Instruction Understanding and Execution:**

  - **Instruction execution accuracy:**
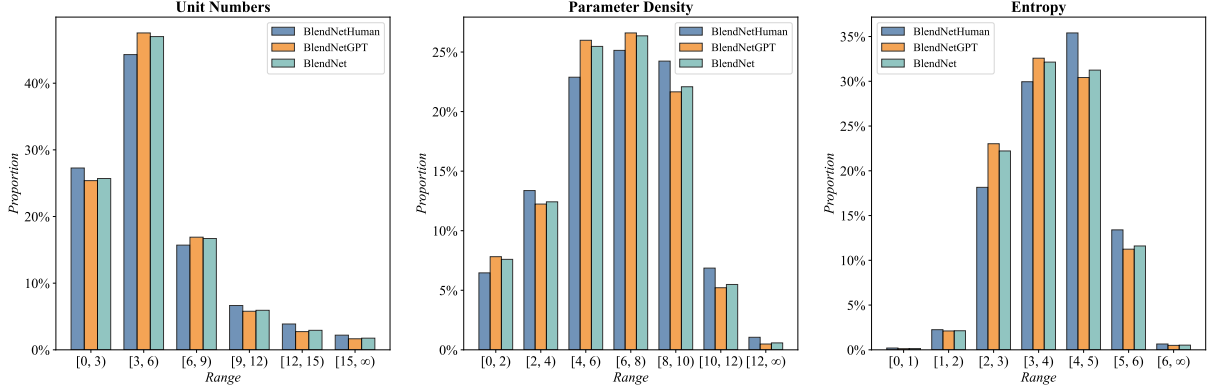   - *All specified attributes are accurately represented.*

Figure 7: The complexity distribution of BlendNet

∗ *There are no deviations from the instructions.*

D.3 Average Number of Criteria across Dimensions

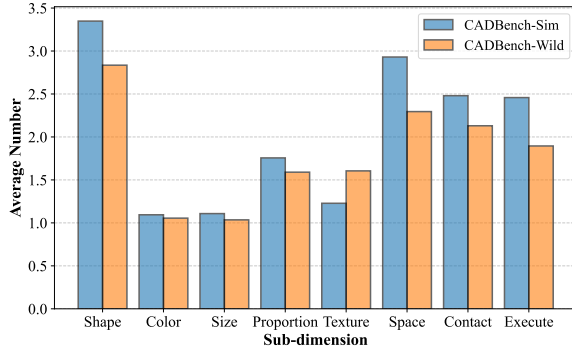The average number of criteria of each sample across dimensions is shown in Figure 8.

Figure 8: Average number of criteria for each sub-dimension.

D.4 Evaluation Metrics

Sub-dimension Scores The average score for sub-dimension j within dimension k, denoted as $SubDimScore_{k,j}$, is calculated as follows. Here, N_{kj} represents the total number of criteria in sub-dimension j, and S_{kji} is the score for the i-th criterion:

$$SubDimScore_{k,j} = \frac{1}{N_{kj}} \sum_{i=1}^{N_{kj}} S_{kji} \qquad (3)$$

Dimension Scores The average score for a specific dimension k, denoted as $DimScore_k$, is calculated using Equation 4. In this equation, N_k represents the number of sub-dimensions within dimension k:

$$DimScore_k = \frac{1}{N_k} \sum_{j=1}^{N_k} SubDimScore_{k,j} \qquad (4)$$

Overall Scores The overall score for a model, denoted as $Avg.$, is calculated using Equation 5. In this equation, k represents the number of dimensions:

$$Avg. = \frac{1}{k} \sum_{l=1}^{k} DimScore_l \qquad (5)$$

Syntax Error Rate In addition to evaluating the generation quality, we also calculated the syntax error rate (E_{syntax}) of the scripts generated by the model. The definition of a syntax error is whether the script generated by the model can successfully produce an image. The E_{syntax} is calculated using Equation 6. In this equation, N_{error} stands for the number of samples with syntax error, N_{total} stands for the total number of samples:

$$E_{syntax} = \frac{N_{error}}{N_{total}} \times 100\% \qquad (6)$$

Standard Deviation To assess the consistency of the model's outputs, we calculate the $Standard\ Deviation\ (SD)$ of the scores across k dimensions, as shown in Equation 7.

$$SD = \sqrt{\frac{\sum_{l=1}^{k} (DimScore_l - Avg.)^2}{k}} \qquad (7)$$

E Benchmark Evaluation Process

For a detailed description of the scoring process, please refer to Figure 9.

F Details of The Data Generation Pipeline

The detailed iterative generation are shown in Algorithm 1. The generation prompt is showed in Figure 5.
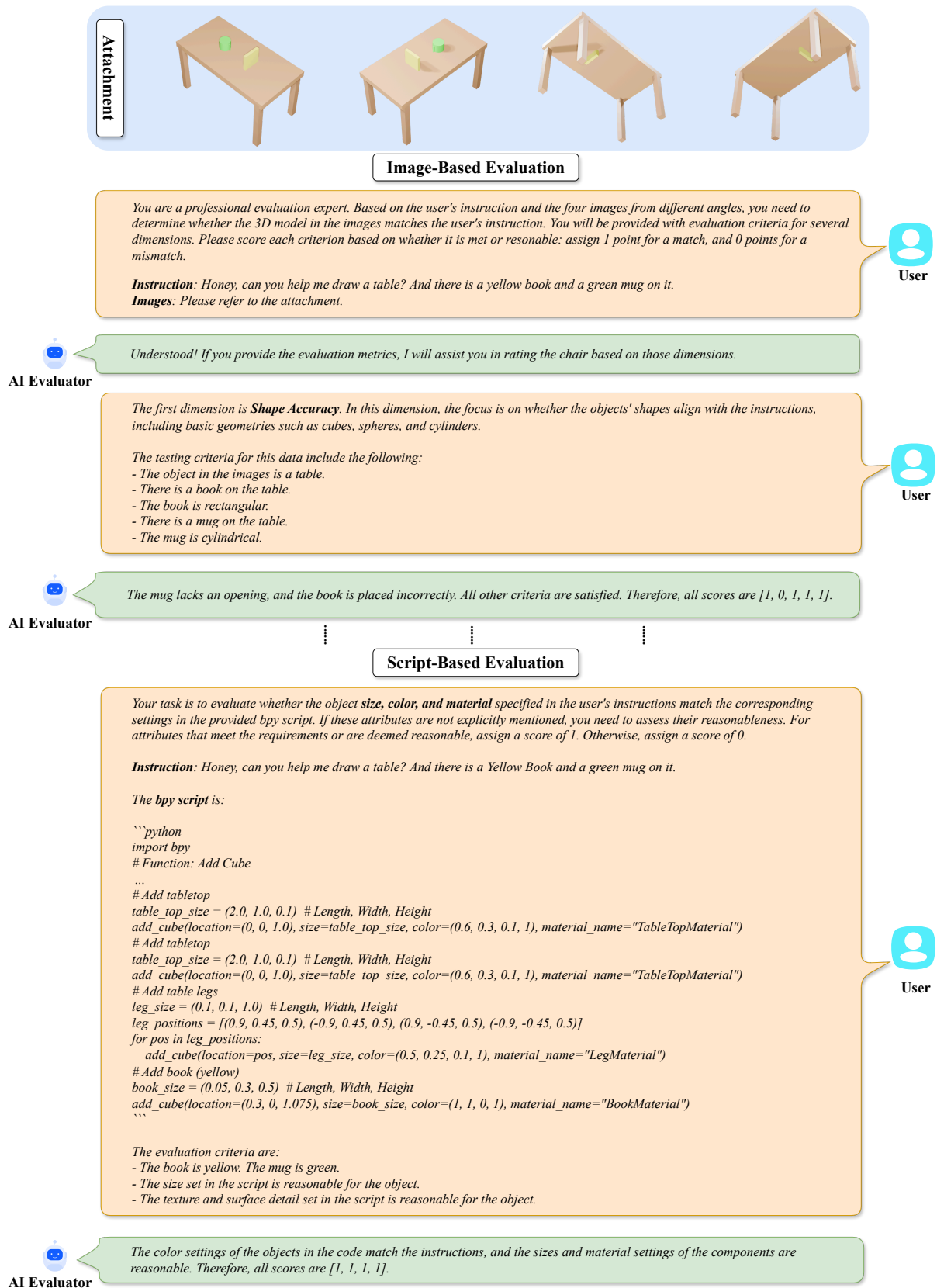
Attachment

Image-Based Evaluation

You are a professional evaluation expert. Based on the user's instruction and the four images from different angles, you need to determine whether the 3D model in the images matches the user's instruction. You will be provided with evaluation criteria for several dimensions. Please score each criterion based on whether it is met or resonable: assign 1 point for a match, and 0 points for a mismatch.

Instruction: Honey, can you help me draw a table? And there is a yellow book and a green mug on it.
Images: Please refer to the attachment.

User

Understood! If you provide the evaluation metrics, I will assist you in rating the chair based on those dimensions.

AI Evaluator

*The first dimension is **Shape Accuracy**. In this dimension, the focus is on whether the objects' shapes align with the instructions, including basic geometries such as cubes, spheres, and cylinders.*

The testing criteria for this data include the following:
- The object in the images is a table.
- There is a book on the table.
- The book is rectangular.
- There is a mug on the table.
- The mug is cylindrical.

User

The mug lacks an opening, and the book is placed incorrectly. All other criteria are satisfied. Therefore, all scores are [1, 0, 1, 1, 1].

AI Evaluator

⋮   ⋮   ⋮

Script-Based Evaluation

*Your task is to evaluate whether the object **size, color, and material** specified in the user's instructions match the corresponding settings in the provided bpy script. If these attributes are not explicitly mentioned, you need to assess their reasonableness. For attributes that meet the requirements or are deemed reasonable, assign a score of 1. Otherwise, assign a score of 0.*

Instruction: Honey, can you help me draw a table? And there is a Yellow Book and a green mug on it.

*The **bpy script** is:*

```python
import bpy
# Function: Add Cube
...
# Add tabletop
table_top_size = (2.0, 1.0, 0.1)  # Length, Width, Height
add_cube(location=(0, 0, 1.0), size=table_top_size, color=(0.6, 0.3, 0.1, 1), material_name="TableTopMaterial")
# Add tabletop
table_top_size = (2.0, 1.0, 0.1)  # Length, Width, Height
add_cube(location=(0, 0, 1.0), size=table_top_size, color=(0.6, 0.3, 0.1, 1), material_name="TableTopMaterial")
# Add table legs
leg_size = (0.1, 0.1, 1.0)  # Length, Width, Height
leg_positions = [(0.9, 0.45, 0.5), (-0.9, 0.45, 0.5), (0.9, -0.45, 0.5), (-0.9, -0.45, 0.5)]
for pos in leg_positions:
    add_cube(location=pos, size=leg_size, color=(0.5, 0.25, 0.1, 1), material_name="LegMaterial")
# Add book (yellow)
book_size = (0.05, 0.3, 0.5)  # Length, Width, Height
add_cube(location=(0.3, 0, 1.075), size=book_size, color=(1, 1, 0, 1), material_name="BookMaterial")
```

The evaluation criteria are:
- The book is yellow. The mug is green.
- The size set in the script is reasonable for the object.
- The texture and surface detail set in the script is reasonable for the object.

User

The color settings of the objects in the code match the instructions, and the sizes and material settings of the components are reasonable. Therefore, all scores are [1, 1, 1, 1].

AI Evaluator

Figure 9: Model Evaluation Process.

G Human Annotation

G.1 Annotation of BlendNet-Human

G.1.1 Objective

Evaluate the quality of `<Instruction, Script, Images>` data by ensuring alignment between images, instructions, and scripts to construct the `BlendNet-Human`.

G.1.2 Annotation Guidelines

- **Image-Instruction Alignment:** Images must correspond to the instructions regarding component position, proportion, and specified conditions (e.g., symmetry, rotation, spatial relationships).

- **Script-Instruction Alignment:** Scripts should accurately implement attributes described in the instructions, such as colors, sizes, materials, and other properties not visible in the images.

G.1.3 Annotation Workflow

1. **Initial Review:** Two annotators independently evaluate each entry, recording `pass/fail` decisions along with reasons for any failures.

2. **Discrepancy Resolution:** A third annotator resolves any disagreements between the initial two annotators.

3. **Quality Control:** A QC team reviews 30% of the data to ensure adherence to guidelines, refining the process based on feedback.

G.1.4 Team and Results

- **Annotators:** 12 annotators for initial reviews and 3 annotators for arbitration and quality control.

- **Scale:** Over $10k$ entries were reviewed, resulting in $2k$ entries for `BlendNet-Human`.

G.2 Annotation of Criteria

G.2.1 Objective

Construct the reliable Criteria for `CADBech` by filtering and modifying $2.5k$ `<Instruction, Criteria>` pairs to ensure consistency and feasibility.

G.2.2 Annotation Guidelines

Instruction Filtering

- **Relevance and Feasibility:** Instructions must describe feasible and logically sound tasks, excluding ambiguous or unrealistic ones.

- **Material, Surface, and Complexity Constraints:** Instructions with multiple constraints for material, surface details and internal complexity should be simplified to retain only one reasonable requirement.

- **Scope Alignment:** Remove instructions unrelated to the test dataset's goals.

Criteria Validation

- **Comprehensiveness:** Criteria must cover all dimensions and sub-dimensions.

- **Specificity:** Replace ambiguous terms with measurable criteria.

- **Default for Unspecified Dimensions:** Add default criteria for missing properties (e.g., "color palette should be harmonious").

G.2.3 Annotation Workflow

1. **Initial Review:** Two annotators independently assess each `<Instruction, Criteria>` pair, recording decisions and flagging unreasonable data.

2. **Discrepancy Resolution:** A third annotator resolves disagreements and finalizes the annotations.

3. **Quality Control:** A QC team reviews 30% of the data to ensure adherence to guidelines, refining the process based on feedback.

G.2.4 Team and Results

- **Annotators:** 3 annotators for the annotation process and 1 members in the quality control team.

- **Results:** From the initial $2.5k$ entries, 500 high-quality `<Instruction, Criteria>` pairs were curated.

G.3 Annotation of Evaluation

G.3.1 Objective

Obtain human preferences for evaluating the quality of the model's outputs by scoring the results of 200 model responses

G.3.2 Scoring Guidelines

Scoring Process

- **1 point (pass)** if the criterion is satisfied.

- **0 points (fail)** if the criterion is not satisfied.

Scoring Criteria

1. Image-Based Evaluation By comparing the images with the requirements in the instruction, evaluate whether the criteria for all sub-dimensions, except for Color, Size, Texture, and Surface Detail, are met.

2. Script-Based Evaluation By comparing the script with the requirements in the instruction, evaluate whether the criteria for Color, Size, Texture and Surface Detail, are met.

3. Default Scoring for Unspecified Properties

- Assign **1 point** if the script logically and harmoniously defines the property.

- Assign **0 points** if the property appears inconsistent or unreasonable.

G.3.3 Annotation Workflow

1. **Data Assignment:** Annotators are assigned all of `<Instruction, Script, Images>` entries (four images per entry).

2. **Scoring and Justification:** Annotators score each criterion and provide explanations for any failing scores.

3. **Quality Control:** A QC team reviews 30% of the data to ensure compliance with guidelines, refining the process based on feedback.

G.3.4 Team and Results

- **Annotators:** 3 scoring annotators and 1 quality control annotators.

- **Results:** The *kappa* value, calculated to reflect the consistency between human evaluators, is 0.883.

H Baseline LLMs

Details about the baseline LLMs are shown below:

- **o1-Preview** (Team, 2024): O1-Preview is a version of OpenAI's O1 model. It provides enhanced efficiency and accuracy for diverse applications, delivering high-performance results with optimized capabilities.

- **GPT-4 turbo** (OpenAI, 2023): GPT-4 Turbo is a version of OpenAI's GPT-4 model. It offers improved performance in responses for a wide range of applications.

- **Claude3.5-sonnet** (Anthropic, 2024): A model developed by Anthropic, known for its safety and alignment features in language generation tasks.

- **GPT-4o** (OpenAI, 2024):GPT-4o is a language model developed by OpenAI that can generate human-like text based on the input it receives.

- **BlenderGPT** (Aarya, 2023): A model developed by Aarya and Flip Phillips, which allows user to use natural language commands to control Blender. It leverages GPT-3.5 (Brown et al., 2020) or GPT-4 (OpenAI, 2023) to generate corresponding bpy scripts based on user-defined prompts for rendering 3D models.

- **Gemini-1.5-pro** (Gemini Team, 2024): Gemini 1.5 is an advanced AI language model developed by Google DeepMind.

- **DeepSeek-V2.5** (Liu et al., 2024): DeepSeek-V2.5 is an advanced language model designed for information retrieval tasks, optimized for search accuracy and efficiency across large datasets.

- **Qwen-2.5-Coder-7B-Instruct** (Hui et al., 2024): Qwen2.5-Coder is the latest series of Code-Specific Qwen large language models

- **Qwen-2.5** (Hui et al., 2024): Qwen-2.5 is a versatile language model that excels in natural language understanding and generation, providing improved context comprehension and response accuracy.

- **LLaMA3** (Touvron et al., 2023): The latest version of the LLaMA model, which has been fine-tuned for a variety of natural language processing tasks.

- **Mistral-7B-Instruct-V0.3** (Jiang et al., 2023): Mistral-7B-Instruct-V0.3 is a highly scalable model known for its performance in both text generation and comprehension tasks, utilizing 8-layer attention mechanisms with a 7B parameter architecture for enhanced processing.

- **CodeLLaMa-7B-Instruct** (Rozière et al., 2024): Code Llama is a collection of pre-trained and fine-tuned generative text models ranging in scale from 7 billion to 34 billion parameters.

I Performance on Sub-Dimensions

The performance of different LLMs on Sub-Dimension is shown in Figure 10.

J Visual Performance of Different Models

The Visual Examples of the Performance of Different Models are shown in Table 9.

K Visual Performance of Different Training strategy

The Visual Examples of the Performance of Different Training strategy are shown in Table 10.

L Characteristics of Annotators

The annotators involved in this study possess the following characteristics:

- Bachelor's degree in one of the following fields: Computer Science, Data Science, Business Administration, English, Music, or Biological Sciences.

- Full English instruction during their academic education.

M AI Assistant

Some of the text has been polished and revised by GPT-4, but the main part is completed by humans.
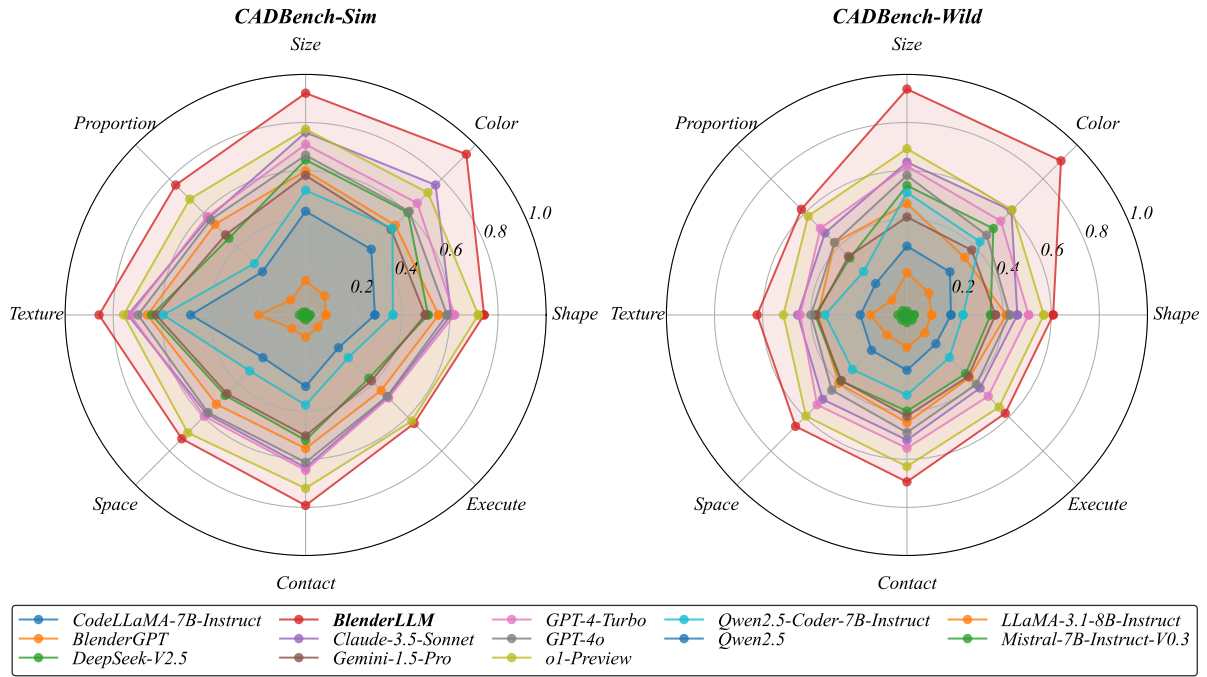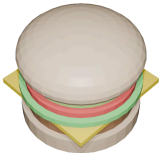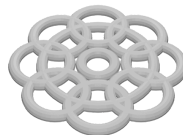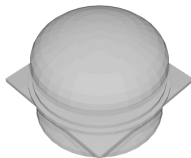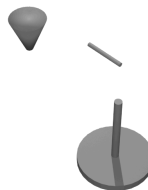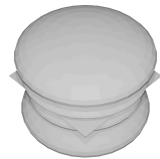
Figure 10: Performance of different LLMs on Sub-Dimensions

Table 9: The Visual Examples of the Performance of Different Models

Models	Dimension					
	Atrr.		*Spat.*		*Inst.*	
	Instruction					
	Create a 3D model of a burger. It consists of a sesame seed bun, a beef patty, a slice of cheese, lettuce, tomato, and pickles.		*I need better lighting on my desk and want a functional and stylish desk lamp, would you be able to give me some functional and stylish construction?*		*Design a 3D model of a Celtic knot. The knot should be intricate, with interlocking loops and a continuous pattern. Ensure the design is symmetrical and has a traditional Celtic feel.*	
	Images	**Scores**	**Images**	**Scores**	**Images**	**Scores**
BlenderLLM		**1.0**		**1.0**		**1.0**
o1-Preview		0.8		0.4		0
GPT-4-Turbo		0.8		0.4	*Syntax Error*	0

Table 9: The Visual Examples of the Performance of Different Models (Continued)
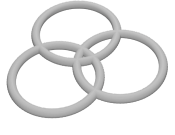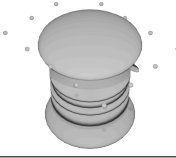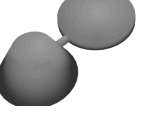
Model	Image 1	Score 1	Image 2	Score 2	Image 3	Score 3
Claude-3.5-Sonnet	*Syntax Error*	0		0.2	*Syntax Error*	0
GPT-4o		0.6		0.8		0.5
BlenderGPT		0.5		0.8	*Syntax Error*	0
Gemini-1.5-Pro		0.5		0.2	*Syntax Error*	0
DeepSeek-V2.5	*Syntax Error*	0		0.2	*Syntax Error*	0
Qwen2.5-Coder-7B-Instruct		0.2		0	*Syntax Error*	0
Qwen2.5		0.2	*Syntax Error*	0	*Syntax Error*	0
LLaMA-3.1-8B-Instruct	*Syntax Error*	0	*Syntax Error*	0	*Syntax Error*	0
Mistral-7B-Instruct-V0.3	*Syntax Error*	0	*Syntax Error*	0	*Syntax Error*	0
CodeLLaMA-7B-Instruct	*Syntax Error*	0	*Syntax Error*	0	*Syntax Error*	0

Table 10: The Visual Examples of the Performance of Different Training strategy

Instruction: *Can you help me to draw a chair? It has regular legs, a square seat and a square back with yellow stripes.*		
Self-improvement Training	Epoch Accumulation Training	Predefined Incremental Training