CircuitPile: A Data Synthesis Framework for Enhancing Multimodal LLMs' Perception on Circuit Schematics

Anonymous ACL submission

Abstract

In recent years, multimodal large language models have achieved substantial progress, yet their performance remains limited in specialized domains such as circuit diagram understanding. A key bottleneck lies in the lack of diverse and large-scale training data tailored to the structural characteristics of circuit diagrams, together with the high cost of expert annotation. To address this challenge, we propose CircuitPile, a systematic data synthesis framework for circuit visual-language modeling. CircuitPile integrates heterogeneous circuit resources from multiple sources and constructs standardized label spaces and multi-granularity visual representations. Based on this framework, we build 738K domain-specific pre-training samples and 100K task-oriented instruction-tuning samples for the pre-training and supervised fine-tuning stages, respectively. Furthermore, to systematically evaluate circuit diagram understanding, we introduce CircuitPerception-Bench, a fine-grained benchmark comprising 2,700 samples across three dimensions: component recognition, spatial understanding, and quantitative reasoning. Experimental results show that multimodal models trained with CircuitPile achieve significant performance gains on the internal benchmark and demonstrate favorable transferability on the external public AMS Benchmark. For example, when applying CircuitPile-based PT+SFT to Qwen3-VL-8B, the scores on CircuitPerception-Bench and AMS Benchmark improve from 0.5759 and 0.5067 to 0.7485 and 0.5973, respectively.

1 Introduction

Circuit diagrams serve as important information carriers in electronic system design, engineering education, and technical documentation, describing component composition, connection relationships, signal propagation, and overall functional structures. Unlike natural images, circuit diagrams exhibit strong symbolic and structural characteristics: they contain textual information such as component symbols, names, and identifiers, as well as structural relations formed by wires, layouts, and local topologies. Therefore, circuit diagram understanding requires not only recognizing objects in the diagram, but also modeling the correspondence between textual information and components, spatial organization, and structural associations. Prior studies have shown that circuit diagram understanding plays an important role in tasks such as automatic drawing processing, schematic analysis, and netlist generation (Bayer et al., 2024; Huang et al., 2025; Aldowaish et al., 2026).

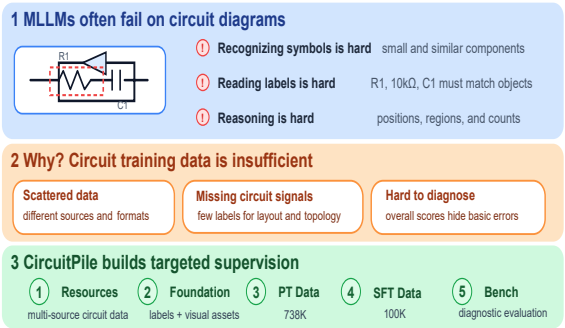


Figure 1: Motivation of CircuitPile. CircuitPile addresses the limitations of general MLLMs on circuit diagrams through circuit-specific data synthesis, stage-aware supervision, and diagnostic evaluation.

In recent years, multimodal large language models have achieved substantial progress in tasks such as visual question answering, vision-language understanding, and complex reasoning (Liu et al., 2024b; Bai et al., 2025). However, circuit diagrams contain fine-grained component symbols with subtle inter-class differences, textual attributes that are tightly bound to specific objects, and visual semantics jointly determined by quanti-

ties, positions, and local structures. As a result, general-purpose MLLMs still face challenges in this scenario, including unstable recognition, insufficient attribute grounding, biased spatial localization, and limited structural reasoning. Existing models therefore remain inadequate for circuit visual question answering, domain-specific perception, and complex structural analysis (Mehta et al., 2024).

Although prior studies have accumulated various data resources for component detection and symbol recognition (Thoma et al., 2021; Roy et al., 2025; Bhanbhro et al., 2023), as well as netlist recovery and circuit question answering (Xu et al., 2025; Shi et al., 2025), most of these resources are designed for specific tasks and remain difficult to directly support the systematic training of multimodal large language models in the circuit domain, as illustrated in Figure 1.

To this end, we propose **CircuitPile**, a systematic data synthesis framework for circuit visual-language modeling. It integrates heterogeneous circuit resources from multiple sources, constructs a standardized label space and multi-granularity visual representations, and further builds domain-specific pre-training corpora and task-oriented instruction-tuning corpora. Meanwhile, we introduce **CircuitPerception-Bench**, a diagnostic benchmark that evaluates models along three dimensions: component recognition, spatial understanding, and quantitative reasoning.

The main contributions of this paper are summarized as follows:

1. We propose CircuitPile, a systematic data synthesis framework for circuit visual-language modeling, based on which we construct 738K PT samples and 100K SFT samples.

2. We introduce CircuitPerception-Bench, a diagnostic evaluation benchmark for fundamental circuit diagram understanding, which assesses models from three dimensions: component recognition, spatial understanding, and quantitative reasoning.

3. We conduct systematic experiments to validate the effectiveness of CircuitPile. The results show that CircuitPile improves the circuit diagram understanding capabilities of open-source multimodal models and demonstrates favorable transfer performance on the external AMS Benchmark (Shi et al., 2025).

2 Related Work

2.1 MLLMs for Specialized Visual Understanding

In recent years, multimodal large language models have achieved substantial progress in image captioning, visual question answering, and complex visual reasoning by aligning visual encoders with large language models. Representative works such as LLaVA, Qwen-VL, and InternVL combine large-scale image-text pre-training with visual instruction tuning, improving open-ended visual dialogue and cross-task generalization capabilities (Liu et al., 2023; Bai et al., 2023; Chen et al., 2024). Subsequent studies have further enhanced high-resolution image understanding, text-rich visual parsing, and fine-grained multimodal reasoning (Wang et al., 2024; Bai et al., 2025; Zhu et al., 2025).

As model capabilities continue to advance, MLLMs have gradually been extended to specialized visual scenarios such as documents, charts, and medical images. Related studies have shown that constructing domain-specific image-text corpora or instruction data can improve model performance in professional content understanding and reasoning tasks (Ye et al., 2023; Han et al., 2023; Liu et al., 2024a; Ahmed Masry, 2024; Moor et al., 2023; Li et al., 2023; Cui et al., 2024). In contrast, circuit diagrams simultaneously involve symbols, text, and structural relations, making it difficult for existing general-purpose MLLMs to adapt directly. Systematic research on domain-specific visual-language modeling for circuit diagrams remains limited.

2.2 Circuit Diagram Understanding and Datasets

Circuit diagram understanding has long attracted attention in electronic design automation, engineering document parsing, and image-to-structure conversion. Prior studies have constructed various data resources for component detection, symbol recognition, handwritten circuit digitization, and graph-structure extraction (Thoma et al., 2021; Ahmed et al., 2025; Bhanbhro et al., 2023). Another line of research further focuses on circuit structure parsing and netlist recovery, exploring automatic conversion from schematics to structured circuit representations (Tao et al., 2024; Al-

dowaish et al., 2026; Bhandari et al., 2024).

With the development of multimodal large language models, circuit diagram research has gradually expanded to visual question answering and comprehensive evaluation. CircuitVQA and ElectroVizQA construct question-answering data for circuit images, while AMS Benchmark, CIRCUIT, EEE-Bench, and MMCircuitEval evaluate model performance on circuit and electronic engineering tasks from a broader perspective (Meshram et al., 2024; Skelic et al., 2025; Li et al., 2025; Zhao et al., 2025). CircuitVQA contains 5,725 circuit diagrams and 115K QA pairs, covering basic recognition and counting tasks; however, it still suffers from shallow task hierarchies, relatively simple images, unstable annotation quality, and a lack of domain-specific pre-training data. ElectroVQA contains only 626 samples, making it extremely small in scale, and its scenarios are limited to basic digital circuit questions. Moreover, the distribution gap between its synthetic images and real industrial schematics makes it difficult to support systematic model training and the validation of generalization ability. Overall, existing datasets remain limited in scale, narrow in task coverage, and insufficient in real-world scenario diversity, making them inadequate for supporting the progressive training of multimodal models from basic semantics to advanced circuit reasoning.

3 CircuitPile Generation

3.1 Data Synthesis Framework

Figure 2 illustrates the overall construction pipeline of CircuitPile. The pipeline starts from heterogeneous circuit resources collected from multiple sources, including printed schematics, handwritten circuit diagrams, PCB and real-world component images, as well as structured or textual annotations. Since different data sources vary in symbol styles, label naming conventions, annotation granularity, and forms of auxiliary information, we first perform label unification, attribute standardization, and multi-source annotation alignment. We further construct multi-granularity visual assets, including cropped single-component images, target-highlighted images, and multi-component regional images, thereby establishing a unified data foundation.

On top of the unified data foundation, CircuitPile constructs PT and SFT corpora for two-stage training. The PT corpus supplements circuit-specific visual-language priors, enabling models to establish fundamental domain representations, while the SFT corpus transforms such representations into task-oriented response capabilities through question-answer supervision. The data organization, statistics, and quality control of CircuitPile are provided in Appendix A.6 and Appendix A.7.

3.1.1 Data Corpus Collection

CircuitPile integrates multiple publicly available circuit-related datasets as its underlying data sources, covering printed schematics, handwritten circuit diagrams, PCB scene images, real-world electronic component images, and structured or textual annotations such as netlists, component lists, and connection relationships. Specifically, handwritten circuit diagram data are mainly drawn from resources such as CGHD, Digitize-HCD, and JUHCCR-v1 (Thoma et al., 2021; Ahmed et al., 2025; Roy et al., 2025); printed schematics and structured circuit resources are primarily collected from datasets such as AMSNet and Image2Net (Tao et al., 2024; Xu et al., 2025); and visual data for electronic component detection and PCB-related scenarios include ElectroCom61, PCB surface defect/component detection datasets, and several publicly available object detection resources (Sayeedi et al., 2025; Lv et al., 2024; Diana, 2024; EZpark, 2024; project-gtqqq, 2022; Dataset Ninja, 2021). The specific types, scales, and usages of each dataset are provided in Appendix A.1.

Since different public datasets vary in category naming, label granularity, and annotation formats, we unify component labels through label mapping and construct a standardized label space. Details of label normalization, category merging, and abnormal annotation filtering are provided in Appendix A.2, while the construction details of multi-granularity visual assets are provided in Appendix A.3.

3.1.2 PT Data Generation

As shown in Figure 3, it contains approximately 738K image-text samples. This corpus consists of five complementary types of image-text samples. At the local level, single-component image-text pairs help the model learn the correspondence between components and basic information such as categories, functions, names, and identifiers. At the compositional level, multi-component merged image-text pairs enable the model to perceive com-

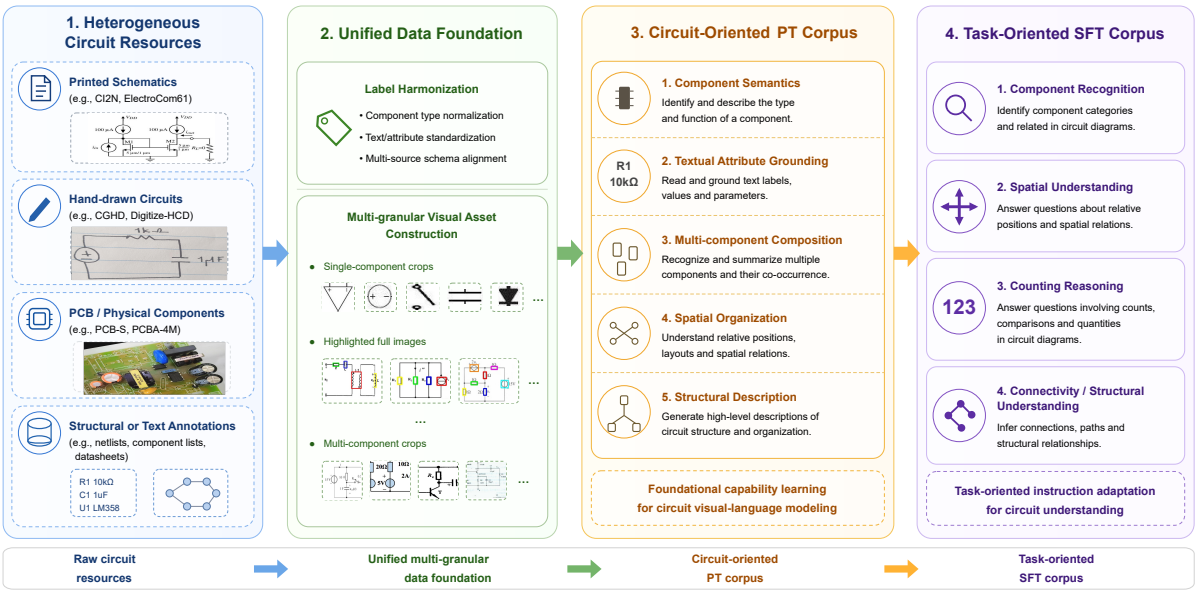


Figure 2: Overall architecture of CircuitPile. CircuitPile first integrates circuit data from multiple sources into a standardized data foundation, and then constructs circuit-oriented PT data and task-oriented SFT data to support the progressive domain adaptation of multimodal LLMs.

ponent distributions, quantity summaries, and layout patterns. At the structural level, programmatically synthesized structural description data further introduce branch structures and connectivity relationships, providing supervision for topological understanding of circuit diagrams. The detailed synthesis procedures are provided in Appendix A.4.

3.1.3 SFT Data Generation

First, according to the requirements of different tasks, we construct multi-granularity input forms, including cropped single-component images, target-highlighted images, multi-target-highlighted images, and multi-component regional images.

Next, the candidate SFT instruction pool constructs VQA pairs according to four types of specific tasks. The first type is component recognition, where the model identifies the category of a component in a given image or target region. The second type is spatial understanding and relational reasoning, with questions involving target localization, position retrieval, relative direction judgment, and nearest or farthest relationship reasoning. The third type is quantitative reasoning, including component counting, regional counting, conditional counting, quantity comparison, and extremum judgment. The fourth type is structural relation and circuit semantic understanding, where questions involve whole-diagram description, target-component description, netlist-related tasks, and direct-connection question answering.

Finally, we construct approximately 100K balanced SFT samples from the original instruction-task pool. This strategy reduces the dominance of high-frequency tasks in the training distribution, enabling the model to achieve more balanced improvements across component recognition, spatial reasoning, quantitative reasoning, and structural understanding.

The detailed construction pipeline and implementation details are provided in Appendix A.5.

4 CircuitPerception-Bench

We construct **CircuitPerception-Bench**, a diagnostic evaluation benchmark for circuit diagram understanding, as shown in Figure 4.

4.1 Benchmark Design

Existing circuit-related evaluation resources, such as AMS Benchmark, offer broad coverage and are suitable for assessing the overall performance of models in complex electronic engineering scenarios. However, some of their tasks are more oriented toward domain knowledge question answering, textual understanding, or high-level functional analysis, making it difficult to directly identify model weaknesses in fundamental perception and reasoning abilities. In contrast,

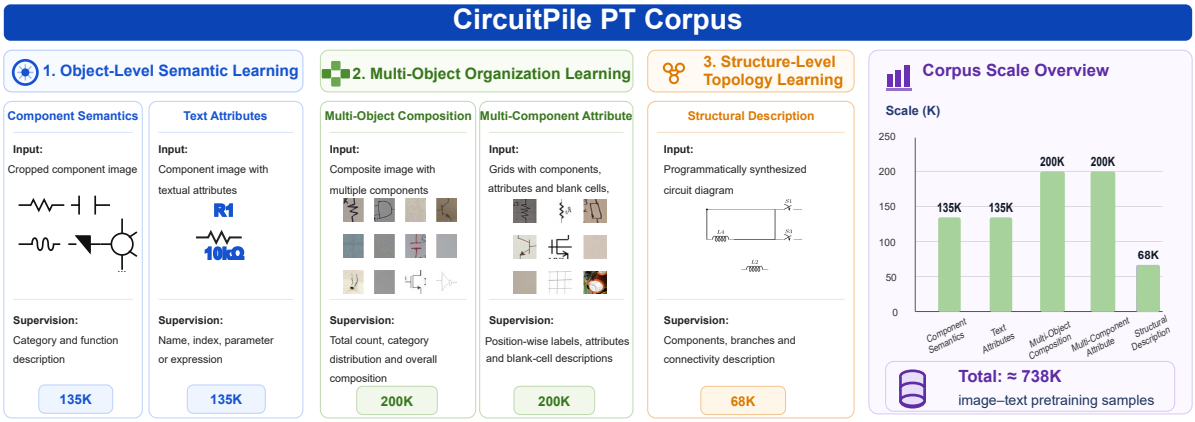


Figure 3: Overview of the CircuitPile pre-training corpus. The corpus consists of five complementary types of image-text supervision, including component semantics, textual attributes, multi-object composition, spatial organization, and structural descriptions, progressively supporting circuit-domain representation learning from local symbol perception to global structural understanding.

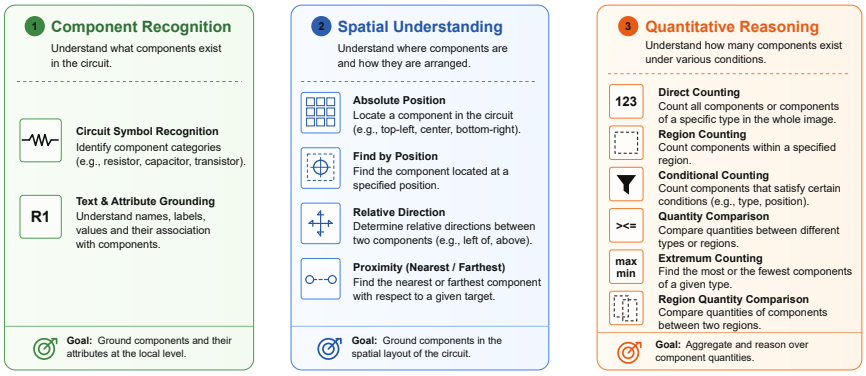


Figure 4: Overview of the CircuitPerception-Bench benchmark. This benchmark is organized around three core capabilities: component recognition, spatial understanding, and quantitative reasoning. It evaluates a model's ability to understand circuit symbols, object positional relationships, and the quantitative structure of multiple components.

CircuitPerception-Bench emphasizes the direct dependence of each task on the visual content of circuit diagrams and focuses on three key dimensions, requiring models to answer based on the symbols, positions, and quantity information in the image, as shown in Figure 5.

4.2 Benchmark Construction

As shown in Figure 4, the benchmark diagnoses three fundamental capabilities. **Component Recognition** evaluates category identification of local circuit symbols using cropped single-component images to minimize background interference. **Spatial Understanding** assesses target grounding, relative positioning, and geometric relationships through single-target highlighted, multi-target annotated, and full-circuit images, covering absolute/relative location judgment, directional reasoning, and nearest/farthest inference. **Quantitative Reasoning** tests quantity aggregation, conditional filtering, and comparison across multi-component scenes, spanning component-level, regional, conditional, and total counting, as well as quantity comparison and extremum judgment. Fine-grained task definitions and their automatic VQA construction are detailed in Appendix B.

First, based on approximately 9.8K circuit images that have been manually annotated and verified, we apply a perceptual hashing algorithm to remove images that overlap with those in the SFT data, retaining only images that do not appear in the SFT corpus.

We construct multiple forms of visual inputs, including cropped single-component images to emphasize local symbolic features; single-target high-

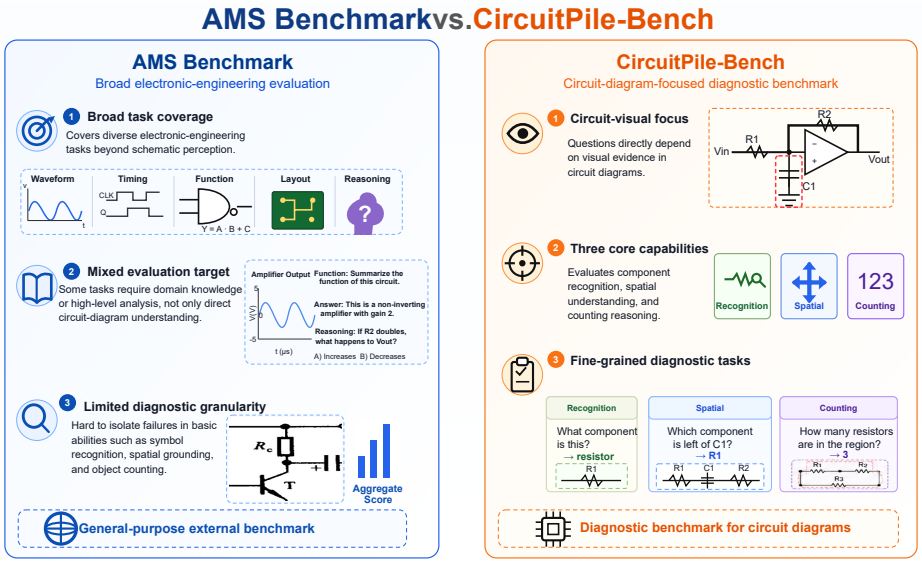


Figure 5: Comparison between AMS Benchmark and CircuitPerception-Bench. The former provides a broad external evaluation for electronic engineering tasks, whereas the latter focuses more on the diagnostic assessment of fundamental visual-language capabilities for circuit diagram understanding.

lighted images and multi-target annotated images for spatial understanding tasks, which explicitly indicate the queried object or candidate objects; and multi-component regional images for quantitative reasoning tasks, enabling the model to perform counting, comparison, and conditional aggregation within local regions or complete circuit diagrams. Ultimately, the candidate task pool contains approximately 45K single-component images, 45K single-target highlighted images, 9.8K multi-target annotated images, and 26K multi-component regional images.

Subsequently, We set sampling quotas for the three capability dimensions: Component Recognition, Spatial Understanding, and Quantitative Reasoning. Within each group, we apply round-robin balanced sampling across fine-grained tasks, visual input formats, and data sources, resulting in an evaluation set of 2,700 MCQ samples.

4.3 Evaluation

During evaluation, models are required to output option labels, which are extracted by regular-expression matching. We use Accuracy for single-choice tasks and F1-score for multiple-choice tasks. When regular-expression extraction fails, an LLM is used for answer extraction. Details of the evaluation prompts, extraction rules, and abnormal-output handling are provided in Appendix C.

5 Experiments

To validate the effectiveness of CircuitPile, we conduct three groups of experiments. First, we evaluate the impact of domain-specific supervision on fundamental circuit visual-language capabilities using CircuitPerception-Bench. Second, we examine its cross-task transfer performance on the external AMS Benchmark. Finally, using a representative model as the study object, we compare direct instruction tuning with the "domain pre-training–instruction tuning" training pipeline to analyze the effect of pre-training corpora on the distribution of model capabilities.

5.1 Experimental Setup

We adopt Qwen2.5-VL-7B, Qwen3-VL-2B/4B, and InternVL3.5-2B/4B/8B as base models, comparing each with its CircuitPile-trained counterpart (denoted CircuitPile-VLM). We also report results from Gemini 3.1 Pro Preview (Google, 2026), a leading closed-source multimodal model, as a reference. To analyze the role of pre-training, we further ablate on Qwen3-VL-8B, comparing the original backbone against SFT-only and PT+SFT variants.

Evaluation employs two complementary benchmarks. CircuitPerception-Bench assesses three fundamental circuit visual-language capabilitiescomponent recognition, spatial understanding, and quantitative reasoningreporting Accuracy on Overall (all 2,700 samples), Recognition, Position,

and Counting subsets. AMS Benchmark evaluates transferability to external complex circuit tasks, using Accuracy for single-choice, judgment, and numerical tasks, and F1-score for multiple-choice tasks.

During training, we set the learning rate to 1×10^{-5}, the per-GPU batch size to 2, and use 32 steps of gradient accumulation. The maximum sequence length is set to 2048, and the maximum number of pixels per image is set to 518,400. To reduce memory consumption and improve training stability, we adopt bf16 precision, gradient checkpointing, and DeepSpeed ZeRO-2 optimization. In the pre-training stage, we freeze all parameters except the MLP, while in the SFT stage, we perform full-parameter fine-tuning. All training experiments are conducted on 8 NVIDIA A800 GPUs.

5.2 Effectiveness of CircuitPile

5.2.1 Results on CircuitPerception-Bench

Table 1 reports the results of different models on CircuitPerception-Bench. Models trained with CircuitPile consistently achieve higher overall accuracy, indicating that domain-specific supervision can steadily improve the fundamental circuit diagram understanding capabilities of most open-source multimodal models. Models with particularly large improvements include Qwen2.5-VL-7B, InternVL3.5-4B, and InternVL3.5-8B, whose Overall scores increase from 0.5396, 0.4059, and 0.3852 to 0.7848, 0.7026, and 0.7322, respectively.

From the perspective of capability dimensions, CircuitPile brings the most significant gains in component recognition. For example, the Recognition score of Qwen2.5-VL-7B increases from 0.5800 to 0.8667, while that of InternVL3.5-8B improves from 0.2940 to 0.7947. Counting and Position also show clear improvements for most models, indicating that the effect of CircuitPile is not limited to local symbol classification, but also extends to multi-object aggregation and spatial relation modeling.

Further qualitative experiments and case studies are provided in Appendix D.

5.2.2 Results on AMS Benchmark

Table 2 reports the results for Overall and 11 fine-grained tasks, where Overall denotes the average performance across all AMS subtasks. The results show that CircuitPile training brings stable overall transfer gains for most backbones. In terms of Overall performance, Qwen2.5-VL-7B improves from 0.4574 to 0.5609, Qwen3-VL-4B from 0.4953 to 0.5944, InternVL3.5-4B from 0.3873 to 0.5044, and InternVL3.5-8B from 0.4662 to 0.5308. These results indicate that the circuit visual-language capabilities learned from CircuitPile not only improve performance on the internal CircuitPerception-Bench, but also transfer effectively to external AMS tasks.

The gains are mainly observed in structure- or semantics-related tasks, such as Connection Identification, Element Classification, Partition, Reasoning, and Total Counting. For example, Qwen2.5-VL-7B improves from 0.4130, 0.2919, and 0.5743 to 0.5961, 0.5624, and 0.6980 on Connection Identification, Partition, and Reasoning, respectively. These results suggest that the fundamental visual semantics, spatial organization, and quantity understanding enhanced by CircuitPile can be transferred to more complex external circuit tasks.

Model	Overall	Count.	Spa.	Rec.
Qwen2.5-VL-7B	0.5396	0.5050	0.4733	0.5800
+CircuitPile-SFT	**0.7848**	**0.7450**	**0.6200**	**0.8667**
Qwen3-VL-2B	0.5226	0.4750	0.5383	0.5353
+CircuitPile-SFT	**0.6881**	**0.6317**	**0.5833**	**0.7527**
Qwen3-VL-4B	0.5563	0.4500	0.4467	0.6427
+CircuitPile-SFT	**0.7552**	**0.7217**	**0.6933**	**0.7933**
Qwen3-VL-8B	0.5759	0.4733	0.5567	0.6247
+CircuitPile-SFT	**0.7703**	**0.7267**	**0.7031**	**0.8147**
InternVL3.5-2B	0.4163	0.5150	0.4650	0.3573
+CircuitPile-SFT	**0.6593**	**0.5816**	**0.5300**	**0.7420**
InternVL3.5-4B	0.4059	0.3567	0.4383	0.4127
+CircuitPile-SFT	**0.7026**	**0.6817**	**0.6050**	**0.7500**
InternVL3.5-8B	0.3852	0.4317	0.5667	0.2940
+CircuitPile-SFT	**0.7322**	**0.6867**	**0.6217**	**0.7947**
Gemini-3.1-pro	0.9080	0.8820	0.8750	0.9500

Table 1: Performance comparison between raw backbones and their CircuitPile-trained counterparts on the CircuitPerception-Bench. Bold indicates the better result within each backbone pair.Count.=Counting, Spa.=Spatial Understanding, Rec.=Component Recognition. For subsequent tables, the same abbreviations are used unless otherwise noted.

5.3 Effect of Circuit-Oriented Pretraining

Table 3 shows that both direct instruction tuning and two-stage training substantially outperform the original model, achieving Overall scores of 0.7703 and 0.7485, respectively. SFT-only per-

Model	Overall	Caption	Conn. Id.	Conn. Judge.	Elect. Char.	Element Class.	Function	Location	Partition	Reasoning	Total Count.	Type Count.
Qwen2.5-VL-7B	0.4574	0.1928	0.4130	0.5711	0.6667	0.8174	0.6600	0.2511	0.2919	0.5743	0.1447	0.4480
+CircuitPile-SFT	**0.5609**	**0.3253**	**0.5961**	**0.6800**	**0.6833**	**0.8613**	**0.7300**	**0.2933**	**0.5624**	**0.6980**	**0.2773**	**0.4633**
Qwen3-VL-2B	0.3796	0.2530	0.4461	0.5889	0.5000	0.6793	**0.6175**	**0.2267**	0.3062	0.3317	0.0907	0.1360
+CircuitPile-SFT	**0.4701**	**0.3253**	**0.5248**	**0.6844**	**0.5500**	**0.8010**	0.4800	0.2111	**0.4676**	**0.5446**	**0.2007**	**0.3820**
Qwen3-VL-4B	0.4953	**0.5542**	0.3359	0.6433	0.6000	0.7264	**0.7275**	0.1756	0.3800	**0.7723**	0.1613	0.3713
+CircuitPile -SFT	**0.5944**	0.4940	**0.6248**	**0.7111**	**0.7500**	**0.8397**	0.6450	**0.3111**	**0.5994**	0.7673	**0.3153**	**0.4807**
Qwen3-VL-8B	0.5067	**0.5062**	0.3293	0.5720	**0.7674**	0.7833	**0.7605**	0.1802	0.3912	0.7676	0.1443	0.3758
+CircuitPile -SFT	**0.5988**	0.4782	**0.6135**	**0.7028**	0.7333	**0.8479**	0.6874	**0.3448**	**0.6017**	**0.7869**	**0.3048**	**0.4857**
InternVL3.5-2B	0.3623	0.2048	0.3826	**0.6344**	**0.6167**	0.5857	**0.4225**	0.1644	0.2996	0.3366	0.1047	0.2333
+CircuitPile-SFT	**0.4085**	**0.2410**	**0.4470**	0.5922	0.5500	**0.7323**	0.3500	**0.1822**	**0.3544**	**0.5446**	**0.1600**	**0.3393**
InternVL3.5-4B	0.3873	**0.3494**	0.4237	**0.6789**	**0.6667**	0.2009	0.4325	0.0800	0.3431	0.4802	0.1860	**0.4187**
+CircuitPile-SFT	**0.5044**	**0.3494**	**0.5224**	0.6700	0.6333	**0.8056**	**0.5750**	**0.2489**	**0.4662**	**0.6287**	**0.2453**	0.4033
InternVL3.5-8B	0.4662	**0.3976**	**0.5499**	**0.7167**	0.7000	0.8056	**0.7175**	0.1378	0.2866	0.1931	0.1640	**0.4593**
+CircuitPile-SFT	**0.5308**	0.3373	0.5323	0.6189	**0.8000**	**0.8469**	0.5700	**0.2733**	**0.5037**	**0.6980**	**0.2400**	0.4180

Table 2: Overall and task-level results on the AMS Benchmark. Accuracy is reported for single-choice, binary, and numerical tasks, while F1-score is reported for multi-choice tasks. Bold indicates the better result between the raw backbone and its CircuitPile-trained counterpart.

Model	Overall	Count.	Spa.	Rec.
Qwen3-VL-8B	0.5759	0.4733	0.5567	0.6247
SFT-only	**0.7703**	**0.7267**	**0.7031**	0.8147
PT+SFT	0.7485	0.6943	0.6073	**0.8267**

Table 3: Overall and capability-level results under different training paradigms on CircuitPerception-Bench. The overall score denotes accuracy over all test samples rather than the average of the three capability-level scores.

Model	Overall
Qwen3-VL-8B	0.5067
SFT-only	**0.5988**
PT+SFT	0.5973

Table 4: Overall and capability-level results under different training paradigms on AMS-Bench.

forms better on Overall, Counting, and Position, indicating that task-level supervision can more directly align with the evaluation format of the current benchmark. In contrast, PT+SFT achieves the best result on Recognition, suggesting that domain pre-training provides complementary benefits for modeling local symbol semantics.

Table 4 further reports the overall performance of the three models on the external AMS-Bench. Similar to the trend observed on CircuitPerception-Bench, both SFT-only and PT+SFT outperform the original Qwen3-VL-8B, reaching Overall scores of 0.5988 and 0.5973, respectively. This indicates that training with CircuitPile not only improves model performance on the internal diagnostic benchmark, but also brings certain cross-benchmark generalization gains. However, SFT-only slightly outperforms PT+SFT, suggesting that under the current training setting, the general circuit representations introduced by domain pre-training have not yet been fully converted into overall performance gains on external tasks.

6 Conclusion

This paper addresses the lack of systematic domain-specific foundational capabilities of multimodal large language models in circuit diagram understanding by proposing CircuitPile, a circuit visual-language data framework, and constructing CircuitPerception-Bench, a diagnostic evaluation benchmark. CircuitPile is centered on fundamental capabilities such as component semantics, spatial organization, quantitative relationships, and structural descriptions, forming a data framework that combines domain pre-training with instruction tuning. CircuitPerception-Bench further provides fine-grained evaluation from three dimensions: recognition, spatial understanding, and quantitative reasoning.

Experimental results show that training with CircuitPile significantly improves the fundamental circuit diagram understanding capabilities of most open-source multimodal models and demonstrates certain cross-task transfer effects on the external AMS Benchmark. This paper further suggests that improving circuit visual-language modeling depends not only on downstream task supervision, but also on systematic data design and evaluation diagnosis centered on domain-specific foundational representations. Future work can further extend the framework toward complex topological reasoning, functional analysis, and higher-level circuit semantic modeling.

7 Limitations

Although this study validates the effectiveness of systematic data synthesis in improving the circuit diagram understanding capabilities of multimodal large language models, several limitations remain and should be addressed in future work.

1. **Gap between data distribution and real industrial scenarios.** Although CircuitPile covers multiple types of data sources, including printed schematics, handwritten circuit diagrams, and textbook illustrations, its overall distribution is still biased toward standardized and clearly presented circuit diagrams. Complex cases in real industrial environments, such as severe occlusion, dense wiring, obsolete non-standard symbols, and scanning noise, remain insufficiently covered. The distribution gap between synthetic data and real schematics may limit the generalization ability of models in practical applications.

2. **Limited task coverage.** This work focuses on three types of fundamental perceptual capabilities: component recognition, spatial understanding, and quantitative reasoning. Higher-level circuit tasks, such as fault diagnosis, functional analysis, and design intent understanding, have not yet been systematically explored or validated.

3. **Preliminary structural supervision in the pre-training stage.** During pre-training, this work uses randomly synthesized circuit diagrams based on grids and predefined templates to help the model learn structural topology. However, the complexity of these synthetic circuits is limited, and there remains a gap between them and the complex connection patterns and hierarchical structures in real circuit diagrams. Stable gains from the pre-training stage are mainly observed in recognition-related tasks, while improvements in other capability dimensions are still not significant.

4. **Lower benchmark complexity than real-world scenarios.** CircuitPile-Bench is designed as a diagnostic evaluation focused on perceptual capabilities and adopts simplified visual input forms such as cropped images, target highlighting, and local regions. Compared with high-density, interference-rich, and unstructured layouts in real industrial schematics, the benchmark has relatively lower complexity, and its evaluation results may not fully reflect the challenges faced by models in practical applications.

5. **Limited model scale.** Due to computational resource constraints, this work only conducts experiments on small- to medium-scale open-source multimodal large language models ranging from 2B to 8B parameters, without evaluating larger models with 32B or more parameters, such as Qwen3-VL-72B and InternVL3.5-78B, on CircuitPile. Larger models may exhibit different patterns of capability improvement.

6. **Single evaluation format.** To enable automated, reproducible, and objective evaluation, CircuitPile-Bench adopts multiple-choice questions (MCQs) for all samples. More complex question formats, such as open-ended question answering, generative explanation, and multi-step reasoning, are not evaluated in this work. These formats may better reflect the interaction requirements in practical applications.

References

Nadim Ahmed, Mirza Fuad Adnan, Ahmad Shafiullah, Hayder Jahan Parash, Md. Saifur Rahman, Irfan Chowdhury Akib, and Golam Sarowar. 2025. Digitize-hcd: A dataset for digitization of handwritten circuit diagrams. *Data in Brief*, 59:111315.

Md Rizwan Parvez Enamul Hoque Shafiq Joty Ahmed Masry, Mehrad Shahmohammadi. 2024. Chartinstruct: Instruction tuning for chart comprehension and reasoning. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 10387–10409. Association for Computational Linguistics.

Saoud Aldowaish, Yashwanth Karumanchi, Kai-Chen Chiang, Soroosh Noorzad, and Morteza Fayazi. 2026. Sina: A circuit schematic image-to-netlist generator using artificial intelligence. *arXiv preprint arXiv:2601.22114*.

Anthropic. 2026. Claude sonnet 4.6. `Multimodallanguagemodel`. Accessed: 2026-04-25.

Jinze Bai, Shuai Bai, Shusheng Yang, Shijie Wang, Sinan Tan, Peng Wang, Junyang Lin, Chang Zhou,

and Jingren Zhou. 2023. Qwen-vl: A versatile vision-language model for understanding, localization, text reading, and beyond. *arXiv preprint arXiv:2308.12966*.

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yuanzhi Zhu, Mingkun Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, and 8 others. 2025. Qwen2.5-vl technical report. *arXiv preprint arXiv:2502.13923*.

Johannes Bayer, Leo van Waveren, and Andreas Dengel. 2024. Modular graph extraction for handwritten circuit diagram images. *arXiv preprint arXiv:2402.11093*.

Hina Bhanbhro, Kwang Hooi Yew, Worapan Kusakunniran, and Zaira Hassan Amur. 2023. A symbol recognition system for single-line diagrams developed using a deep-learning approach. *Applied Sciences*, 13(15):8816.

Jitendra Bhandari, Vineet Bhat, Yuheng He, Siddharth Garg, Hamed Rahmani, and Ramesh Karri. 2024. Auto-spice: Leveraging llms for dataset creation via automated spice netlist extraction from analog circuit diagrams. *arXiv preprint arXiv:2411.14299*.

Zhe Chen, Jiannan Wu, Wenhai Wang, Weijie Su, Guo Chen, Sen Xing, Muyan Zhong, Qinglong Zhang, Xizhou Zhu, Lewei Lu, Bin Li, Ping Luo, Tong Lu, Yu Qiao, and Jifeng Dai. 2024. Internvl: Scaling up vision foundation models and aligning for generic visual-linguistic tasks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Cheng Cui, Ting Sun, Suyin Liang, Tingquan Gao, Zelun Zhang, Jiaxuan Liu, Xueqing Wang, Changda Zhou, Hongen Liu, Manhui Lin, Yue Zhang, Yubo Zhang, Handong Zheng, Jing Zhang, Jun Zhang, Yi Liu, Dianhai Yu, and Yanjun Ma. 2025. Paddleocr-vl: Boosting multilingual document parsing via a 0.9b ultra-compact vision-language model. *Preprint*, arXiv:2510.14528.

Hejie Cui, Lingjun Mao, Xin Liang, Jieyu Zhang, Hui Ren, Quanzheng Li, Xiang Li, and Carl Yang. 2024. Biomedical visual instruction tuning with clinician preference alignment. *arXiv preprint arXiv:2406.13173*.

Dataset Ninja. 2021. Pcb component detection dataset. https://datasetninja.com/pcb-component-detection. Object detection dataset for PCB components.

Diana. 2024. circuitobjectsdetection dataset. https://universe.roboflow.com/diana-eottv/circuitobjectsdetection. Open Source Dataset, Roboflow Universe.

EZpark. 2024. Cv-circuit computer vision dataset. https://universe.roboflow.com/ezpark-gcxxl/cv-circuit. Open Source Dataset, Roboflow Universe.

Google. 2026. Gemini 3.1 pro preview. Multimodal language model. Accessed: 2026-04-21.

Yucheng Han, Chi Zhang, Xin Chen, Xu Yang, Zhibin Wang, Gang Yu, Bin Fu, and Hanwang Zhang. 2023. Chartllama: A multimodal llm for chart understanding and generation. *arXiv preprint arXiv:2311.16483*.

Chun-Yen Huang, Hsuan-I Chen, Hao-Wen Ho, Pei-Hsin Kang, Mark Po-Hung Lin, Wen-Hao Liu, and Haoxing Ren. 2025. Netlistify: Transforming circuit schematics into netlists with deep learning. In *2025 ACM/IEEE 7th Symposium on Machine Learning for CAD (MLCAD)*. IEEE.

Glenn Jocher, Ayush Chaurasia, and Jing Qiu. 2023. Ultralytics yolov8. Version 8.4.51. Accessed: 2026-05-25.

Chunyuan Li, Cliff Wong, Sheng Zhang, Naoto Usuyama, Haotian Liu, Jianwei Yang, Tristan Naumann, Hoifung Poon, and Jianfeng Gao. 2023. Llava-med: Training a large language-and-vision assistant for biomedicine in one day. *arXiv preprint arXiv:2306.00890*.

Ming Li, Jike Zhong, Tianle Chen, Yuxiang Lai, and Konstantinos Psounis. 2025. Eee-bench: A comprehensive multimodal electrical and electronics engineering benchmark. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13337–13349.

Fuxiao Liu, Xiaoyang Wang, Wenlin Yao, Jianshu Chen, Kaiqiang Song, Sangwoo Cho, Yaser Yacoob, and Dong Yu. 2024a. Mmc: Advancing multimodal chart understanding with large-scale instruction tuning. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1287–1310. Association for Computational Linguistics.

Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. 2024b. Improved baselines with visual instruction tuning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. 2023. Visual instruction tuning. In *Advances in Neural Information Processing Systems*, volume 36.

Shengping Lv, Bin Ouyang, Zhihua Deng, Tairan Liang, Shixin Jiang, Kaibin Zhang, Jianyu Chen, and Zhuohui Li. 2024. A dataset for deep learning based detection of printed circuit board surface defect. *Scientific Data*, 11(1):811.

Rahul Mehta, Bhavyajeet Singh, Vasudeva Varma, and Manish Gupta. 2024. Circuitvqa: A visual question answering dataset for electrical circuit images. In *Machine Learning and Knowledge Discovery in Databases. Research Track*, pages 440–460. Springer Nature Switzerland.

Pragati Shuddhodhan Meshram, Swetha Karthikeyan, Bhavya, and Suma Bhat. 2024. Electrovizqa: How well do multi-modal llms perform in electronics visual question answering? *arXiv preprint arXiv:2412.00102*.

Michael Moor, Qian Huang, Shirley Wu, Michihiro Yasunaga, Yash Dalmia, Jure Leskovec, Cyril Zakka, Eduardo Pontes Reis, and Pranav Rajpurkar. 2023. Med-flamingo: a multimodal medical few-shot learner. In *Proceedings of the 3rd Machine Learning for Health Symposium*, volume 225 of *Proceedings of Machine Learning Research*, pages 353–367. PMLR.

OpenAI. 2026. Gpt-5.3 instant. https://openai.com/index/gpt-5-3-instant/. Accessed: 2026-04-25.

project-gtqqq. 2022. V3qwe computer vision dataset. https://universe.roboflow.com/project-gtqqq/v3qwe. Open Source Dataset, Roboflow Universe.

Massimo Redaelli, Stefan Lindner, Stefan Erhard, and Romano Giannetti. *CircuiTikZ: Draw Electrical Networks with TikZ*. Comprehensive TeX Archive Network (CTAN). Version 1.8.0, released on 2025-05-25. LaTeX package for drawing circuit diagrams.

Ayush Roy, Saptarshi Pani, Samir Malakar, Erik Cuevas, Marco Pérez-Cisneros, and Ram Sarkar. 2025. Juhccr-v1: A database for hand-drawn electrical and electronics circuit component recognition. *Scientific Reports*, 15(1):38617.

Md. Faiyaz Abdullah Sayeedi, Anas Mohammad Ishfaqul Muktadir Osmani, Taimur Rahman, Jannatul Ferdous Deepti, Raiyan Rahman, and Salekul Islam. 2025. Electrocom61: A multiclass dataset for detection of electronic components. *Data in Brief*, 59:111331.

Yichen Shi, Ze Zhang, Hongyang Wang, Zhuofu Tao, Zhongyi Li, Bingyu Chen, Yaxin Wang, Zhiping Yu, Ting-Jung Lin, and Lei He. 2025. Amsbench: A comprehensive benchmark for evaluating mllm capabilities in ams circuits. *arXiv preprint arXiv:2505.24138*.

Lejla Skelic, Yan Xu, Matthew Cox, Wenjie Lu, Tao Yu, and Ruonan Han. 2025. Circuit: A benchmark for circuit interpretation and reasoning capabilities of llms. *arXiv preprint arXiv:2502.07980*.

Zhuofu Tao, Yichen Shi, Yiru Huo, Rui Ye, Zonghang Li, Li Huang, Chen Wu, Na Bai, Zhiping Yu, Ting-Jung Lin, and Lei He. 2024. Amsnet: Netlist dataset for ams circuits. In *2024 IEEE LLM Aided Design Workshop (LAD)*, pages 1–5. IEEE.

Felix Thoma, Johannes Bayer, Yakun Li, and Andreas Dengel. 2021. A public ground-truth dataset for handwritten circuit diagram images. In *International Conference on Document Analysis and Recognition*, pages 20–27. Springer.

Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Yang Fan, Kai Dang, Mengfei Du, Xuancheng Ren, Rui Men, Dayiheng Liu, Chang Zhou, Jingren Zhou, and Junyang Lin. 2024. Qwen2-vl: Enhancing vision-language model's perception of the world at any resolution. *arXiv preprint arXiv:2409.12191*.

Haohang Xu, Chengjie Liu, Qihang Wang, Wenhao Huang, Yongjian Xu, Weiyu Chen, Anlan Peng, Zhijun Li, Bo Li, Lei Qi, Jun Yang, Yuan Du, and Li Du. 2025. Image2net: Datasets, benchmark and hybrid framework to convert analog circuit diagrams into netlists. *arXiv preprint arXiv:2508.13157*.

Jiabo Ye, Anwen Hu, Haiyang Xu, Qinghao Ye, Ming Yan, Yuhao Dan, Chenlin Zhao, Guohai Xu, Chenliang Li, Junfeng Tian, Qian Qi, Ji Zhang, and Fei Huang. 2023. mplug-docowl: Modularized multimodal large language model for document understanding. *arXiv preprint arXiv:2307.02499*.

Chenchen Zhao, Zhengyuan Shi, Xiangyu Wen, Chengjie Liu, Yi Liu, Yunhao Zhou, Yuxiang Zhao, Hefei Feng, Yinan Zhu, Gwok-Waa Wan, Xin Cheng, Weiyu Chen, Yongqi Fu, Chujie Chen, Chenhao Xue, Guangyu Sun, Ying Wang, Yibo Lin, Jun Yang, and 3 others. 2025. Mmcircuiteval: A comprehensive multimodal circuit-focused benchmark for evaluating llms. *arXiv preprint arXiv:2507.19525*.

Jinguo Zhu, Weiyun Wang, Zhe Chen, Zhaoyang Liu, Shenglong Ye, Lixin Gu, Yuchen Duan, Hao Tian, Weijie Su, Jie Shao, Zhangwei Gao, Erfei Cui, Yue Cao, Yangzhou Liu, Weiye Xu, Hao Li, Jiahao Wang, Han Lv, Dengnian Chen, and 28 others. 2025. Internvl3: Exploring advanced training and test-time recipes for open-source multimodal models. *arXiv preprint arXiv:2504.10479*.

A CircuitPile Data Construction Details

This section provides additional details on the data construction of CircuitPile, including data sources, label space unification, multi-granularity visual asset construction, self-constructed pre-training data generation, task-oriented instruction-tuning data construction, data organization and statistics, and data quality control.

A.1 Data Sources and Preprocessing

The data in CircuitPile are mainly derived from three types of sources: public circuit-related datasets, our manually annotated CircuitHub-9k, and self-constructed circuit-domain pre-training data. Public datasets are used to expand the visual distribution of circuit diagrams and electronic components, CircuitHub-9k is used to construct the diagnostic and verifiable CircuitPerception-Bench, and the self-constructed pre-training data are used to supplement large-scale controllable visual-language supervision. Table 5 summarizes the major data sources used in CircuitPile, their annotation formats, and their roles in data construction.

The public data cover a wide range of circuit visual scenarios, including printed circuit schematics, handwritten circuit diagrams, real-world electronic component images, PCB images, and analog-circuit netlist-related data. Printed or analog circuit schematic data mainly include CircuitObjects, CV-Circuit, V3QWE, CI2N-series datasets, and AMSNet (Diana, 2024; EZpark, 2024; project-gtqqq, 2022; Xu et al., 2025; Tao et al., 2024). Handwritten circuit diagrams and handwritten component data mainly include Digitize-HCD, JUHCCR-v1, and CGHD (Ahmed et al., 2025; Roy et al., 2025; Thoma et al., 2021). PCB or real-world electronic component image data mainly include PCB Component, DsPCBSD+, and ElectroCom61 (Dataset Ninja, 2021; Lv et al., 2024; Sayeedi et al., 2025). These datasets are complementary in image style, annotation granularity, and task format, which helps improve the coverage of the data framework across different circuit visual scenarios.

CircuitHub-9k is a manually annotated dataset constructed for CircuitPerception-Bench. The data are collected from circuit images in textbooks, books, and engineering materials, with manual annotations of component categories, bounding boxes, and spatial information. Unlike public datasets that are mainly used to expand the training corpus, CircuitHub-9k emphasizes annotation consistency, computable relationships, and verifiable answers, and is therefore primarily used to generate benchmark samples for component recognition, spatial understanding, and counting reasoning.

The self-constructed pre-training data are used to supplement controllable supervision signals that are difficult to obtain at scale from public datasets. This part consists of five types of data: component semantics, textual attributes, multi-object composition, spatial organization, and programmatic structural descriptions. The first four types focus on local components, textual attributes, multi-component combinations, and spatial layouts, respectively. The programmatic structural description data further provide additional supervision for learning topological relations and connection structures by automatically generating circuit diagrams and their structural metadata.

Since different data sources vary in image type, annotation format, and category granularity, we convert the original samples into a standardized intermediate representation. For data with bounding boxes, we retain the image path, data source, component category, bounding-box coordinates, and original annotation fields. For data containing auxiliary annotations such as text, connection points, ports, directions, or intersections, we distinguish component annotations from non-component annotations according to task requirements. For data that only provide image-level categories, we use them for local symbol recognition and component semantic learning. After unified parsing, all data can be used for subsequent image cropping, target highlighting, local region construction, description generation, and question-answer sample generation.

A.2 Label Space Unification

One major source of variation in multi-source circuit data lies in inconsistent label spaces. Different datasets may use different names to denote the same type of component, and may also include non-target component annotations such as text, connection points, ports, direction markers, and similar auxiliary elements. To reduce this heterogeneity, we uniformly clean and map all available annotations.

Specifically, we first map semantically equivalent or highly related raw

Data Source	Image Type	Annotation Format	Main Usage
CircuitObjects (Diana, 2024)	Printed circuit diagrams	Component bounding boxes and category labels	Component recognition and visual asset construction
CV-Circuit (EZpark, 2024)	Printed circuit diagrams	Component bounding boxes and category labels	Component recognition and spatial relation modeling
V3QWE (project-gtqqq, 2022)	Circuit diagrams / component images	Component bounding boxes and category labels	Component recognition and visual asset construction
CI2N series (Xu et al., 2025)	Analog circuit schematics	Device boxes, junctions, direction labels, and netlist-related information	Component recognition, structure learning, and netlist-related tasks
AMSNet (Tao et al., 2024)	Analog circuit schematics	Circuit images and SPICE-netlist-related annotations	Structural description and external task adaptation
Digitize-HCD (Ahmed et al., 2025)	Handwritten circuit diagrams	Component boxes, text boxes, text content, and port information	Handwritten component recognition and textual attribute association
JUHCCR-v1 (Roy et al., 2025)	Handwritten component images	Image-level component category labels	Local symbol semantic learning
CGHD (Thoma et al., 2021)	Handwritten circuit diagrams	Component boxes, text, connection points, endpoints, and junctions	Handwritten circuit understanding and component-level visual asset construction
PCB Component (Dataset Ninja, 2021)	PCB images	Real-world component bounding boxes and category labels	Supplementing real-world component visual semantics
DsPCBSD+ (Lv et al., 2024)	PCB images	PCB defect bounding boxes and category labels	Supplementing circuit-related visual data
ElectroCom61 (Sayeedi et al., 2025)	Real-world electronic component images	Component bounding boxes and category labels	Supplementing electronic component visual semantics
CircuitHub-9k	Circuit diagrams from textbooks and books	Manually annotated component categories, bounding boxes, and spatial information	CircuitPerception-Bench construction and evaluation
Self-constructed PT data	Cropped images, attribute images, composite images, and synthetic circuit diagrams	Categories, textual attributes, layouts, and connection relationships	Circuit-domain pre-training

Table 5: Overview of the data sources used in CircuitPile. Public datasets are used to expand the circuit visual distribution, CircuitHub-9k is used to construct the benchmark, and self-constructed PT data are used to supplement controllable large-scale circuit visual-language supervision.

labels to unified categories. For example, `resistor`, `capacitor.unpolarized`, `diode.light_emitting`, `transistor.bjt`, and `operational_amplifier` are normalized to `Resistor`, `Unpolarized_Capacitor`, `LED`, `BJT`, and `Op_Amp`, respectively. We similarly unify the naming of categories such as voltage sources, logic gates, probes, adjustable resistors, variable capacitors, relays, motors, antennas, and optoelectronic devices. We then filter auxiliary annotations that are not treated as target components, including text regions, connection points, endpoints, wire junctions, ports, direction markers, jumper symbols, and unknown labels. Finally, we perform string-level normalization, such as replacing hyphens with underscores, to ensure a consistent format for the final category names.

In addition to global rules, we apply data-source-specific filtering rules to accommodate the annotation definitions of different datasets. For example, `Ground` in CircuitObjects, `Text` and `Ground` in Digitize-HCD, `Port`, `Ground`, and `VDD` in CI2N-DeviceID, as well as `text`, `junction`, `terminal`, `crossover`, `gnd`, and `unknown` in CGHD are filtered when constructing component-centric samples. This process preserves valid circuit annotation information while reducing the interference of non-component objects in recognition, spatial understanding, and counting tasks.

A.3 Multi-Granularity Visual Asset Construction

Based on the unified annotations and label space, we reconstruct the original circuit images into multi-granularity visual assets. Table 6 summarizes these asset types from three perspectives: data source, learning objective, and training stage.

Visual Asset Type	Construction Source	Main Learning Objective	Training Stage
Component-level visual assets	Cropped from valid component bounding boxes, with optional preservation of textual labels such as names, identifiers, values, or expressions.	Learn local symbol semantics, component categories, and the associations between components and textual attributes.	PT / SFT
Target-aware full-diagram visual assets	Constructed by drawing bounding boxes or highlight markers on the original circuit diagram to indicate one or multiple target components.	Support target localization, component grounding, relative-position reasoning, and multi-target comparison.	SFT
Region-level visual assets	Constructed by cropping local windows around target components while retaining multiple neighboring components in the same region.	Learn local co-occurrence patterns, neighborhood relationships, regional counting, and conditional counting.	PT / SFT
Structure-synchronized synthetic visual assets	Programmatically generated circuit diagrams based on Circuitikz, with synchronized metadata such as components, parameters, junctions, and connection relationships.	Provide explicit structural descriptions, topological supervision, and connection-relation learning.	PT

Table 6: Overview of the multi-granularity visual assets constructed in CircuitPile. This table summarizes the types of visual assets used at different training stages, while Table 7 further presents concrete examples of processed visual inputs in the SFT corpus.

Furthermore, during the SFT stage, we instantiate part of these assets into visual input forms with clearer task orientations. In addition to the original full diagrams, Table 7 presents four representative types of processed inputs, including cropped single-component images, single-target highlighted images, multi-target highlighted images, and multi-component regional images. These inputs are not new asset categories, but concrete usage forms of the component-level, target-aware, and region-level assets in Table 6 for instruction-tuning tasks.

The first type is cropped single-component images, which are obtained from component bounding boxes and used to emphasize local symbolic features, mainly serving component recognition and attribute judgment tasks. The second type is single-target highlighted images, which highlight a single target component in the complete circuit diagram and support target reference, position retrieval, and conditional reasoning. The third type is multi-target highlighted images, where multiple target components are simultaneously annotated in the same image to support relative-position judgment, relation understanding, and multi-object comparison. The fourth type is multi-component regional images, which are cropped local regions containing multiple components from the complete circuit diagram and support region-level composition analysis, local counting, and multi-object reasoning. Through these processed visual inputs, the SFT corpus covers different granularities of understanding, ranging from local components to regional and diagram-level comprehension.

In addition to the visual assets reconstructed from real annotated images, we also construct programmatic structural-description images, corresponding to the synthetic visual assets in Table 6, to supplement topological-relation supervision that is difficult to obtain at scale from real public data. These samples are generated automatically using LaTeX, with synchronized structured metadata recording component composition, parameters, junction nodes, and connection relationships. The generation process includes two forms. The first is based on predefined circuit templates, covering typical circuits such as passive networks, amplifiers, filters, oscillators, power supply circuits, power drivers, RF matching networks, and sensor interfaces. The second is based on random topology generation, where junction nodes are sampled on a two-dimensional grid, horizontal or vertical wire connections are established, and components are randomly inserted along the wires. The former provides typical circuit structures, while the latter supplements non-templated connection patterns. The construction details of this part of the data are described in Section A.4.4.

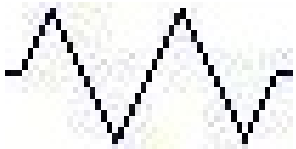
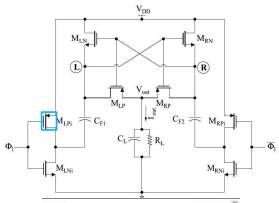
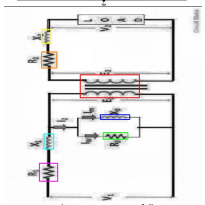
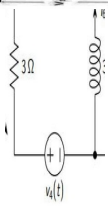
SFT Visual Input Type	Example	Description and Main Usage
Cropped Single-Component Image		Cropped from the bounding box of a single component to emphasize local symbol appearance. This format is mainly used for component recognition and attribute-related question answering.
Single-Target Highlighted Image		The full circuit image with one target component highlighted. This format supports target grounding, location retrieval, and conditional reasoning about a specific component.
Multi-Target Highlighted Image		The full circuit image with multiple target components highlighted simultaneously. This format is mainly used for relative position reasoning, relationship understanding, and multi-object comparison.
Multi-Component Region Image		A cropped local region containing multiple components. This format is used for local composition analysis, regional counting, and multi-object reasoning.

Table 7: Concrete visual input formats used to instantiate the CircuitPile SFT tasks in Table 6, illustrating how different visual forms support tasks such as localization, local perception, comparison, counting, and regional reasoning.

A.4 Self-Constructed Pre-training Data Construction

A.4.1 Construction Objectives and Overall Design

During the pre-training stage, we construct self-generated circuit-domain pre-training corpora from three perspectives: Object-Level Semantic Learning, Multi-Object Organization Learning, and Structure-Level Topology Learning. Unlike directly using raw image–text pairs, our goal is to decompose the fundamental capabilities required for circuit diagram understanding into controllable visual-language supervision signals, enabling the model to acquire local component semantic recognition, textual attribute binding, multi-object composition understanding, and structural topology modeling before entering task-oriented instruction tuning. Specifically, the self-constructed PT corpus consists of five types of data: Component Semantics, Text Attributes, Multi-Object Composition, Multi-Component Attribute, and Structure-Level Topology.

All PT samples are ultimately organized into a unified VQA-style JSONL format. Each sample contains an image path, a question, an answer, a task type, an image type, an answer type, and necessary structured metadata. Through this unified organization, the pre-training stage learns not only general image description capabilities, but also provides controllable supervision for key capabilities in circuit diagram understanding, including local symbol recognition, attribute binding, multi-object composition, position description, and topological structure understanding.

A.4.2 Object-Level Semantic Learning

Object-Level Semantic Learning consists of two types of data, namely Component Semantics and Text Attributes, which are mainly used to learn the local visual semantics of individual components and their textual attributes.

For Component Semantics, we first extract valid component bounding boxes from public datasets and CircuitHub-9k based on the unified component annotations, and crop them into single-component images. Each sample retains metadata such as the cropped image path, unified category

Circuit Type	Example Templates	Complexity
Basic passive circuits	Resistor divider, current divider, RC low-pass/high-pass filters, RLC series/parallel circuits, Pi filters, T filters, integrator/differentiator circuits	1–4
Amplifier circuits	Common-emitter amplifier, inverting/non-inverting proportional amplifier, summing amplifier, differential amplifier, instrumentation amplifier, op-amp integrator, op-amp differentiator, logarithmic amplifier	3–5
Filters and passive networks	LC resonant circuit, second-order RC filter, bridged-T all-pass filter, T/π resistor networks, transformer-coupled network	2–4
Active filters	Sallen–Key low-pass/high-pass filters, multiple-feedback band-pass/low-pass filters, active twin-T notch filter, state-variable filter, Tow–Thomas biquad filter	4–5
Oscillators and waveform generators	RC phase-shift oscillator, Wien-bridge oscillator, crystal oscillator, Schmitt-trigger oscillator, triangular-wave generator, monostable multivibrator	3–4
Power supply and regulation circuits	Three-terminal regulator, LM317 adjustable regulator, voltage multiplier rectifier, Buck/Boost converter, charge pump	3–4
Power drivers and interface circuits	H-bridge motor driver, relay driver, thyristor dimmer, high-side switch, IGBT driver, sensor interface circuit	3–5
RF and matching networks	π-type matching network, L-type matching network, microstrip-line coupler, RF common-emitter amplifier	2–4
Random-topology circuits	Non-templated circuits generated based on grid junctions, random wire connections, and random component insertion	2–5

Table 8: Circuit types covered by the programmatic structural-description data. Predefined templates provide typical circuit structures, while random topology generation further supplements non-templated connection patterns.

label, original image path, original bounding box, cropped bounding box, and data source. We then organize the single-component images into VQA-style image-text samples: the input is the cropped component image, the question is randomly sampled from general image description templates, such as "Please describe this image.", "What do you see in this image?", and "Describe the content of this picture.", and the answer is generated from the semantic description or category name corresponding to the component category. This type of corpus is used to establish the correspondence between the local appearance of circuit symbols and component concepts, enabling the model to recognize resistors, capacitors, diodes, transistors, power sources, logic gates, and other common electronic components across different visual styles.

For Text Attributes, we further construct textual-attribute supervision based on the single-component images. Specifically, we define name prefixes and attribute generation rules for different component categories. For example, resistor components can be assigned prefixes such as `R` or `Resistor`, as well as resistance values in Ω/kΩ/MΩ; capacitor components can be assigned

Pre-training Subset	Input Format	Supervision Content	Scale
Component semantics	Cropped single-component images	Category and functional descriptions	135K
Textual attributes	Component images with names, identifiers, or parameters	Associations between textual attributes and components	135K
Multi-object composition	Multi-component images	Total counts, category distributions, and overall composition descriptions	200K
Spatial organization	Multi-component images with controllable layouts	Object descriptions with positional information	200K
Structural description	Circuit diagrams programmatically generated with Circuitikz	Descriptions of components, parameters, junctions, and connection relationships	68K
Total	–	–	738K

Table 9: Statistics of the circuit-domain pre-training corpus. The structural-description subset consists of programmatically generated circuit diagrams and provides structure-level supervision, including component composition, parameters, and connection relationships.

prefixes such as `C` or `Capacitor`, as well as capacitance values in pF/nF/μF. Voltage sources, current sources, diodes, transistors, logic gates, sensors, displays, and other categories are also assigned corresponding attribute generation rules. For each cropped component, we randomly generate a name, an attribute, or an expression, such as `R3`, `10 k`Ω, or `R3 = 10 k`Ω, and use it as the VQA answer. The input of this type of sample remains the single-component image, while the question is sampled from image description or attribute recognition templates, and the answer is the automatically generated `name`, `attr`, or `expression`. In this way, the model learns not only component categories themselves, but also the associations between components and identifiers, parameters, models, and engineering expressions.

A.4.3 Multi-Object Organization Learning

Multi-Object Organization Learning consists of two types of data, namely Multi-Object Composition and Multi-Component Attribute, which are mainly used to extend learning from the single-component level to multi-object composition and spatialized attribute understanding.

For Multi-Object Composition, we randomly construct grid images of different sizes and fill each non-empty grid cell with a cropped single-component image. Specifically, we randomly sample several component images from the Component Semantics data, resize them proportionally, pad them to a unified size, and place them into grids of different scales, such as 2×2, 2×3, 3×3, and 4×4. To avoid generating overly regular images, we replace some grid cells with blank background images with a certain probability, thereby forming multi-component composite images with varying numbers, categories, and spatial distributions. The corresponding VQA samples take the synthetic grid image as input, with questions randomly sampled from general image description templates. The answers are automatically generated based on the categories and quantities of all non-empty components in the image, for example, describing that the image contains several resistors, capacitors, and diodes. This type of data is mainly used to train the model's capabilities in multi-object recognition, category co-occurrence modeling, and local counting.

For Multi-Component Attribute, we further mix ordinary component images with component images containing textual attributes to construct multi-component attribute corpora with spatial descriptions. Specifically, we generate a fixed 3×3 grid and sample the content of non-empty cells from two types of sources with predefined probabilities: ordinary cropped single-component images and attribute-enhanced samples generated in the Text Attributes stage. For each grid position, we record its row and column coordinates and map them to natural-language position names, such as `top-left`, `top-middle`, `center`, and `bottom-right`. If an ordinary component is placed at a given position, the answer is described as "At the top-left, there is a resistor." If an attribute-enhanced sample is placed at that position, the answer is described as "At the center, there is a component labeled 'R3 = 10 kΩ'." If the position is empty, it is described as a blank background. The final answer is formed by concatenating these descriptions following the order of the nine grid cells. This type of data enables the model to jointly learn the correspondence among component categories, textual attributes, and spatial positions, providing foundational capabilities for subsequent position reasoning, attribute recognition, and multi-object question answering.

A.4.4 Structure-Level Topology Learning

While the first four types of corpora mainly focus on local component semantics, textual attributes, and multi-object spatial organization, Structure-Level Topology Learning further targets circuit connectivity and structural topology modeling. For this type of data, we use the Circuitikz tool (Redaelli et al.) for rendering, including two generation forms: predefined circuit templates and random topology generation.

For predefined circuit templates, we design common circuit structure templates such as passive networks, amplifiers, and filters, and randomly modify the component names and attribute values for each template during generation. Table 8 summarizes the major circuit types covered by the programmatic structural-description data. Each programmatically generated sample contains both a rendered image and structured metadata. The metadata record component names, types, positional endpoints, associated junctions, parameters, direct connection relationships, and circuit descriptions. Since the image and structural information are synchronously produced by the same generation process, this type of data can provide stable supervision without additional manual an-

Algorithm 1 Random Circuit Generation

Require: Grid size g, spacing s, component length l; $J_{\min}, J_{\max}$; max comps per wire $C_{\max}$; safe gap g_s; symbol set $\mathcal{S}$.

Ensure: `circuitikz` code, components, direct junctions.

Generate $n_j \sim \text{Uniform}(J_{\min}, J_{\max})$ distinct grid points as junctions J_i.
for each J_a **do**
  Find collinear J_b with no intervening junction; randomly select 12 as neighbors.
  Add raw wire (J_a, J_b) if selected.
end for
for each raw wire (A, B) **do**
  $c \leftarrow \text{Uniform}(0, C_{\max})$.
  if $c = 0$ **then**
    Add (A, B) to final wires.
  else
    Mark (A, B) as blocked.
    Place c components at equal spacing, each of length l, random symbol from $\mathcal{S}$.
    Split wire into segments around components; add to final wires and components list.
  end if
end for
for each junction J **do**
  Count incident directions (up/down/left/right) from raw wires and components.
  $draw[J] \leftarrow$ True if total incident edges $\neq 2$ or the two directions are opposite; else $draw[J] \leftarrow$ False (bend/corner).
end for
Generate `circuitikz` output: draw junctions (if $draw[J]$), draw final wires as thick lines, draw components using mapped styles.
return TikZ code, component list, unblocked junction pairs, all junctions.

notation, and is mainly used for structure-level semantic learning during pre-training.

For random topology generation, we draw components on randomly generated grids and establish wire connections between components. Specifically, we first randomly generate several junction nodes and then establish candidate connections between collinear nodes. For each connection, we randomly decide whether to insert a component, and split the original connection into multiple wire segments according to the occupied position of the component. We then determine whether to explicitly render each node based on the incident directions of the junctions, and finally output the `circuitikz` code, component list, junction information, and direct connection relationships. The detailed generation procedure is shown in Algorithm 1.

Through the above five types of data, the self-constructed PT corpus progressively expands from individual components, textual attributes, multi-object composition, and spatial organization to structural topology relations. Object-Level data provide supervision for local symbols and textual attributes, Multi-Object data provide supervision for multi-component composition and spatialized descriptions, and Structure-Level data supplement explicit connection relationships and topological structure supervision. Together, these three groups of data form a circuit-domain pre-training corpus that progresses from local to global understanding and from appearance-level perception to structure-level modeling.

A.5 Task-Oriented Instruction-Tuning Data Construction

Based on the pre-training corpus, we further construct task-oriented instruction-tuning data by converting circuit images and their structured annotations into verifiable VQA samples. Unlike the PT stage, which mainly focuses on learning fundamental visual-language representations, the SFT stage places greater emphasis on task execution capabilities, requiring the model to answer specific questions based on the input image, including component recognition, quantitative counting, spa-

tial localization, relative-position judgment, and multi-object ranking. To this end, we automatically generate questions, answers, options, and structured metadata based on the unified annotations, cropped images, highlighted images, and local regional images, and save all samples in a unified JSONL format.

A.5.1 Visual Input Construction

The SFT corpus uses multiple visual input forms to cover circuit diagram understanding requirements at different granularities. First, the original full circuit diagrams are used to train the model for global image understanding, overall counting, and full-diagram spatial reasoning. Second, cropped single-component images are used for component recognition, enabling the model to focus on local symbolic appearances. Third, single-target highlighted images emphasize a specified component in the original diagram and are used for target grounding and target-category recognition. Fourth, multi-target highlighted images annotate multiple valid components simultaneously and are used for relative-position comparison, multi-object counting, and spatial relation understanding. Finally, local multi-component regional images are cropped around several neighboring components and are used for local composition analysis, region-level counting, and multi-object reasoning. Through these visual input forms, the SFT corpus covers circuit diagram understanding capabilities at the local, regional, and global levels.

A.5.2 Component Recognition Question Answering

Component recognition is a fundamental task in the SFT corpus, designed to train the model to identify a specified component in both locally cropped images and complete circuit contexts. Different from Text Attributes in the PT stage, the component recognition task in the SFT stage does not additionally generate names, identifiers, parameters, or expressions. Instead, it directly constructs verifiable recognition QA samples based on the unified component category labels. According to the visual input format, we further divide this part into cropped-component recognition and highlighted-target recognition.

For cropped-component recognition, we use cropped single-component images as input. Each cropped image is automatically generated from a unified component bounding box and retains metadata such as the original image path, original bounding box, cropped bounding box, data source, and standardized category label. Given a cropped image, the supervised answer is the standardized component category. We randomly sample questions from multiple open-ended templates, such as "What is this component?", "Please identify the component in the image.", "What kind of electronic component is shown?", and "Which component is displayed in the image?". The answer directly uses the standardized category name, such as `Resistor`, `Capacitor`, `LED`, or `Op_Amp`. This type of sample is mainly used to train the model to establish the mapping between local visual symbols and standardized component categories.

For highlighted-target recognition, we use single-target highlighted images as input. Unlike cropped images, single-target highlighted images preserve the complete circuit diagram context and highlight one target component in the original image with a bounding box. This design requires the model not only to recognize the target symbol itself, but also to focus on the specified object within a complex circuit background. Question templates include "What is the highlighted component?", "Please identify the component marked in the image.", "What type of component is indicated by the bounding box?", and "Which component is highlighted in the circuit diagram?". The answer is generated from the standardized category label of the highlighted target. Compared with cropped-component recognition, this type of task places greater emphasis on target grounding, enabling the model to locate and identify the specified component according to visual prompts in the full-diagram context.

In terms of sample organization, each component recognition VQA record is stored in a unified JSONL format. In addition to general fields such as `qa_id`, `image_path`, `question`, `answer`, and `answer_type`, we also retain structured fields including `component_label`, `normalized_label`, `bbox`, `crop_bbox`, `image_type`, and `data_source`. Among them, `image_type` is used to distinguish cropped single-component images from single-target highlighted images, `normalized_label` records the standard answer under the unified label space, and `bbox` and `crop_bbox` are used to track the target location in the original and cropped images. Through this design, Component Recognition VQA covers both local cropped recognition

and full-diagram target grounding, providing component-level perceptual foundations for subsequent counting reasoning, spatial reasoning, and connection-relation understanding.

A.5.3 Counting Reasoning Question Answering

Counting reasoning is one of the core tasks in the SFT corpus. Rather than simply asking for the number of target objects in an image, we further decompose counting tasks into three forms: total counting, category-wise counting, and zero-count reasoning, so as to systematically train the model's capabilities in global counting, category-conditioned counting, and target-absence judgment.

For each input image, we extract all valid component objects from the unified annotations and filter out non-target elements such as text boxes, connection points, ports, direction markers, and junctions. For each retained component, we read its standardized category label and bounding-box information, and construct image-level category frequency statistics:

$$\mathcal{C}(I) = \{(y_k, n_k)\}_{k=1}^{K},$$

where y_k denotes the k-th component category appearing in image I, and n_k denotes the occurrence count of that category. Based on these statistics, we further obtain the total number of valid components in the image:

$$N(I) = \sum_{k=1}^{K} n_k.$$

This statistical process provides verifiable answers for subsequent category counting, total counting, and multiple-choice question construction.

For category-wise counting, we generate category-conditioned counting questions for each component category that actually appears in the image. Given a target category y_k, the question asks the model to answer the number of occurrences n_k of that category in the image. To improve the diversity of question expressions, we design two groups of templates. The first group consists of direct question-style templates, such as "How many resistors are shown in the circuit diagram?" and "What is the number of capacitors in the image?" The second group consists of task-oriented instruction templates, such as "Please count all resistors in the image." and "Identify and count all capacitors in the diagram." These two types of templates express the same counting objective, but the former is closer to natural question answering, while the latter is closer to instruction following, thereby improving the model's robustness to different query styles.

For total counting, we do not specify a particular category, but instead require the model to count the total number of all valid components in the image, denoted as $N(I)$. Templates for this type of question include "How many components are in the image?", "What is the total number of components in the diagram?", and "Please count all components in the circuit diagram." Compared with category-wise counting, total counting places greater emphasis on the model's global perception of all targets in the entire image, preventing the model from focusing only on a local subset of objects from a specific category.

In addition, we explicitly construct zero-count reasoning samples to train the model to determine whether a certain target category is absent. Specifically, for each image, we first obtain the set of categories appearing in the current image, denoted as $\mathcal{Y}_{pos}(I)$, and then sample several categories that do not appear in the image from the global category set $\mathcal{Y}_{all}$:

$$\mathcal{Y}_{zero}(I) \subseteq \mathcal{Y}_{all} \setminus \mathcal{Y}_{pos}(I).$$

For each $y_z \in \mathcal{Y}_{zero}(I)$, we generate a question in the same form as category-wise counting, but with the answer fixed to 0. For example, when no inductor appears in the image, we can generate "How many inductors are present in the image?", with the answer being 0. This design explicitly models target-absence cases and reduces the tendency of models to produce non-zero answers based on category priors or language biases in counting tasks.

Each counting question can be formulated as either open-ended question answering or a multiple-choice question. For open-ended question answering, the answer is directly stored as an integer, such as `0`, `2`, or `5`. For multiple-choice questions, we generate several numerical distractors centered around the correct count value and combine them with the correct answer to form a fixed number of candidate options. During option generation, candidate values are required to be non-negative and non-duplicated, and their order is randomly shuffled before being written into the sample. Finally, each sample stores the option letters, option

texts, correct option, and original numerical answer, making it applicable to both generative training and standardized multiple-choice evaluation.

In terms of sample organization, each counting VQA record retains counting-task-related structured metadata in addition to the general fields, including the target category, standardized label, ground-truth count, question type, whether it is a zero-count sample, candidate options, and answer type. For category-wise counting, the target category field records the component category being queried; for total counting, the target category field can be set to `all components`; for zero-count samples, an additional flag `is_zero_count=True` is recorded. These fields allow us to separately analyze model performance on positive category counting, target-absence judgment, and total counting in subsequent experiments, and support more fine-grained error analysis.

A.5.4 Spatial Reasoning Question Answering

Spatial reasoning tasks are designed to train the model to understand relative positional relationships among multiple components in circuit diagrams. Unlike component recognition, which focuses only on the category of a single target, Spatial Reasoning VQA requires the model to simultaneously perceive the positions of multiple valid components and perform comparison, selection, or ordering according to the direction or ranking rule specified by the question. We automatically construct spatial reasoning samples based on the unified component bounding boxes, so that the answers to such questions can be directly computed from structured annotations.

For each input image, we extract all valid component objects and filter out non-target annotations such as text boxes, connection points, ports, direction markers, and junctions. For each valid object o_i, we read its bounding box $b_i = (x_i^1, y_i^1, x_i^2, y_i^2)$ and compute its center coordinate:

$$c_i = \left(\frac{x_i^1 + x_i^2}{2}, \frac{y_i^1 + y_i^2}{2} \right).$$

We then sort the object set according to the horizontal and vertical coordinates of the center points. Sorting by the horizontal coordinate is used to generate left-to-right and right-to-left questions, while sorting by the vertical coordinate is used to generate top-to-bottom and bottom-to-top questions. Based on these sorted results, we construct three types of spatial reasoning questions: extreme-position, second-position, and ordering.

The first type is extreme-position questions, which ask for the target object at an extreme position along a certain direction. Specifically, we identify the leftmost, rightmost, topmost, and bottommost components according to the minimum x coordinate, maximum x coordinate, minimum y coordinate, and maximum y coordinate, respectively. The corresponding question templates include "Which component is the leftmost in the image?", "What is the rightmost component in the image?", "Please identify the component located highest in the image.", and "Which component is located at the bottommost position?". The answer is the component category or target reference at the corresponding extreme position. This type of question mainly trains the model's absolute-position judgment ability.

The second type is second-position questions, which ask for the second target object counted from a certain direction. Based on the sorted object sequence, we select the component that is second from the left, second from the right, second from the top, or second from the bottom as the answer. The question templates include "Which component is second from the left in the image?", "What is the second-rightmost component in the image?", "Please identify the component that is second from the top.", and "Which component is second from the bottom?". Compared with extreme-position questions, this type of question requires the model not only to identify the boundary object, but also to understand the relative order among objects, thereby better training fine-grained spatial ranking ability.

The third type is ordering questions, which require the model to output the overall spatial order of multiple components. For images containing an appropriate number of distinguishable targets, we sort the object set along the horizontal or vertical direction and generate ordered answers in left-to-right or top-to-bottom order. The question templates include "Please list the components from left to right.", "Order the components from left to right in the image.", "Please list the components from top to bottom.", and "What is the top-to-bottom order of the components?". The answer is saved as an ordered component sequence, such as `Resistor, Capacitor, LED`. This type of question trains the model to extend from local positional judgment to global multi-object order-

ing.

To reduce visual ambiguity in spatial questions, we introduce a distance-filtering mechanism during sample generation. For extreme-position and second-position questions, if adjacent candidate targets in the sorted sequence are too close along the corresponding coordinate axis, the question is skipped. Specifically, we use both relative and absolute pixel thresholds to control the minimum required distance: when the axial distance between two candidate targets is below a certain proportion of the image width or height, or below a predefined minimum pixel distance, the corresponding spatial question is not generated. This strategy avoids unstable answers caused by object overlap, nearby bounding boxes, or visually indistinguishable positions. For ordering questions, we also restrict the number of sortable targets and generate samples only when the number of targets falls within an appropriate range, so as to avoid overly long answer sequences or excessively difficult ordering cases.

In terms of answer format, Spatial Reasoning VQA supports both open-ended question answering and multiple-choice questions. For extreme-position and second-position questions, the open-ended answer directly records the target category or target reference. For multiple-choice questions, the correct target is used as the answer, while distractors are sampled from other valid components in the same image, followed by random shuffling of the option order. For ordering questions, the answer is saved as an ordered list and is typically not converted into a multiple-choice format, so as to preserve complete spatial-order supervision. In addition to general fields, each sample also records structured fields such as `position_task`, `axis`, `direction`, `target_object`, `candidate_objects`, and `answer_type`, supporting subsequent task-level statistics and error analysis.

Through the above design, Spatial Reasoning VQA covers different levels of spatial understanding, ranging from single-target extreme-position judgment and relative-order recognition to multi-object ordering. This type of corpus enables the model to reason over object positions and layout structures in circuit diagrams, rather than relying solely on the local visual appearance of individual components, thereby providing training signals for layout understanding, target-relation judgment, and region-level reasoning in complex circuit diagrams.

A.5.5 Sample Organization and Answer Normalization

To support standardized evaluation and stable training, we further convert part of the open-ended QA samples into multiple-choice questions. For counting tasks, the options consist of the correct numerical value and several nearby or randomly sampled numerical distractors. For spatial localization tasks, the options consist of the correct target reference and other candidate component references. All options are randomly shuffled before being written into the samples, and the option letters, correct option, original answer text, and answer type are saved simultaneously. For open-ended questions, the answer type is marked as integer, string, or string list; for multiple-choice questions, the answer type is marked as multiple choice. This design allows the same task to be used for both generative training and automated evaluation.

Finally, each SFT sample is organized into a unified JSONL format. The main fields include `qa_id`, `task_type`, `image_type`, `image_path`, `question`, `answer`, `answer_type`, and `data_source`. For counting tasks, we additionally save `count_target`, `num_objects`, and `is_zero_count`; for spatial reasoning tasks, we additionally save `position_task`, `position_mode`, and candidate target information; for multiple-choice samples, we additionally save `options` and `answer_text`. These structured fields enable the SFT corpus to support not only model training, but also task-level statistics, error analysis, and automated metric computation in subsequent experiments.

Overall, the SFT corpus converts unified annotations and multi-granularity visual assets into circuit diagram VQA samples with clear task definitions, verifiable answers, and standardized formats. This construction process covers a wide range of capabilities, from local component recognition to global counting reasoning and from single-target localization to multi-object spatial ordering, enabling the model to further learn task-oriented response behaviors based on the circuit visual-language representations acquired during pre-training.

A.6 Data Organization and Statistics

After data preprocessing, label space unification, and visual asset construction, we organize the resulting samples into a circuit-domain pre-training corpus and a task-oriented instruction-tuning corpus.

The circuit-domain pre-training corpus is designed for learning fundamental circuit visual-language representations and consists of five subsets: component semantics, textual attributes, multi-object composition, spatial organization, and structural description. Component semantics samples are used to learn the correspondence between circuit symbols and component concepts; textual attribute samples are used to model the binding relationships between component entities and names, identifiers, parameters, or expressions; multi-object composition and spatial organization samples are used to learn object distributions, category combinations, and spatial layouts; and structural description samples are composed of programmatically generated circuit diagrams, providing structural supervision such as component lists, parameters, junctions, and connection relationships. Table 9 reports the statistics of the pre-training corpus.

The task-oriented instruction-tuning corpus is used to transform the aforementioned visual representations into circuit diagram visual question answering capabilities. Based on unified annotations and programmatic task templates, we first construct approximately 31.9M raw SFT question-answering samples, covering seven types of tasks: component recognition, position perception, spatial relation judgment between components, quantitative reasoning, description generation, netlist-related tasks, and connection-relation question answering. Since the sample scales of different tasks vary substantially, we adopt a task-aware balanced sampling strategy and sample 100K balanced instruction-tuning instances from the original task pool for training in the main experiments. Table 10 reports the statistics of each type of SFT corpus.

Overall, by integrating public data, manually annotated benchmark data, and self-constructed pre-training data, CircuitPile forms a multi-level data framework that covers local symbol semantics, textual attribute association, multi-object organization, spatial layout, and structural relationships. This framework provides data support for the transition from fundamental circuit visual-language representation learning to task-level circuit diagram understanding.

A.7 Data Quality Control

A.7.1 Component Bounding Box

Object detection serves as the foundation of our dataset construction, as multiple downstream tasks rely on the quality of detection results. However, open-source datasets are collected from diverse sources and often follow inconsistent annotation standards. To address this issue, we start from our self-constructed CircuitHub-9k dataset, for which all 9K samples are fully manually annotated and cross-checked by multiple annotators. The annotation types include electronic components, junctions, terminals, and text.

Based on this dataset, we train a YOLOv8-l object detection model (Jocher et al., 2023) and use it to perform object detection inference on all open-source datasets. During inference, we set `Confidence` to `0.5` and `IoU_threshold` to `0.6`. We then compare the inference results with the original annotations provided by the open-source datasets and perform matching under the criterion of `IoU` > 0.6. For samples with missing components, excessive components, or label types that cannot be mapped to the unified label space, we conduct further manual annotation.

A.7.2 Component Related Text Content

During the construction of component images with textual attributes and the Netlist instruction-tuning dataset, textual information in circuit diagrams is required. In the object detection model described above, we annotate text regions as a separate category. Using this object detection model, we perform text-region inference on CircuitHub-9k and the open-source datasets, and manually inspect the annotations of text boxes.

For all extracted text boxes, we use PaddleOCR (Cui et al., 2025) for recognition and further conduct manual verification. Since text in circuit diagrams often contains a large number of formula symbols, we represent textual content in LaTeX format. For different forms of LaTeX expressions, we normalize them as much as possible through rule-based replacement, for example, replacing 5Ω with `5\Omega`.

Instruction Data Type	Task Content	Scale
Component recognition	Open-ended and multiple-choice component recognition based on cropped or highlighted images	396K
Position perception	Judging the absolute position of a target component in the diagram, or retrieving the corresponding component according to a specified position	7.043M
Spatial relation judgment between components	Judging relative directions, distances, and spatial proximity relationships between two or more components	3.482M
Quantitative reasoning	Covering single-category counting, regional counting, conditional counting, quantitative comparison, extremum judgment, and global counting tasks	20.607M
Description generation	Whole-diagram description and target-component description	45K
Netlist-related tasks	Tasks related to netlist understanding and structured circuit representation	9K
Connection-relation question answering	Question answering about direct connections and structural relationships	356K
Raw instruction task pool	The collection of raw SFT question-answering samples automatically constructed based on the above task templates	31.939M
Balanced instruction-tuning corpus	A subset obtained from the raw instruction task pool through task-aware balanced sampling	100K

Table 10: Statistics of the task-oriented instruction-tuning corpus.

A.7.3 LLM-Based Answers

In the SFT data, the answers to many specific tasks depend on LLM-generated outputs, such as captioning tasks and semantic understanding tasks. We use three closed-source LLMs, namely Gemini-3.1-Pro, GPT-5.3-Instant, and Claude Sonnet-4.6, to generate the corresponding answers. For each generated answer, we further use the other two LLMs to judge whether the result is correct. Only answers that pass the verification of the other two LLMs are retained (Google, 2026; OpenAI, 2026; Anthropic, 2026).

B CircuitPerception-Bench Construction Details

This section provides additional details on the task design, composition, and evaluation metrics of CircuitPerception-Bench. The main text primarily introduces the overall objective, capability dimensions, and experimental results of the benchmark, while this section further explains how different fine-grained tasks are defined and how these tasks are organized into automatically evaluable visual question answering samples.

B.1 Task Taxonomy

CircuitPerception-Bench is designed to evaluate fundamental visual-language capabilities for circuit diagram understanding, with a focus on three capability dimensions: component recognition, spatial understanding, and counting reasoning. Instead of directly treating complex circuit functional analysis as the primary evaluation objective, we focus on more fundamental, verifiable, and diagnostic capability components in circuit diagram understanding. This design helps more clearly determine whether a model has truly acquired local symbol semantics, target positional relationships, and multi-object quantitative aggregation capabilities in circuit diagrams. To ensure comparability across tasks and stability in automated scoring, all samples in the comprehensive candidate task pool are constructed in the form of multiple-choice questions (MCQs), thereby avoiding the influence of answer expression variations and normalization errors in open-ended responses.

Specifically, the component recognition task evaluates whether a model can identify component categories based on local visual evidence, serving as the foundation for subsequent spatial understanding and counting reasoning. The spatial understanding task further examines the model's

Capability Dimension	Fine-Grained Task	Main Input Format
Component recognition	Component Recognition	Cropped single-component images and single-target highlighted images
Spatial understanding	Absolute Location	Single-target highlighted images
Spatial understanding	Find-by-Location	Multi-target highlighted images and complete circuit diagrams
Spatial understanding	Relative Position	Single-target or dual-target contextual images
Spatial understanding	Nearest/Farthest	Multi-target highlighted images and local regional images
Counting reasoning	Component Counting	Complete circuit diagrams and multi-target highlighted images
Counting reasoning	Region Counting	Complete circuit diagrams and local regional images
Counting reasoning	Conditional Counting	Complete circuit diagrams and local regional images
Counting reasoning	Count Comparison	Complete circuit diagrams and multi-target highlighted images
Counting reasoning	Count Extremum	Complete circuit diagrams and multi-target highlighted images
Counting reasoning	Region Count Comparison	Complete circuit diagrams and regional images
Counting reasoning	Total Counting	Complete circuit diagrams and multi-target highlighted images

Table 11: Task taxonomy of CircuitPerception-Bench. Different tasks correspond to fundamental circuit diagram understanding capabilities, including local symbol recognition, spatial grounding, relational reasoning, and quantitative aggregation.

ability to model target locations, relative directions, and neighborhood relationships, requiring the model to align language descriptions with specific targets or regions in the image. The counting reasoning task requires the model to perform component counting, region counting, conditional counting, and quantitative comparison in multi-component scenarios, emphasizing the joint modeling of recognition, spatial constraints, and quantitative aggregation. Table 11 summarizes the task taxonomy and input formats of the comprehensive candidate task pool. The final CircuitPerception-Bench is obtained by selecting samples from this candidate pool through capability-balanced sampling.

B.2 Task Examples

Table 12 presents representative question examples for different fine-grained tasks in the comprehensive candidate task pool.

These question examples reflect the diagnostic nature of the comprehensive candidate task pool. Component recognition questions mainly rely on local visual evidence of circuit symbols; spatial understanding questions require the model to align language descriptions with target objects, spatial regions, or relative geometric relationships in the image; and counting reasoning questions further require the model to perform counting, comparison, or conditional aggregation based on recognized objects. Since all these questions are automatically generated from structured annotations, their ground-truth answers can be deterministically computed from category labels, bounding-box coordinates, and region partition rules.

B.3 CircuitPerception-Bench Construction

We set sampling quotas for the three capability dimensions, namely component recognition, spatial understanding, and counting reasoning. Within each capability dimension, we adopt round-robin balanced sampling to alternately select samples from different fine-grained tasks and visual input formats, thereby preventing a few high-frequency subtasks from dominating the final evaluation results. In other words, CircuitPerception-Bench is MCQ-only in terms of answer format, while still preserving comprehensive coverage across multiple capabilities and subtasks. The resulting CircuitPerception-Bench contains 2,700 samples and serves as the standard evaluation set for all major model comparison experiments in the main

Capability Dimension	Task Type	Example Question
Component recognition	Component Recognition	What type of component is highlighted in the image?
Spatial understanding	Absolute Location	Where is the highlighted component located in the circuit diagram?
Spatial understanding	Find-by-Location	Which component is located in the upper-left region of the image?
Spatial understanding	Relative Position	What is the relative position of R1 with respect to C1?
Spatial understanding	Nearest/Farthest	Which component is closest to the highlighted component?
Counting reasoning	Component Counting	How many resistors are there in the circuit diagram?
Counting reasoning	Region Counting	How many components are located in the right half of the image?
Counting reasoning	Conditional Counting	How many capacitors are located above the highlighted resistor?
Counting reasoning	Count Comparison	Are there more resistors than capacitors in the image?
Counting reasoning	Count Extremum	Which component type appears most frequently in the circuit diagram?
Counting reasoning	Region Count Comparison	Which region contains more components, the left half or the right half?
Counting reasoning	Total Counting	How many components are there in total?

Table 12: Representative question examples for different task types in the comprehensive candidate task pool. The actual questions are automatically generated from structured annotations and paired with corresponding image inputs and ground-truth answers.

Stage	Operation	Purpose
Capability-dimension quota	Set sampling sizes according to component recognition, spatial understanding, and counting reasoning	Ensure coverage of different fundamental capabilities
Subtask-balanced sampling	Perform round-robin sampling over fine-grained tasks within each capability dimension	Reduce the influence of high-frequency tasks on the overall results
Unified evaluation format	Evaluate all samples using option-level accuracy	Reduce the impact of answer normalization and output-format variations

Table 13: Construction process of CircuitPerception-Bench. This benchmark supports more stable and comparable standardized evaluation across different models.

text. Table 13 summarizes the construction process of CircuitPerception-Bench.

Therefore, the main experiments report four metrics based on this benchmark: Overall, Recognition, Position, and Counting. Among them, Overall denotes the overall accuracy across all samples, while Recognition, Position, and Counting correspond to the three capability dimensions of component recognition, spatial understanding, and counting reasoning, respectively.

C Prompt Templates and Output Normalization

This section provides additional details on the prompt templates, answer extraction rules, and abnormal-output handling procedures used in the evaluation of the comprehensive candidate evaluation resource and CircuitPerception-Bench. Since multimodal large language models may differ in their response formats, a unified prompt design and output normalization pipeline are important for ensuring evaluation stability. We divide prompt design into two levels: task-level question templates used to generate specific question texts, and a unified evaluation wrapper template used to organize the image, question, and candidate options into the model input. During evaluation, we further automatically parse and normalize model responses to reduce the influence of linguistic variations on the final metrics.

Question Type	Prompt Template	Expected Output

Single-choice	`Please answer according to the image, question, and options provided.`
	`Question: {question}`
	`Options: {options}`
	`The final response format should be "**<correct option letter>**".`
		Option letter

Multiple-choice	`Please answer according to the image, question, and options provided.`
	`Question: {question}`
	`Options: {options}`
	`The final response format should be "**<correct option letters>**".`
		Multiple option letters

Table 14: Unified prompt wrapper templates used in the comprehensive candidate evaluation pool. {question} denotes the question text generated by task-level templates, and {options} denotes the candidate options in the multiple-choice question.

C.1 Prompt Templates

Each evaluation sample consists of an image, a question text, and optional candidate options. For CircuitPerception-Bench, all samples are formulated as MCQs. The prompt explicitly requires the model to select the correct answer from the candidate options based on the image and question, and to output the corresponding option letter.

In actual evaluation, `{question}` is replaced with the specific question, and `{options}` is replaced with the option list. To prevent lengthy model explanations from interfering with automatic evaluation, all wrapper templates explicitly specify the required final response format.

In addition to the unified wrapper templates, we design task-level question templates for different fine-grained tasks. The task-level templates directly determine the `question` field in each sample, while the evaluation wrapper template combines the question with the image and candidate options to form the final model input. Table 15 summarizes the question templates, answer generation methods, and supported answer formats for different tasks.

For position-related tasks, questions are generated from the geometric positional relationships of target objects. For example, the extreme-position task determines the leftmost, rightmost, topmost, or bottommost target according to component center coordinates; the second-position task selects the second target after sorting objects along the same coordinate axis; and the ordering task requires the model to list components in left-to-right or top-to-bottom order. To avoid unstable answers caused by components being too close to each other, we set a minimum coordinate-distance threshold when generating such samples and retain only samples with clear spatial distinctions.

For counting-related tasks, questions are generated from the cleaned set of valid components. For example, the total-counting task counts the number of all valid components in the image. Candidate options are constructed around the correct count value, including the correct value and neighboring counts, thereby forming multiple-choice questions with reasonable distractors. For all tasks,

C.2 Answer Extraction and Normalization

Since different models may include explanatory text, Markdown markers, or additional natural-language descriptions in their final responses, we uniformly extract and normalize model outputs before computing the metrics. We extract valid option letters from the model output and convert them into a unified uppercase format. If the output contains explanatory text, such as "The correct answer is B", only the option letter is retained. For multiple-choice questions, we extract all valid option letters, remove duplicates, sort them, and compare the resulting set with the ground-truth answer set.

If the model output is empty, no valid answer can be parsed, or a single-choice question contains multiple options that cannot be uniquely resolved, the sample is treated as an incorrect prediction. If the output format does not fully follow the instruction but contains an explicit answer, we extract the valid answer according to the above rules and continue evaluation. The same answer extraction and normalization pipeline is applied to all models to ensure the comparability of evaluation results.

D Qualitative Analysis

This section further analyzes the capability changes brought by CircuitPile training through

qualitative cases. Unlike the quantitative results in the main text, the qualitative analysis focuses on model behaviors on specific samples, aiming to illustrate whether CircuitPile enables models to perform component recognition, spatial relation judgment, and counting reasoning more reliably in realistic circuit diagram understanding scenarios. We mainly present examples where CircuitPile gives correct answers while other models fail, and further analyze common error patterns of baseline models in circuit diagram understanding tasks, as well as the limitations of the current method.

D.1 Examples Correctly Solved by CircuitPile

Table 16 and Table 17 present representative cases from different tasks. Each table lists the input image, question and candidate options, ground-truth answer, the output of CircuitPile-{VLM}, and incorrect outputs from other models.

D.2 Common Error Patterns of Baseline Models

Based on the above cases, we further summarize common error patterns of baseline models on CircuitPerception-Bench. These errors do not simply arise from output-format issues, but instead reflect limitations in symbol recognition, target grounding, spatial relation modeling, and answer binding.

First, baseline models are prone to symbol confusion. Component symbols in circuit diagrams are usually highly abstract, with subtle differences in local shapes. For example, in the second case, several models incorrectly recognize `Power_Source` as `Switch` or `Capacitor`, indicating that general visual representations do not necessarily capture the key discriminative features of circuit symbols.

Second, baseline models remain unstable in spatial grounding and candidate-option binding. In the first case, the model needs to simultaneously locate the cyan-boxed component, compare its spatial distance to other color-boxed components, and map the nearest target to the correct option. Some models already approach or even identify the correct target in their textual explanations, but still output an incorrect option, suggesting that the alignment among visual targets, language descriptions, and option letters is not stable.

Finally, baseline models are more likely to fail on tasks requiring multi-step compositional judgment. Nearest-component judgment is not a simple classification problem, but involves a combined process of target localization, distance comparison, and answer selection. Similarly, single-component recognition relies not only on local shape, but also on domain-specific symbolic knowledge. These error patterns suggest that circuit diagram understanding is not equivalent to general natural image recognition. Models need to simultaneously acquire symbol-level visual discrimination, target grounding, spatial geometric relation modeling, and stable answer alignment. CircuitPile is designed around these foundational capabilities, and therefore can improve the error behaviors of baseline models across multiple cases.

D.3 Failure Cases and Limitations

Although CircuitPile outperforms non-domain-adapted base models on multiple tasks, the qualitative analysis also reveals several limitations of the current method. First, the model may still make component recognition errors when image quality is low, symbols are drawn in non-standard forms, or local regions have insufficient resolution. Second, in circuit diagrams with dense layouts, the distances between target objects can be small, which may reduce the stability of spatial grounding and relational reasoning. Third, the current benchmark mainly focuses on fundamental capabilities such as component recognition, spatial understanding, and counting reasoning, while its coverage of higher-level topological relation understanding, connection consistency verification, and functional reasoning remains limited.

Therefore, future work can be extended in three directions. First, higher-quality and larger-scale structure-level supervision can be introduced to strengthen the model's understanding of connection relations and topological semantics. Second, more complex open-ended reasoning tasks, such as functional analysis, fault diagnosis, and design intent understanding, can be constructed to further evaluate the model's reasoning capability in the circuit domain. Third, explicit structural parsing modules or netlist-level supervision can be incorporated into the training and inference pipeline, enabling the model not only to recognize objects and spatial relationships in images, but also to form more stable structured circuit representations.

Capability Dimension	Task Type	Example Question Template	Answer Generation Method
Component recognition	Component Recognition	`What type of component is {target}?`	Generate the answer according to the standardized category label of the target component.
Spatial understanding	Position Finding	`Which component is the {position} in the image?`	Determine the leftmost, rightmost, topmost, bottommost, or specified ranked target by sorting component center coordinates.
Spatial understanding	Ordering	`Please list the components from {direction}.`	Generate the target sequence from left to right or from top to bottom based on component center coordinates.
Spatial understanding	Absolute / Relative Location	`Where is {target} located?` `What is the relative position of {target} with respect to {reference}?`	Generate the answer according to the region containing the target bounding-box center, or the coordinate difference between the target and reference component centers.
Spatial understanding	Nearest / Farthest	`Which component is {distance_relation} to {target}?`	Determine the nearest or farthest object based on the center-point distance between the target and other components.
Counting reasoning	Component / Total Counting	`How many {component_type} are there in the image?` `How many components are there in total?`	Count the number of components of the specified category or the total number of components based on the cleaned valid component set.
Counting reasoning	Region / Conditional Counting	`How many components are located in the {region}?` `How many {component_type} are located {spatial_condition} {reference}?`	Count components satisfying the requirements according to region partitioning, category constraints, and spatial conditions.
Counting reasoning	Count Comparison / Extremum	`Are there more {type_A} than {type_B}?` `Which component type appears {most/least} frequently?`	Generate comparison answers or extremum categories based on category-level or region-level counting results.

Table 15: Question templates and answer generation methods for different fine-grained tasks in the comprehensive candidate evaluation pool. {target}, {reference}, {component_type}, {region}, {position}, {direction}, and {spatial_condition} denote fields automatically filled from structured annotations. The system constructs candidate options based on the correct answer and distractors, and performs evaluation using the unified wrapper templates in Table 14.

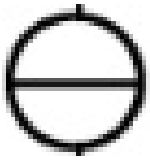
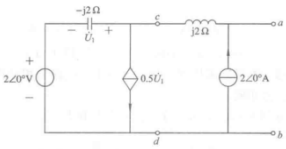
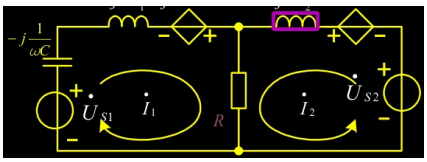
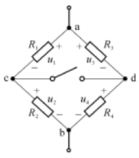
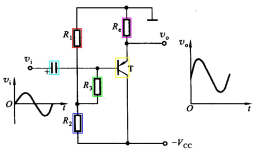
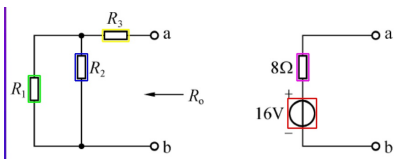
Image	Task	Question / Options	GT	Predictions
	Recognition	**Q:** What is the correct classification of the component shown here? **A.** Switch **B.** Electret_Microphone **C.** Power_Source **D.** Resistor	C	CircuitPile-VLM: C Qwen3-VL-8B: B InternVL3.5-8B: D
	Position Finding	**Q:** What is the component farthest to the right? **A.** the rightmost power_source **B.** the capacitor **C.** the bottommost power_source **D.** the inductor	A	CircuitPile-VLM: A Qwen3-VL-8B: D InternVL3.5-8B: B
	Absolute Location	**Q:** Where is the component inside the purple box located vertically, top or bottom? **A.** top side **B.** bottom side	A	CircuitPile-VLM: A Qwen3-VL-8B: B InternVL3.5-8B: B
	Relative Position	**Q:** Where is the switch relative to the leftmost resistor? **A.** above **B.** below **C.** left **D.** right	D	CircuitPile-VLM: D Qwen3-VL-8B: B InternVL3.5-8B: C
	Nearest	**Q:** What is the nearest highlighted component to the blue-boxed component? **A.** the red-boxed component **B.** the green-boxed component **C.** the yellow-boxed component **D.** the cyan-boxed component	B	CircuitPile-VLM: B Qwen3-VL-8B: C InternVL3.5-8B: R3
	Farthest	**Q:** Which boxed component is farthest from the purple-boxed component? **A.** the red-boxed component **B.** the blue-boxed component **C.** the green-boxed component **D.** the yellow-boxed component	C	CircuitPile-VLM: C Qwen3-VL-8B: D InternVL3.5-8B: D

Table 16: CircuitPerception-Bench CircuitPile-VLM

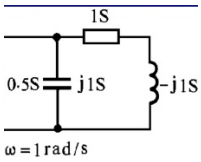
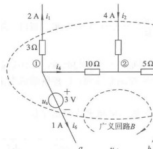
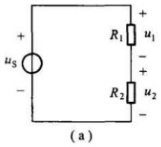
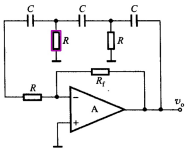
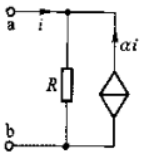
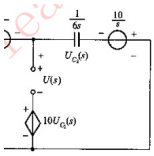
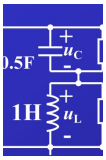
Image	Task	Question / Options	GT	Predictions
	Component Counting	**Q:** Please identify all Capacitors and count them. **A.** 0 **B.** 1 **C.** 3 **D.** 2	B	CircuitPile-VLM: B Qwen3-VL-8B: A InternVL3.5-8B: A
	Total Counting	**Q:** What is the number of all components in the diagram? **A.** 4 **B.** 3 **C.** 6 **D.** 5	D	CircuitPile-VLM: D Qwen3-VL-8B: C InternVL3.5-8B: None
	Region Counting	**Q:** How many highlighted objects are present in the top-right region of the diagram? **A.** 3 **B.** 0 **C.** 1 **D.** 2	C	CircuitPile-VLM: C Qwen3-VL-8B: D InternVL3.5-8B: A
	Component Conditional Counting	**Q:** Count the components below the purple-boxed Resistor. **A.** 3 **B.** 2 **C.** 0 **D.** 7	A	CircuitPile-VLM: A Qwen3-VL-8B: B InternVL3.5-8B: B
	Count Comparison	**Q:** Between Resistor and Power Source, which one has a larger count in the image? **A.** Resistor **B.** Power Source **C.** cannot determine **D.** same	D	CircuitPile-VLM: D Qwen3-VL-8B: A InternVL3.5-8B: A
	Region Count Comparison	**Q:** Which side of the diagram has more components, top or bottom? **A.** bottom half **B.** top half	B	CircuitPile-VLM: B Qwen3-VL-8B: A InternVL3.5-8B: A
	Count Extremum	**Q:** Which component category has the highest count in the diagram? **A.** Resistor **B.** Capacitor **C.** Inductor **D.** none of the above	A	CircuitPile-VLM: A Qwen3-VL-8B: D InternVL3.5-8B: D

Table 17: Additional qualitative cases of different task types in CircuitPerception-Bench. These cases show that CircuitPile-VLM can correctly answer representative questions involving component recognition, spatial understanding, and counting reasoning, while other models produce incorrect outputs.