CURVLM: Circuit Understanding Via Group Relative Policy Optimization (GRPO) on Vision Language Models

Zixuan Zhao
zzhao24@stanford.edu

Adhi Daiv
adhidaiv@stanford.edu

Diego Porras
dpg0622@stanford.edu

Abstract

Vision-Language Models (VLMs) have demonstrated impressive capabilities in text understanding, but their performance on complex structured images remains limited, especially in technical domains such as engineering diagrams. In this work, we investigate the challenge of teaching a VLM to interpret digital circuit diagrams, a task requiring both fine-grained visual recognition and domain-specific reasoning. We introduce a novel dataset of circuit diagrams containing both synthetic and hand-drawn examples, and apply Group Relative Policy Optimization (GRPO) - a reinforcement learning approach - to fine-tune the VLM. We analyze our results on both synthetic and hand drawn images (77.0% synthetic, 62.5% hand drawn, 22.5% base) and discuss potential areas for further improving accuracy.

1. Introduction

1.1. Problem

Large, modern organizations, particularly those in highly technical sectors such as the semiconductor industry, grapple with a significant volume of multimodal information. This information is not limited to just text but includes a rich tapestry of structured visual elements: detailed diagrams, complex flowcharts, and domain-specific images like intricate circuit schematics.An AI agent proficient in understanding these diagrams can significantly streamline various processes. Thus, enhancing AI's capability to decode and utilize multimodal information, specifically circuit diagrams, is crucial for driving efficiency and innovation within the chip and semiconductor industry. However, while current LLMs excel at text comprehension, their image understanding capabilities, particularly for domain specific diagrams or charts, are less developed. [9]

1.2. Potential Impact

The transformative effect of code-proficient Large Language Models (LLMs) on software engineering offers a compelling precedent. These models have revolutionized workflows by assisting in code generation, debugging, and documentation, significantly boosting developer productivity and accelerating innovation cycles. Similarly, an AI agent adept at interpreting intricate circuit schematics and hardware description languages could herald a paradigm shift in the semiconductor industry. Such a system could automate or augment tasks like design verification, schematic comparison, component identification, documentation generation from diagrams, and even assist in debugging hardware designs. This would free up hardware engineers from time-consuming manual analysis, reduce errors, shorten design-to-production timelines, and ultimately foster a more agile and innovative hardware development ecosystem, much like their counterparts have in software.

1.3. Overview of Results

We selected Qwen2.5-VLM-3B-Instruct [1] due to its small size, open source documentation, and compatibility with existing libraries. We experimented with two methods of fine tuning: SFT and GRPO. We constructed a training and evaluation dataset of synthetic digital circuits diagrams. We found we had poor results with SFT, but GRPO achieved an 77.0% accuracy in our evaluations, up from 22.5% accuracy for the base model. In addition, we used 20 hand-drawn diagrams for further evaluation and found GRPO was able to achieve 62.0% compared to the base model's 22.5% accuracy. Our results show that with proper fine-tuning, VLMs can get much better at recognizing circuit elements correctly, although there is still more work to do before we can consider them reliable enough to provide useful analysis.

2. Related Work

The automated understanding of circuit diagrams has evolved from early rule-based systems to sophisticated deep learning models.

2.1. Early and Computer Vision-Based Approaches

Initial efforts relied on traditional Computer Vision (CV) techniques, involving image pre-processing (e.g., binarization, skeletonization), component detection (using pattern

matching, SVMs, or later, object detectors like YOLO [16]), line and junction detection (e.g., Hough Transform [6]), and text recognition (OCR). These multi-stage pipelines aimed to reconstruct a circuit's netlist. Efforts to improve these traditional algorithms have also been explored, demonstrating advancements in recognition speed and accuracy for both printed and hand-drawn circuit diagrams [10]. In addition, a computer vision-based framework has also been developed for power converter identification and analysis [2]. However, these methods were often brittle, sensitive to variations in drawing styles and image quality, and lacked deep semantic understanding beyond spatial proximity. Errors in early stages could propagate, leading to incorrect interpretations. The need for large, high-quality annotated datasets was also a significant bottleneck.

2.2. Large Language Models (LLMs) for Circuit Analysis

Recently, foundation models, including LLMs and VLMs, have transformed circuit analysis. Circuit Foundation Models (CFMs) [7] are pre-trained on extensive circuit data and then fine-tuned for specific Electronic Design Automation (EDA) tasks. CFMs are categorized into encoder-based models (for learning circuit representations for predictive tasks like power estimation) and decoder-based, LLM-centric models (for generative tasks like HDL code generation, optimization, and analog circuit reasoning). Related research also focuses on AI-driven schematic image recognition for RTL generation in integrated circuit design [22]. Moreover, the CIRCUIT [21] benchmark was developed to assess LLM reasoning on analog circuits, often using netlists alongside diagrams. While many CFMs operate on textual representations (like Verilog or netlists), projects focusing on interpreting circuits directly from visual schematics face the added complexity of robust visual grounding, where the Vision Transformer (ViT) component of a VLM plays a crucial role and can be a bottleneck. Research also distinguishes between analog circuit analysis (continuous signals, complex functions) and digital circuit analysis (discrete logic levels, Boolean algebra), each presenting unique challenges for AI.

2.3. Vision-Language Models for Understanding Structured Visual Data

VLMs have advanced in joint image-text understanding, but their proficiency with general imagery doesn't always extend to structured visuals like scientific diagrams, charts, and schematics, which have their own grammar and rely on precise spatial and symbolic conventions.

General Challenges for VLMs: VLMs struggle with tasks requiring precise visual arithmetic (e.g., counting, comparing lengths in charts), parsing fine-grained details in dense diagrams, and recognizing geometric shapes and their relational structures (e.g., connectivity in flowcharts or circuit diagrams). This indicates a "semantic gap" where VLMs may perform surface-level recognition but falter on deeper, structured semantic understanding.

Methodologies for Enhancing VLM Comprehension: Researchers are exploring various methods to improve VLM understanding of structured graphics. These include: Post-training strategies: COGALIGN [17], inspired by cognitive development, trains VLMs on invariant visual properties (length, angle) using synthetic data and Direct Preference Optimization (DPO) [15] to improve foundational visual arithmetic. Decomposition and region-focused analysis: The Chain-of-Regions (CoR) [11] approach uses traditional CV to segment diagrams into smaller elements before VLM processing. Specialized models and benchmarks: ChartVLM and the ChartX [24] benchmark are examples tailored for chart understanding.

2.4. Reinforcement Learning for (Vision-) Language Model Capabilities

Reinforcement Learning for Advanced (Vision-)Language Model Capabilities Reinforcement Learning (RL) is increasingly used to fine-tune LLMs and VLMs, enabling them to learn complex, multi-step reasoning strategies by receiving feedback as rewards. Group Relative Policy Optimization (GRPO) [20]: GRPO has emerged as a promising RL method, particularly for tasks with objectively verifiable outcomes. It works by generating multiple candidate responses for a query and evaluating them relative to each other, computing advantages based on the comparison of each response's reward to the group's average reward. This is well-suited for reasoning-intensive tasks like mathematical problem-solving, code generation, and circuit interpretation. Key advancements using GRPO include: VL-Rethinker [23]: This model aims to cultivate "slow thinking" (explicit reflection) in VLMs. It addresses the "vanishing advantages" problem (where small reward variance across candidate responses leads to weak learning signals) by incorporating Selective Sample Replay (SSR) – replaying high-value past samples. It also uses "Forced Rethinking" (prompting self-reflection) to achieve state-of-the-art results on multimodal math reasoning benchmarks. Curr-ReFT (Curriculum Reinforcement Fine Tuning) [5]: This paradigm addresses out-of-distribution generalization and reasoning issues in smaller VLMs. It uses curriculum-based RL with GRPO and a difficulty-aware reward design, and also identified the "Brick Wall" phenomenon, where small VLMs improve on simple tasks but struggle with complex ones requiring simultaneous visual understanding and reasoning.

3. Data

While multi-level datasets and benchmarks like AICircuit have been developed for AI-driven analog integrated circuit design [13], since our focus is on digital circuits, we created our own synthetic dataset of 1800 training and 200 test {image, question, answer} tuples.

First, we randomly generate a combinational logic string like `"a and (b or (not c))"`. We constrain the expression to have an integer range of 1 to 5 logical operators inclusive (i.e., 1–5 logic gates).

Since we know the ground truth combinational logic expression, we can ask questions about the image such as:

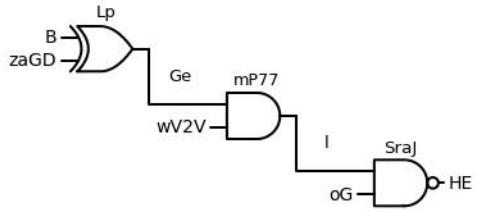

Figure 1. circuit_000287.jpg

Image: `circuit_000287.jpg`
Question Type: `gate_count`
Conversation:

- **Prompt:** `<image> How many logic gates are in this circuit diagram? Your answer should be a single number (e.g., 5).`
- **Answer:** `3`

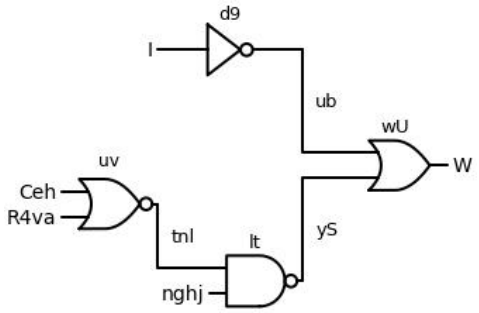

Figure 2. circuit_000840.jpg

Image: `circuit_000840.jpg`
Question Type: `gate_inputs`
Conversation:

- **Prompt:** `<image> What are the inputs of gate 'uv'? List the inputs separated by commas (e.g., 'w1, w2, w3'). Do not include any other words or explanations.`
- **Answer:** `Ceh, R4va`

We selected a total of 8 question types to test the model's ability to understand the circuit diagrams, focusing on tasks such as visual counting, label association, element connectivity, and classification.

- **Gate Count**: Number of gates in the circuit.
- **Gate Input Count**: Number of inputs to a logic gate.
- **Gate Type Count**: Number of gates of a specific type (e.g., AND, OR, XOR).
- **Gate Type**: Type of logic gate (e.g., AND, OR, XOR).
- **Gate Inputs**: Names of input wires to a logic gate.
- **Gate Outputs**: Names of the outputs of a logic gate.
- **Primary Inputs List**: Top-level inputs to the entire circuit diagram (non-intermediate wires).
- **Gate Fan Out**: Number of circuits a single wire drives.

We then use the Schemdraw library's `logicparse()` [18] function to generate a corresponding image given this combinational logic. Certain parameters, like text font size and relative location to each logic gate, were perturbed according to a Gaussian distribution. Schemdraw did not support intermediate wire labeling, so we had to partially rewrite its circuit drawing functionality to achieve this (as well as adding random perturbations to label font size and gate spacing).

3.1. Handdrawn Circuits

We also evaluated our fine-tuned model on 20 handdrawn circuit diagrams, aiming to see how they perform on real-world data.

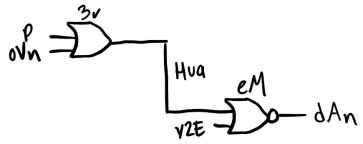

Figure 3. Hand-drawn circuit image 1.

3.2. Limitations of Our Dataset

We did not allow for cycles in our combinational logic, as we found it would complicate the data generation excessively. DAG (Directed Acyclic Graph) circuits were sufficient to challenge the model on visual understanding of circuit images. However, digital circuits with feedback

and cyclic connections are common in the wild, and this is something we hope to add in our future work.

We also constrained both our circuit image size and the number of logical gates in the interest of keeping training time reasonable.

Finally, our hand-drawn circuit images may not accurately describe "real world" distribution of circuit images due to its limited number of samples.

4. Methods

Our approach focuses on efficiently fine-tuning a pre-trained vision-language model for a specialized task. We selected methods prioritizing computational efficiency and performance, leveraging recent advancements in large language model (LLM) training.

4.1. Base Model

We selected Qwen2.5-VL-Max (3B Instruct) as our base model [1]. Qwen2.5-VL is a series of large-scale vision-language models designed to perceive and understand both text and visual information, making it a suitable foundation for tasks requiring multimodal understanding.

4.2. Fine-Tuning with LORA

Given the substantial computational resources typically required to fine-tune LLMs, we employed Low-Rank Adaptation (LoRA) [8]. LoRA significantly reduces the number of trainable parameters, and thus GPU memory requirements, by injecting trainable rank decomposition matrices into the layers of the Transformer architecture while keeping the pre-trained model weights frozen. This allows for more efficient fine-tuning without a substantial loss in performance. For a pre-trained weight matrix $W_0 \in \mathbb{R}^{d\times k}$, its update ΔW is represented by the product of two smaller matrices: $B \in \mathbb{R}^{d\times r}$ and $A \in \mathbb{R}^{r\times k}$, where the rank $r \ll \min(d, k)$. The modified weight matrix W' is expressed as:

$$W' = W_0 + \alpha\frac{BA}{r}$$

where α is a scaling factor [8]. As [8] demonstrated, LoRA can match or exceed the performance of full fine-tuning while being considerably more parameter-efficient.

4.3. Group Relative Policy Optimization (GRPO)

For the core training methodology, we adopted Group Relative Policy Optimization (GRPO), first introduced in DeepSeekMath [20]. GRPO is a reinforcement learning-based approach that has shown strong performance in mathematical reasoning tasks [4]. It focuses on optimizing the policy by comparing the rewards of generated solutions within a group, rather than relying solely on absolute reward signals, which can be noisy or difficult to define precisely. The GRPO advantage estimate $\hat{A}_{GRPO}(x, y)$ for an input x and a sampled output $y \sim \pi_k(\cdot|x)$ (where π_k is the current policy) is formulated as [20]:

$$\hat{A}_{GRPO}(x, y) = \frac{r(x, y) - \hat{\mu}_{\pi_k,r}(x)}{\hat{\sigma}_{\pi_k,r,\epsilon}(x)}$$

where $r(x, y)$ is the reward, $\hat{\mu}_{\pi_k,r}(x)$ is the estimated mean reward for input x under policy π_k:

$$\hat{\mu}_{\pi_k,r}(x) = \frac{1}{G}\sum_{l=1}^{G} r(x, y_l)$$

and $\hat{\sigma}_{\pi_k,r,\epsilon}(x)$ is the estimated standard deviation of rewards for input x under policy π_k, with a small constant ϵ for numerical stability:

$$\hat{\sigma}_{\pi_k,r,\epsilon}(x) = \sqrt{\frac{1}{G}\sum_{l=1}^{G}(r(x, y_l) - \hat{\mu}_{\pi_k,r}(x))^2 + \epsilon}$$

This relative comparison helps stabilize the training process and guides the model towards generating higher-quality outputs.

4.4. DAPO-Inspired Enhancements

To further enhance training efficiency and potentially improve performance, we incorporated several techniques inspired by Decoupled Clip and Dynamic Sampling Policy Optimization (DAPO) [12]. Specifically:

Clip Higher: We adjusted the clipping range for policy updates. DAPO suggests that a less restrictive clip can sometimes speed up convergence by allowing for larger, more confident updates when the advantage is high [12].

No KL Divergence: We opted to remove the Kullback-Leibler (KL) divergence constraint often used in methods like Proximal Policy Optimization (PPO) [19] to keep the fine-tuned policy close to the original policy. The PPO objective with a KL penalty is typically formulated as [19]:

$$L^{KLPEN}(\theta) = \hat{\mathbb{E}}_t\left[r_t(\theta)\hat{A}_t - \beta\mathrm{KL}[\pi_{\theta_{old}}(\cdot|s_t), \pi_\theta(\cdot|s_t)]\right]$$

While KL divergence helps in stabilizing training, DAPO and related works have explored scenarios where removing or adjusting its influence can lead to faster adaptation, especially when a significant shift from the base model's behavior is desired. These modifications aim to accelerate the learning process by allowing the model more freedom to explore the policy space.

4.5. Alternatives Approaches Considered

In the initial phases of our project, we briefly experimented with Supervised Fine-Tuning (SFT) [14, 3]. SFT involves training the model on pairs of (input, desired output)

examples in a standard supervised learning manner. The standard loss function for SFT in language models is the negative log-likelihood (cross-entropy loss) [14]:

$$\mathcal{L}_{SFT}(\theta) = -\sum_{(x,y)\in\mathcal{D}} \sum_{t=1}^{|y|} \log P(y_t|y_{<t}, x; \theta)$$

However, we observed that our models struggled to reliably learn the target task using SFT alone, often producing incoherent or irrelevant outputs. While this could potentially be attributed to an insufficient number of training iterations or suboptimal hyperparameter tuning, the initial results prompted us to explore more advanced reinforcement learning-based fine-tuning strategies like GRPO, which are designed to handle more complex generation and reasoning tasks. This combined approach—a strong base model, efficient LoRA fine-tuning, a robust GRPO training algorithm, and DAPO-inspired speed enhancements—was chosen to balance performance with computational feasibility, drawing upon demonstrated successes in the field.

5. Experiments

We conducted experiments to demonstrate our AI agent's ability to understand digital logic circuit diagrams. Our goal was to confirm the model could accurately interpret visual circuit representations and answer related queries, thereby establishing a strong framework for automated circuit analysis.

Our initial approach involved Supervised Fine-Tuning (SFT) of a pre-trained Vision Language Model (VLM) using synthetically generated circuit image and question-answer pairs. The model's objective was to predict the correct answer given a circuit image and a question.

However, this SFT method showed significant limitations. The model frequently produced inaccurate final answers and, more critically, flawed reasoning. Its explanations were often superficial, relying on pattern matching rather than true logical deduction. This suggested a shallow understanding of circuit semantics and connectivity, highlighting the need for a more sophisticated training approach to achieve deeper logical reasoning and generalization.

In response to the inherent limitations observed with SFT, our research strategically pivoted towards a reinforcement learning-based approach. Specifically, we leveraged a modified Group Relative Policy Optimization (GRPO) algorithm. GRPO, unlike purely supervised methods, allows the agent to learn from its own generated outputs and the subsequent evaluation of those outputs, enabling a more adaptive and exploratory learning process.

5.1. Question Types

One of our experiments involved the expansion of question types. Initially, our dataset encompassed only four fundamental question types, which provided a limited scope for evaluating the agent's comprehensive understanding. To thoroughly probe the agent's capabilities across various facets of circuit analysis and to identify specific areas of strength and weakness, it was systematically expanded to eight question types. These question types were mentioned in the data section earlier.

This expansion served not only to test the model on a larger variety of tasks, but also to provide more comprehensive and granular information about the agent's understanding of each circuit image. The increased diversity in queries allowed for a more holistic and robust evaluation of the agent's reasoning abilities.

5.2. Dynamic Sampling

Modifications to the base GRPO algorithm were implemented to enhance both training efficiency and overall model performance. One key innovation was the introduction of dynamic sampling. This adaptive mechanism leverages an exponential moving average of the model's accuracy on each specific question type. During training, if the model's moving average accuracy on a particular question type consistently exceeds a predefined threshold (i.e., 95%), the probability of sampling that question type for subsequent training batches is dynamically decreased, up to a maximum `skip_probability` of 0.9. This strategic reduction in sampling frequency for well-mastered question types prevents redundant training and allows computational resources to be more efficiently allocated. Conversely, if a subsequent performance check reveals a decline in accuracy for a previously skipped question type, it is promptly re-introduced into the training pool. This adaptive sampling strategy reduced overall training time by directing the learning process towards areas where the model required more attention. Furthermore, by continuously focusing on areas of current weakness, dynamic sampling inherently helps to achieve more uniform performance across all question types, which prevents the model from over-optimizing on easily mastered tasks while neglecting more challenging ones. This led to a more balanced and robust agent. The same is shown in Figure 4.

5.3. Image Resolution

Experiments were also conducted to determine the optimal image resolution for the input circuit diagrams while balancing visual information fidelity with computational efficiency. We tested three distinct resolutions for training: the original 72 DPI (dots per inch), 1.5 times the original (resulting in 108 DPI), and 2 times the original (144 DPI). While the 1.5 times original DPI (108 DPI) resolution consistently yielded the best results in terms of model accuracy, we observed a substantial increase in training time with higher image resolutions due to increased computational

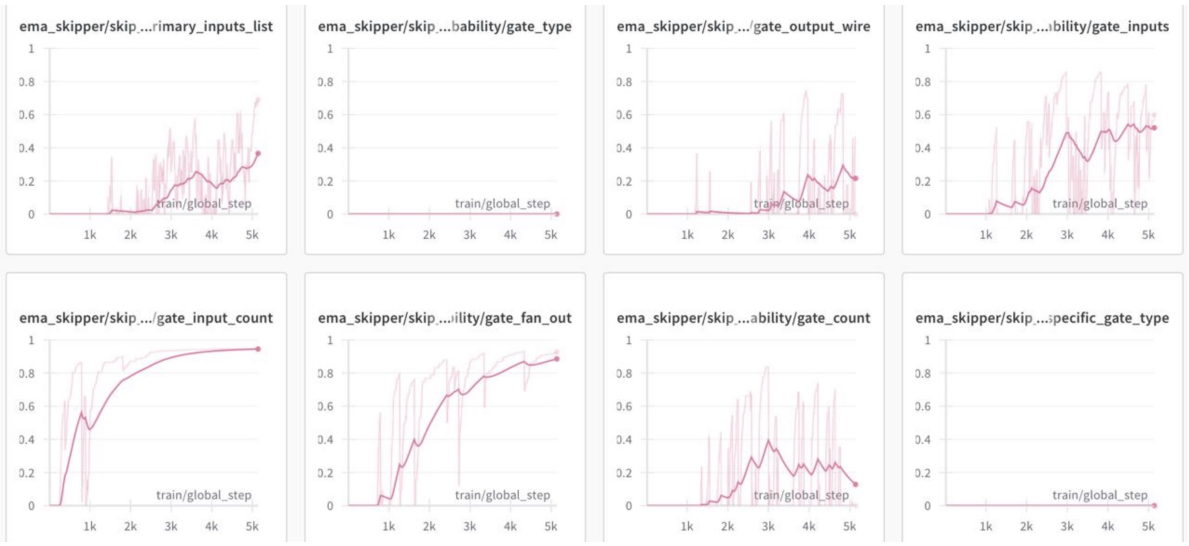

Figure 4. Image representing Dynamic Sampling. Y axis: Probability of skipping a training step. X axis: Steps of training.

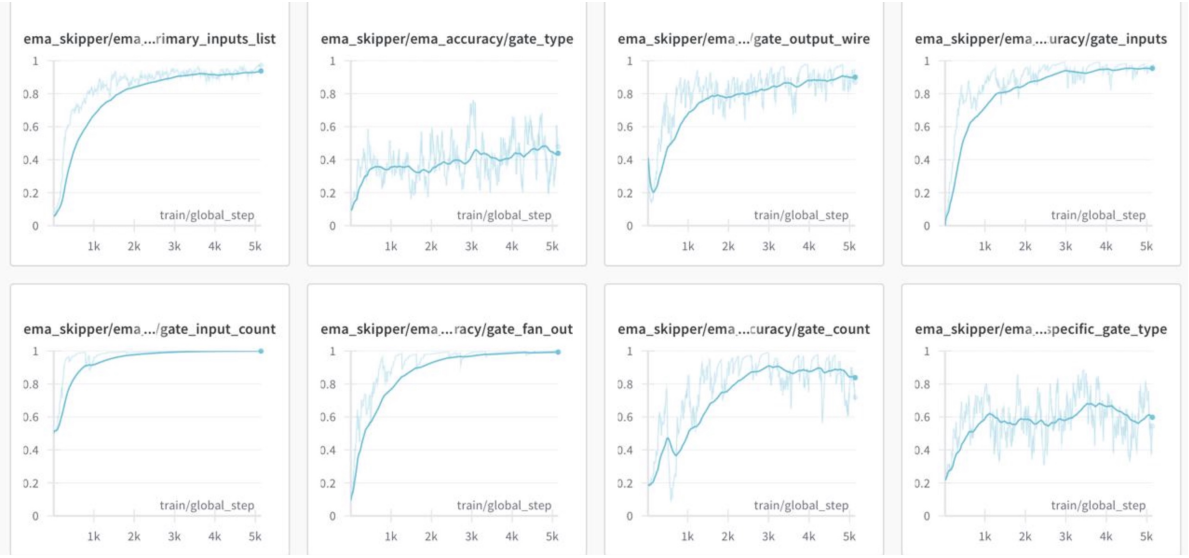

Figure 5. Image highlighting the accuracy of GRPO based on each question type. The hyperparameters are same as those mentioned in section 5.4 with temperature value of 1.2. Y axis: Accuracy. X axis: Steps of training.

load and memory footprint. Therefore, to prioritize training speed and maintain practical feasibility for our project while still achieving acceptable performance, we opted to retain the original 72 DPI resolution for our primary training runs. This decision represented a pragmatic trade-off between maximizing accuracy and managing computational resources effectively.

5.4. Hyperparameter Tuning

Extensive hyperparameter tuning was performed to identify the optimal configurations for the GRPO model - a critical step in maximizing its performance. We systematically tested various values for key hyperparameters, including batch size, number of generations, learning rate, LoRA rank, and temperature. Our findings indicated that the best results were achieved with the following hyperparameter values:

LoRA rank: 80. LoRA (Low-Rank Adaptation) is a parameter-efficient fine-tuning technique that injects trainable low-rank matrices into the transformer layers. The rank determines the dimensionality of these matrices that influences the expressiveness of the adaptation. A rank of 80 was found to provide sufficient capacity for adaptation without introducing excessive parameters.

LoRA alpha: 80. LoRA alpha is a scaling factor for the LoRA weights. Keeping it identical to the LoRA rank (80) is a common practice, as it helps to maintain the relative importance of the LoRA adaptation proportional to its rank, ensuring stable and effective fine-tuning.

Learning rate: 1×10^{-5}. This relatively low learning rate was chosen to ensure stable training over a large number of optimization steps, effectively preventing oscillations or divergence that can occur with higher rates, especially in complex reinforcement learning setups.

Batch size: 32. This value represents the number of training examples processed in one forward/backward pass. A batch size of 32 provided a good balance between stable gradient estimates and efficient memory utilization.

Number of generations: 32. This parameter refers to the number of samples (e.g., generated responses or trajectories) produced by the policy in a single optimization step for calculating the policy gradient. Keeping it identical to the batch size (32) often aligns well with the model's processing capabilities and memory constraints, ensuring that each generated sample contributes effectively to the policy update in a balanced manner.

Temperature: Experiments were conducted with temperature values of 1.1 and 1.2. Temperature in language models controls the randomness of predictions by scaling the logits before the softmax function. Higher values of temperature increase randomness (promoting exploration) whereas lower values make the model's predictions more deterministic (promoting exploitation). While temperature 1.1 achieved slightly higher accuracy during training (78.9% at 3400 steps) compared to temperature 1.2 (74.8% at 3400 steps). This is shown in figure 5.

Our actual evaluation revealed that temperature 1.2 performed marginally better on the evaluation dataset (77.0% accuracy) compared to temperature 1.1 (76.3% accuracy). This suggests that the increased randomness introduced by a higher temperature (1.2) facilitated better generalization and robustness during evaluation, potentially encouraging the model to explore a wider range of valid reasoning paths during training.

5.5. Hand-drawn Dataset

To further test the vision component and generalization capabilities of our model, particularly against less idealized and more challenging inputs, we created a supplementary dataset of 20 hand-drawn logic circuit images. These images inherently possess greater visual noise, stylistic variations, and less uniformity compared to their synthetically generated counterparts. For each of these hand-drawn circuits, 8 distinct questions were meticulously formulated. These questions mirrored those used for the synthetic data, which yielded a total of 160 question-answer pairs. Two JSON Lines files, `qa_dataset.jsonl` and `circuit.jsonl`, were also generated for this "handwritten" data subset, maintaining consistency in data structure with our primary synthetic dataset and stored under the dedicated 'handwritten' file of the 'data' directory. Testing against this challenging dataset, we found that the base model (Qwen2.5 VLM), without any GRPO fine-tuning, exhibited a low accuracy of 22.5%. In stark contrast, our GRPO model, employing the specified hyperparameters mentioned above, achieved a significantly higher accuracy of 53.1% with temperature 1.1 and an even more impressive 62.5% with temperature 1.2. This substantial performance improvement on hand-drawn data further corroborates the benefit of increased randomization (temperature 1.2) for improved generalization and robustness to real-world visual variability, which aligns with the trends observed earlier on the synthetically generated evaluation dataset.

5.6. Reward Function

Initially, our GRPO experiments utilized a binary reward system, providing a simple "correct" or "incorrect" signal. However, we found that this binary reward was insufficient for nuanced learning. Consequently, we experimented with incorporating F1 scoring as the reward function in our GRPO framework.

The F1 score - the harmonic mean of precision and recall - offers a more granular and balanced measure of correctness. It is particularly useful when dealing with situations where true positive and false positive classifications carry different weights, or when datasets are imbalanced, which means that some correct answers appear less frequently. Unlike a binary reward, the F1 score incentivizes the model to balance two key aspects: precision and recall.

The F1 score is a special case of the more general F_β measure, which allows weighting recall more heavily than precision (or vice versa) via the parameter β. When $\beta = 1$, the F_β measure becomes the F1 score, treating precision and recall as equally important:

$$F_\beta = (1 + \beta^2) \cdot \frac{\text{precision} \cdot \text{recall}}{\beta^2 \cdot \text{precision} + \text{recall}}$$

$$F1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

The F1 score penalizes models more heavily if there is a significant imbalance between precision and recall that compels the agent to achieve a strong performance across both. This shift from a binary to an F1-based reward system within GRPO significantly boosted the accuracy of two previously low-performing question types: "gate_count" and "primary_inputs_list", which demonstrates its effectiveness in guiding the agent towards more precise and complete answers for these specific queries by encouraging a balanced focus on both correctly identifying instances and avoiding incorrect predictions.

5.7. Results

The table below shows the base performance of the model before fine tuning, the fine tuned model without improvements submitted during the milestone, and the model's final performance after adding improvements to the training process such as dynamic sampling, an updated F1 reward function, and tuning hyperparameters such as num_generations and temperature.

Task	Model		
	Base	**Milestone**	**Final (Ours)**
gate_count	14.3%	76.0%	**82.0%**
gate_inputs	34.40%	**84.0%**	**84.0%**
gate_output_wire	24.8%	**89.7%**	88.0%
gate_type	35.3%	38.7%	**56.0%**
count_specific_gate_type	16.0%	N/A	**62.0%**
gate_fan_out	34.6%	N/A	**98.0%**
gate_input_count	19.7%	N/A	**97.0%**
primary_inputs_list	1.5%	N/A	**49.0%**
Average	22.6%	72.1%	**77.0%**

6. Conclusion

In this work, we addressed the challenge of enabling Vision-Language Models (VLMs) to interpret digital circuit diagrams, a task critical for technical domains. We introduced a novel dataset of synthetic and hand-drawn circuit diagrams and employed a modified version of Group Relative Policy Optimization (GRPO) to fine-tune the Qwen2.5-VLM-3B-Instruct model. Our experiments demonstrated a significant improvement in performance, with the GRPO-tuned model achieving 77.0% accuracy on synthetic diagrams and 62.5% on hand-drawn ones, a substantial increase from the base model's 22.5% accuracy. Enhancements such as dynamic sampling, an F1-based reward function, and careful hyperparameter tuning, including LoRA rank and temperature, were crucial in achieving these results.

Despite these advancements, there are areas for future exploration. Our current dataset does not include circuits with cyclic dependencies, which are common in real-world digital designs. Future work could involve expanding the dataset to include such complex circuits and increasing the diversity and size of the hand-drawn dataset to better represent real-world variability. Further refinement of the VLM architecture and training strategies could also lead to even higher accuracy and robustness, moving closer to a reliable AI agent for automated circuit analysis in the semiconductor industry.

7. Contributions

Z.Z. set up codebase, researched GRPO and DAPO to implement modifications to the training algorithms to improve performance, wrote Data and Methods section of report. A.D. wrote the script for SFT. ran SFT training, GRPO training, synthetic and hand-drawn dataset evaluations on SFT and GRPO, and conducted hyperparameter tuning. D.P. generated the synthetic and handwritten dataset. A.D., Z.Z., and D.P. wrote the paper. D.P. made the poster.

References

[1] S. Bai, K. Chen, X. Liu, J. Wang, W. Ge, S. Song, K. Dang, P. Wang, S. Wang, J. Tang, H. Zhong, Y. Zhu, M. Yang, Z. Li, J. Wan, P. Wang, W. Ding, Z. Fu, Y. Xu, J. Ye, X. Zhang, T. Xie, Z. Cheng, H. Zhang, Z. Yang, H. Xu, and J. Lin. Qwen2.5-VL Technical Report, 2025.

[2] B. Bohara and H. S. Krishnamoorthy. Computer Vision based Framework for Power Converter Identification and Analysis. In *2022 IEEE International Conference on Power Electronics, Drives and Energy Systems (PEDES)*, pages 1–6, 2022.

[3] H. Chen, H. Tu, F. Wang, H. Liu, X. Tang, X. Du, Y. Zhou, and C. Xie. SFT or RL? An Early Investigation into Training R1-Like Reasoning Large Vision-Language Models, 2025.

[4] DeepSeek-AI, D. Guo, D. Yang, H. Zhang, J. Song, R. Zhang, R. Xu, Q. Zhu, S. Ma, P. Wang, X. Bi, X. Zhang, X. Yu, Y. Wu, Z. F. Wu, Z. Gou, Z. Shao, Z. Li, Z. Gao, A. Liu, B. Xue, B. Wang, B. Wu, B. Feng, C. Lu, C. Zhao, C. Deng, C. Zhang, C. Ruan, D. Dai, D. Chen, D. Ji, E. Li, F. Lin, F. Dai, F. Luo, G. Hao, G. Chen, G. Li, H. Zhang, H. Bao, H. Xu, H. Wang, H. Ding, H. Xin, H. Gao, H. Qu, H. Li, J. Guo, J. Li, J. Wang, J. Chen, J. Yuan, J. Qiu, J. Li, J. L. Cai, J. Ni, J. Liang, J. Chen, K. Dong, K. Hu, K. Gao, K. Guan, K. Huang, K. Yu, L. Wang, L. Zhang, L. Zhao, L. Wang, L. Zhang, L. Xu, L. Xia, M. Zhang, M. Zhang, M. Tang, M. Li, M. Wang, M. Li, N. Tian, P. Huang, P. Zhang, Q. Wang, Q. Chen, Q. Du, R. Ge, R. Zhang, R. Pan, R. Wang, R. J. Chen, R. L. Jin, R. Chen, S. Lu, S. Zhou, S. Chen, S. Ye, S. Wang, S. Yu, S. Zhou, S. Pan, S. S. Li, S. Zhou, S. Wu, S. Ye, T. Yun, T. Pei, T. Sun, T. Wang, W. Zeng, W. Zhao, W. Liu, W. Liang, W. Gao, W. Yu, W. Zhang, W. L. Xiao, W. An, X. Liu, X. Wang, X. Chen, X. Nie, X. Cheng, X. Liu, X. Xie, X. Liu, X. Yang, X. Li, X. Su, X. Lin, X. Q. Li, X. Jin, X. Shen, X. Chen, X. Sun, X. Wang, X. Song, X. Zhou, X. Wang, X. Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Y. Zhang, Y. Xu, Y. Li, Y. Zhao, Y. Sun, Y. Wang, Y. Yu, Y. Zhang, Y. Shi, Y. Xiong, Y. He, Y. Piao, Y. Wang, Y. Tan, Y. Ma, Y. Liu, Y. Guo, Y. Ou, Y. Wang, Y. Gong, Y. Zou, Y. He, Y. Xiong, Y. Luo, Y. You, Y. Liu, Y. Zhou, Y. X. Zhu, Y. Xu, Y. Huang, Y. Li, Y. Zheng, Y. Zhu, Y. Ma, Y. Tang, Y. Zha, Y. Yan, Z. Z. Ren, Z. Ren, Z. Sha, Z. Fu, Z. Xu, Z. Xie, Z. Zhang, Z. Hao, Z. Ma, Z. Yan, Z. Wu, Z. Gu, Z. Zhu, Z. Liu, Z. Li, Z. Xie, Z. Song, Z. Pan, Z. Huang, Z. Xu, Z. Zhang, and Z. Zhang. DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning, 2025.

[5] H. Deng, D. Zou, R. Ma, H. Luo, Y. Cao, and Y. Kang. Boosting the Generalization and Reasoning of Vision Language Models with Curriculum Reinforcement Learning, 2025.

[6] R. O. Duda and P. E. Hart. Use of the Hough transformation to detect lines and curves in pictures. *Commun. ACM*, 1972.

[7] W. Fang, J. Wang, Y. Lu, S. Liu, Y. Wu, Y. Ma, and Z. Xie. A Survey of Circuit Foundation Model: Foundation AI Models for VLSI Circuit Design and EDA, 2025.

[8] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen. LoRA: Low-Rank Adaptation of Large Language Models, 2021.

[9] M. S. Islam, R. Rahman, A. Masry, M. T. R. Laskar, M. T. Nayeem, and E. Hoque. Are Large Vision Language Models up to the Challenge of Chart Comprehension and Reasoning? An Extensive Investigation into the Capabilities and Limitations of LVLMs, 2024.

[10] Q. Li, D. Liang, Y. Xu, N. Xiao, and Y. Li. Improved Algorithm for Circuit Diagram Image Recognition. In *Proceedings of the 2nd International Conference on Computer Science and Software Engineering*, CSSE '19, page 96–101, New York, NY, USA, 2019. Association for Computing Machinery.

[11] X. Li, Y. Sun, W. Cheng, Y. Zhu, and H. Chen. Chain-of-region: Visual Language Models Need Details for Diagram Analysis, 2025.

[12] J. Liu, C. Wang, C. Y. Liu, L. Zeng, R. Yan, Y. Sun, Y. Liu, and Y. Zhou. Improving Multi-Step Reasoning Abilities of Large Language Models with Direct Advantage Policy Optimization, 2024.

[13] A. Mehradfar, X. Zhao, Y. Niu, S. Babakniya, M. Alesheikh, H. Aghasi, and S. Avestimehr. AICircuit: A Multi-level Dataset and Benchmark for AI-Driven Analog Integrated Circuit Design, 2024.

[14] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. L. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, J. Schulman, J. Hilton, F. Kelton, L. Miller, M. Simens, A. Askell, P. Welinder, P. Christiano, J. Leike, and R. Lowe. Training language models to follow instructions with human feedback, 2022.

[15] R. Rafailov, A. Sharma, E. Mitchell, S. Ermon, C. D. Manning, and C. Finn. Direct Preference Optimization: Your Language Model is Secretly a Reward Model, 2024.

[16] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You Only Look Once: Unified, Real-Time Object Detection, 2016.

[17] Y. Ren and D. Xiong. CogAlign: Learning to Align Textual Neural Representations to Cognitive Language Processing Signals, 2023.

[18] Schemdraw. A Python library for producing high-quality electrical circuit schematic diagrams, 2025.

[19] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal Policy Optimization Algorithms, 2017.

[20] Z. Shao, P. Wang, Q. Zhu, R. Xu, J. Song, X. Bi, H. Zhang, M. Zhang, Y. K. Li, Y. Wu, and D. Guo. DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models, 2024.

[21] L. Skelic, Y. Xu, M. Cox, W. Lu, T. Yu, and R. Han. CIRCUIT: A Benchmark for Circuit Interpretation and Reasoning Capabilities of LLMs, 2025.

[22] S. N. Subramanyam. AI-Driven RTL Generation from Schematic Images: Revolutionizing IC Design. *International Journal For Multidisciplinary Research*, 2024.

[23] H. Wang, C. Qu, Z. Huang, W. Chu, F. Lin, and W. Chen. VL-Rethinker: Incentivizing Self-Reflection of Vision-Language Models with Reinforcement Learning, 2025.

[24] R. Xia, B. Zhang, H. Ye, X. Yan, Q. Liu, H. Zhou, Z. Chen, P. Ye, M. Dou, B. Shi, J. Yan, and Y. Qiao. ChartX and ChartVLM: A Versatile Benchmark and Foundation Model for Complicated Chart Reasoning, 2025.