Extended Abstract

Motivation Modern Vision-Language Models (VLMs) excel at text processing and complex reasoning but struggle to accurately interpret diagrams, charts, and graphs, which are essential in scientific and technical fields. Recent studies show that even state-of-the-art VLMs often fail to accurately recognize relational structures from visual information, relying instead on prior knowledge. Existing studies primarily use metadata extraction rather than purely vision-based relational understanding. By enabling end-to-end graph reasoning from pixels alone, such a capability could improve efficiency processing visual information across research and industry.

Method We fine-tuned a vision-language model (VLM) for graph reasoning using a reinforcement learning (RL) framework tailored to structured visual tasks. Specifically, we adopted Group Relative Policy Optimization (GRPO), leveraging a custom-designed dataset of synthetic graph images paired with structured question-action formats that target key aspects of graph comprehension. To further enhance training efficiency and model performance, we incorporated several techniques inspired by Decoupled Clip and Dynamic sAmpling Policy Optimization (DAPO), such as relaxed policy clipping, KL-free updates, and adaptive question skipping. Our study also includes two ablation experiments: one investigating the impact of question types during training, and another evaluating the affects of the DAPO-inspired enhancements.

Implementation With Qwen2.5-VL-3B as a base model, fine-tuning was conducted with the generated dataset. For all question types, responses received a format reward if they matched the structure specified in the prompt. An additional accuracy reward was assigned based on exact matches for simple tasks, and based on the F1-score for the complex task. Although supervised fine-tuning is typically performed before reinforcement learning, it was skipped because preliminary tests showed no performance gains. Low-Rank Adaptation (LoRA) was used to fine-tune only a small subset of layers and applied an exponential moving average skipping strategy, dropping the training samples whose question types scored above a certain threshold. For algorithmic enhancements, we removed the KL-divergence penalty and increasing the clipping threshold, which is known to yield faster convergence and higher reward for some tasks.

Results For simple question types that require only numeric or short categorical responses, the fine-tuned model achieved a notable improvement in accuracy, from an average baseline performance of 61.1% to 83.6%. For complex question types involving full graph structure prediction, the model attained an accuracy from a baseline performance of 31.1% to 95.8%. These results were obtained using GRPO fine-tuning augmented with DAPO-inspired enhancements, which effectively improved performance on the fine-tuning target and closely related tasks. However, this came at the cost of reduced generalization to tasks less correlated with the training objectives. In contrast, the vanilla GRPO framework yielded more modest improvements on the target task but maintained more consistent performance across a broader range of task types, including those not directly emphasized during training.

Discussion While our approach shows promising results, limitations remain. The synthetic dataset may not fully reflect real-world diagram diversity, and more complex tasks and broader domain coverage could improve generalizability. Also, additional training parameter tuning or using larger models may yield further gains. Real-world deployment, particularly in safety-critical applications, must be accompanied by careful oversight. Overall, challenges in model scaling, data design, and training stability underscore the need for continued refinement.

Conclusion We demonstrated that RL–based fine-tuning can significantly improve the graph reasoning capabilities of vision–language models. By combining GRPO, LoRA, and DAPO-inspired training enhancements, our approach boosted performance on graph understanding tasks. Dataset quality was critical, both in image generation and question-action design. Notably, we observed that training on global graph prediction implicitly transfers to simpler subtasks, suggesting implicit hierarchical learning. These results indicate that with well-designed data and training pipelines, end-to-end RL can effectively bridge raw visual inputs and structured graph understanding. Future work will explore real-world diagram settings, uncertainty-aware modeling, and hierarchical RL to further enhance robustness and interpretability.

Graph Reasoning-Tailored (GReaT) VLMs

Mike Zhao
Department of Electrical Engineering
Stanford University
zzhao24@stanford.edu

Raina Song
Department of Electrical Engineering
Stanford University
rainas@stanford.edu

Joonwon Kang
Department of Mechanical Engineering
Stanford University
jwkang@stanford.edu

Abstract

Vision–Language Models (VLMs) excel at text processing and general reasoning but still struggle to interpret diagrams, charts and graphs, the very formats often used to convey complex relationships in science and industry. State-of-the-art VLMs often fail to accurately recognizing relational structures from visual information, relying instead on prior knowledge. Existing studies primarily use metadata extraction rather than purely vision-based relational understanding. To overcome such limitations, we present an end-to-end reinforcement learning approach to fine-tune a Qwen2.5-VL-3B model for visual graph reasoning. Leveraging low-rank adaptation (LoRA) to reduce computational cost, we apply Group Relative Policy Optimization (GRPO) and introduce DAPO-inspired enhancements, relaxed clipping, omission of the KL penalty and adaptive sampling via EMA-based skipping, to accelerate convergence and improve stability. We train on a diverse synthetic dataset of directed graphs paired with structured question-action examples covering counting, connectivity, label naming and full graph listing. After one epoch, simple-task accuracy rises from 61.1% to 83.6% and complex-task accuracy from 30.4% to 89.0%; with DAPO methods, complex listing reaches 95.8%, and semantic subtasks (node and edge naming) improve by over 15 points. Qualitative analysis identifies off-by-one, occlusion and tokenization errors, pointing to future work on OCR robustness and hierarchical RL. These results demonstrate that targeted RL fine-tuning can bridge the gap between raw visual inputs and structured graph understanding without external metadata.

1 Introduction

In scientific and technical domains, professionals frequently interact with multimodal data that extends beyond text to encompass structured visual elements such as diagrams, flowcharts, and graphs. These graphical representations play a crucial role in conveying complex relationships, procedural flows, and hierarchical structures. An Artificial Intelligence (AI) system capable of understanding the structural and relational information embedded in such visuals could significantly advance tasks involving analysis, automation, and decision-making. While large vision-language models (VLMs) have shown strong performance in text processing and reasoning, their capabilities in interpreting the underlying structure of visual graphs remain limited.

Unlocking this capability could have far-reaching implications. Just as Large Language Models (LLMs) have transformed software engineering by streamlining code generation, debugging, and documentation, VLMs with graph comprehension skills could revolutionize workflows that depend

heavily on visual information. Such systems could support or automate a broad range of tasks, including structural pattern recognition, visual information extraction, diagram-to-text conversion, and the integration of visual and textual content for more comprehensive analysis. This would reduce reliance on manual interpretation, increase accuracy, and enhance productivity across a wide range of scientific, engineering, and data-intensive disciplines.

We fine-tune Qwen2.5-VL-3B-Instruct model Bai et al. (2025) using a custom dataset composed of synthetic graphs paired with structured question-action examples. The base model had already demonstrated enough baseline performance to start reinforcement learning (RL) fine-tuning via GRPO, achieving over 30% accuracy across all question types, which led us to forego Supervised Fine-Tuning (SFT) after preliminary tests showed no substantial benefit.

Through GRPO fine-tuning, the model's performance improved significantly: accuracy on simpler questions rose from 61.1% to a peak of 83.6%, while accuracy on more complex, structure-dependent questions increased from 31.1% to 95.8%. These results demonstrate that with targeted training, VLMs can develop a meaningful understanding of graph structures and move closer to general multimodal reasoning capabilities.

2 Related Work

2.1 Computer Vision–Based Approaches

Early diagram understanding relied on classical computer vision pipelines: image preprocessing (e.g., binarization, skeletonization), component detection via pattern matching or SVMs, line and junction detection (Hough Transform), and OCR to recover text. Skeletonization and binarization reduce noise but can lose fine detail; line detection by Hough voting Duda and Hart (1972) and object detectors such as YOLO Redmon et al. (2016) were layered to reconstruct flowchart graphs. Pansare et al. built a multi-stage system that detects shapes, arrows, and text to build explicit graph models for flowchart-to-code conversion Pansare et al. (2019). Julca-Aguilar and Hirata showed Faster R-CNN can generalize symbol detection across handwritten graphics Julca-Aguilar and Hirata (2017), and Kembhavi et al. introduced Diagram Parse Graphs for syntactic parsing and reasoning over diverse diagrams Kembhavi et al. (2016). While robust on constrained inputs, these methods remain sensitive to style variations and lack end-to-end semantic reasoning.

2.2 Vision–Language Model and Metadata-Augmented Approaches

Vision–language models (VLMs) have enabled end-to-end diagram understanding by jointly processing visual and textual cues. Benchmarks such as DVQA Kafle et al. (2018) and PlotQA Methani et al. (2019) demonstrate that VLMs struggle with visual reasoning beyond simple OCR and require specialized datasets. Pan et al. (2024) evaluated performance of large vision–language models on flowchart understanding using a custom dataset of flow diagrams. Although several models achieved high accuracy on node recognition, all failed to surpass an F1 score of 0.3 on link recognition. The authors also applied a Chain-of-Thought (CoT) prompt template, which unexpectedly degraded link extraction performance. Hou et al. (2024) tested three state-of-the-art LVLMs on diagram reasoning tasks and found that models often rely on prior world knowledge rather than structural cues from the diagrams, leading to frequent errors in relational inference. The study highlights a core limitation: current VLMs lack deep visual–structural understanding. To mitigate this, recent studies integrate metadata alongside visual input: ChartInstruct Masry et al. (2024) first extracts chart data tables before LLM reasoning, and AskChart Yang et al. (2024) incorporates axis labels and captions via a mixture-of-experts to boost semantic extraction. Such metadata-augmented pipelines achieve superior performance but depend on availability of underlying data or code. There remains a gap for models that infer relational structure directly from raw visuals without metadata assistance.

2.3 Reinforcement Learning–Based Fine-Tuning of Vision–Language Models

Contemporary work has begun to harness RL techniques to directly optimize VLMs for complex reasoning tasks. Shao et al. Shao et al. (2024) introduce GRPO, a reward-weighted policy gradient approach that significantly improves mathematical reasoning in open language models. More recently, Yu et al. Yu et al. (2025) propose DAPO (Decoupled Clip and Dynamic sAmpling Policy

Optimization), which combines decoupled clipping of importance ratios with dynamic sampling to stabilize and accelerate large-scale RL fine-tuning while maintaining high performance.

3 Method

3.1 Data

3.1.1 Synthetic Graph Image Generation

The synthetic data generation process creates randomized directed graphs with varying difficulty levels, scaled from 1 to 5. We randomly select a graph type from grid, tree, scale-free, visibility, fully connected, bipartite and uniform to be the base graph. We then vary the size, depth, fan-out, cycles, parallel paths, and edge label density of the graphs for different difficulty levels. To enhance dataset variability, randomized visual elements such as colors, rotations, and positions were also incorporated into the generation. Example synthetic graph images are shown in Figure 1.

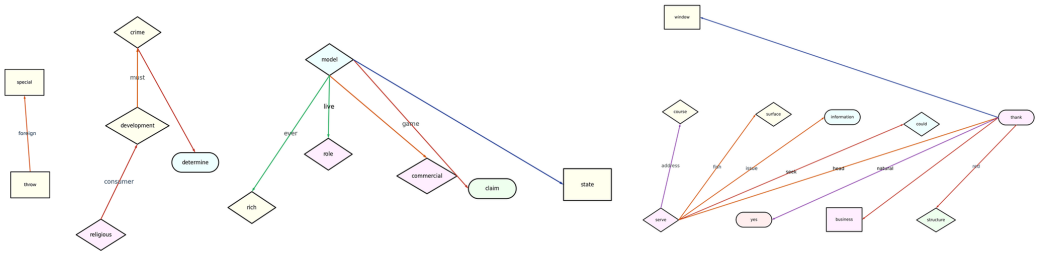

Figure 1: Example synthetic graph data with different difficulties

3.1.2 Question-Action Pair Generation

In addition to generating synthetic images, we constructed corresponding question-action pairs designed to produce structured outputs rather than binary (yes/no) responses, thereby mitigating the risk of overfitting. We defined six distinct question types aimed at evaluating the model's capacity for structural graph understanding, with a focus on tasks such as quantitative assessment of visual elements and analysis of topological relationships within the graph. Examples of the question types employed in this study are provided below.

- **Simple Tasks:**
  1. **Node Counts** The total number of nodes in the graph.
  2. **Edge Counts** The total number of edges in the graph.
  3. **Node Connection Counts** The total number of nodes connected to a given node.
  4. **Naming Node** The name of the node connected to a given node with a specified edge.
  5. **Naming Edge** The label of the edge connecting two specified nodes.
- **Complex Task: Entire Node and Edge Listing**
  A complete list of *all* node and edge connections in the image.

Example question types are shown below, based on the graph image in Figure 2.

Question Type: `Node Connection Counts`
Conversation:

- **Prompt:** `<image> How many nodes are connected with the node named 'through' in the given flowchart image? Provide the answer as a single integer (e.g., 3). If the node does not exist, the answer is 0.`
- **Answer:** `5`

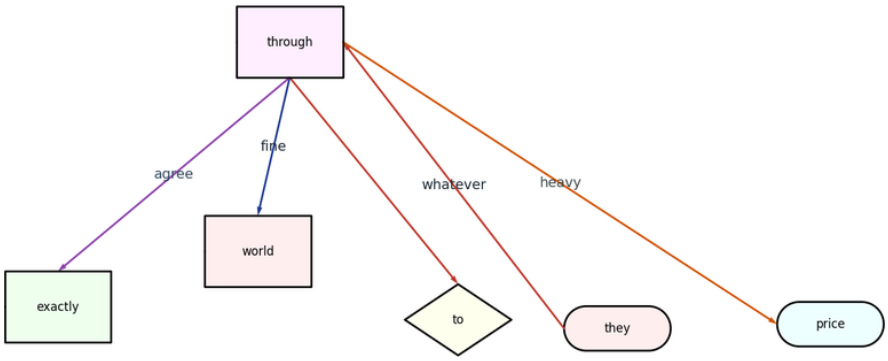

Figure 2: Target Graph Image for Showing Question Action Pair Examples

Question Type: `Naming Edge`
Conversation:

- **Prompt:** `<image> What is the label of the edge connecting 'they' and 'through'? Provide the edge label as a string. If there is no label or the edge/nodes do not exist, answer as None.`
- **Answer:** `whatever`

Question Type: `Entire Node and Edge Listing`
Conversation:

- **Prompt:** `<image> Describe the entire graph structure, listing all nodes and edges. Provide the answer by first listing all nodes (e.g., 'nodes: NodeA, NodeB, NodeC') and then the list of all edges. The list of edges should be prefixed by 'edges:', and each edge should be on a new line. For edges, use the format 'NodeA -()-> NodeB' for unlabeled edges and 'NodeA -(label)-> NodeB' for labeled edges. Preserve original casing for node names and labels.`
- **Answer:**

```
nodes: exactly, price, they, through, to, world
edges:
they -(whatever)-> through
through -()-> to
through -(agree)-> exactly
through -(fine)-> world
through -(heavy)-> price
```

3.2 Base Model

For our experiments, we selected `Qwen2.5-VL-3B Instruct` as the base model Bai et al. (2025). Qwen2.5-VL is a family of VLMs designed to handle multimodal inputs, integrating both textual and visual understanding capabilities. While larger variants such as 7B and 32B are available, we opted for the 3B model due to its favorable trade-off between performance and computational efficiency. Despite its smaller size, the 3B model demonstrated sufficient baseline competence across various question types, achieving over 61.1% accuracy without any task-specific fine-tuning. This level of performance made it a viable candidate for RL-based fine-tuning. Moreover, our choice was influenced by practical constraints, including limited GPU memory and compute resources, as well as the need to iterate rapidly within a constrained timeline.

3.3 Low-Rank Adaptation (LoRA)

To further reduce the computational burden associated with fine-tuning large models, we adopted the LoRA technique Hu et al. (2022). LoRA improves training efficiency by introducing trainable

low-rank matrices into the Transformer architecture, while keeping the majority of the pre-trained weights frozen. Specifically, for a pre-trained weight matrix $W_0 \in \mathbb{R}^{d\times k}$, the update is expressed as a low-rank decomposition $\Delta W = BA$, where $B \in \mathbb{R}^{d\times r}$ and $A \in \mathbb{R}^{r\times k}$, with $r \ll \min(d, k)$. The resulting adapted weight is:

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

where α is a scaling factor. This approach dramatically reduces the number of trainable parameters and memory usage, enabling efficient fine-tuning without significantly compromising performance, making it well-suited for our resource-constrained training pipeline.

3.4 Group Relative Policy Optimization (GRPO)

GRPOShao et al. (2024) was utilized as our main finetuning method. GRPO is a variant of Proximal Policy Optimization (PPO) Schulman et al. (2017) that eliminates the need for a value function estimator, thereby reducing computational overhead. For each state s, the algorithm samples a group of G actions $\{a_1, a_2, \ldots, a_G\}$ from the current policy π_{θ_t}. The rewards for these actions are denoted as $r(s, a_1), r(s, a_2), \ldots, r(s, a_G)$. The mean and standard deviation of these rewards are computed as:

$$\mu = \frac{1}{G}\sum_{i=1}^{G} r(s, a_i), \quad \sigma = \sqrt{\frac{1}{G}\sum_{i=1}^{G}(r(s, a_i) - \mu)^2}$$

The group-relative advantage for each action a_i is then calculated using the standardized score:

$$A^{\pi_{\theta_t}}(s, a_i) = \frac{r(s, a_i) - \mu}{\sigma}$$

The objective function aims to maximize the expected advantage while ensuring that the updated policy does not deviate significantly from the current policy.

$$\max_{\theta} \frac{1}{G}\sum_{i=1}^{G} \mathbb{E}_{(s,a_1,\ldots,a_G)\sim\pi_{\theta_t}} \left[\begin{cases} \min\left(\frac{\pi_\theta(a_i|s)}{\pi_{\theta_t}(a_i|s)}, 1+\epsilon\right) A^{\pi_{\theta_t}}(s, a_i) & \text{if } A^{\pi_{\theta_t}}(s, a_i) > 0 \\ \max\left(\frac{\pi_\theta(a_i|s)}{\pi_{\theta_t}(a_i|s)}, 1-\epsilon\right) A^{\pi_{\theta_t}}(s, a_i) & \text{if } A^{\pi_{\theta_t}}(s, a_i) < 0 \end{cases} \right] \quad (1)$$

Here, ϵ is a hyperparameter that controls the extent of policy update, preventing large deviations and ensuring stable training. By focusing on the relative performance of actions within a group, GRPO encourages the policy to favor actions that perform better than their peers, leading to improved reasoning capabilities in LLMs without the computational burden of value function estimation.

3.5 Reward Function Design

For all question types, responses were evaluated using a two-part reward mechanism. First, a *format reward* was granted when the output adhered to the structural constraints specified in the prompt. Second, an *accuracy reward* was assigned based on exact matches with the ground truth. This binary reward strategy was sufficient for simple tasks with discrete, unambiguous answers.

However, the complex question requiring the identification of all nodes and edges in the graph—could not be adequately evaluated using a binary reward. In this case, partial correctness is meaningful and should be rewarded accordingly. To better capture this, we incorporated the F1 score as a more informative reward signal within the GRPO framework.

The F1 score, defined as the harmonic mean of precision and recall, offers a more granular and balanced measure of correctness. It is especially useful in settings involving imbalanced outputs or where both over-prediction and under-prediction should be penalized. Unlike binary rewards, the F1 score encourages the model to optimize both precision (the fraction of predicted elements that are correct) and recall (the fraction of correct elements that are predicted), leading to more accurate and comprehensive outputs.

Based on the calculated precision and recall, we calculated the score as below:

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

This continuous, fine-grained reward enabled more effective policy learning for the complex graph-understanding task.

3.6 Training Enhancements Inspired by DAPO

To further improve training efficiency and model performance, we incorporated several optimization strategies motivated by Dynamic sAmpling Policy Optimization (DAPO) Yu et al. (2025). These modifications were designed to allow more flexible policy updates and accelerate convergence:

Relaxed Clipping Range (Clip Higher) We increased the upper bound of the policy ratio clipping range during updates. While the standard GRPO framework applies symmetric clipping to constrain the policy ratio, DAPO suggests that relaxing this bound—particularly on the upper end—can facilitate faster convergence by permitting larger updates when the estimated advantage is high. This adjustment enables the model to make more confident improvements without being overly restricted by conservative update bounds.

Omission of KL Divergence Constraint The original GRPO objective includes a Kullback-Leibler (KL) divergence penalty to regularize the updated policy against the previous one, as is common in PPO-style algorithms. The KL constraint is typically formulated as:

$$-\beta D_{\mathrm{KL}}\left[\pi_\theta(\cdot|s_t) \parallel \pi_{\theta_{\text{old}}}(\cdot|s_t)\right]$$

This term has been included in the maximization target introduced in Equation 1 in the original GRPO. While this constraint is useful for stabilizing training, prior work including DAPO has shown that removing or down-weighting the KL term can accelerate learning, especially when rapid deviation from the initial policy is desirable. Given our goal of substantial behavior change through fine-tuning, we omitted the KL penalty to grant the model greater flexibility in policy exploration.

These training adjustments collectively allow the agent to explore the policy space more freely and adapt more quickly, particularly in settings where fast learning and structural changes from the base model are crucial.

Adaptive Sampling via EMA-Based Skipping To enhance training efficiency and focus on underperforming areas, we developed an adaptive sampling scheme inspired by dynamic sampling from DAPO.

For each question type, we maintain a running exponential moving average (EMA) of performance. When the EMA exceeds a high-confidence threshold (e.g., 95%), we reduce the selection probability for that question type in future training batches, capping the maximum `skip_probability` at 0.9. This allows us to deprioritize well-learned tasks without completely excluding them, reallocating training capacity to more challenging areas. This method is reversible such that when the EMA for a downsampled task drops below the threshold, its sampling probability is restored, ensuring continued attention to tasks that may regress.

This adaptive sampling mechanism streamlines training by concentrating updates where they are needed most and promotes balanced generalization across tasks, reducing the risk of overfitting to easier categories.

4 Experimental Setup

4.1 Dataset Experiments

For the two different datasets, Simple and Complex tasks (as mentioned in section 3.1.2):

We would like to answer the following questions:

1. Will a VLM model trained on ONLY the Simple dataset be able to solve a harder task like in the Complex dataset?
2. Conversely, will a VLM model trained on ONLY the Complex dataset be able to master easier subtasks like in the Simple dataset?

- **Simple tasks dataset:** This is used to test a specific visual skill of the VLM, such as counting objects, object comprehension, and graph connectivity.
- **Complex tasks dataset:** This is used to test a VLM's ability to leverage multiple visual skills (label, object comprehension, connectivity) in addition to reasoning. Given an image, the model's job is to name all the nodes and edges (along with edge labels).

Since training on a 3B VLM model is time-consuming (around 12 hours per experiment), we limit our training samples for this experiment to 1000 training pairs (images, question).

4.2 Algorithmic Experiments

To assess the impact of the DAPO-inspired modifications on GRPO, we use a dataset of the complex etire-listing task only (2,000 samples). We modified the GRPO algorithm by removing the KL-divergence penalty and increasing the clipping threshold. We then compare the results with the results generated from the base GRPO algorithm on the same dataset.

Both experiments leverage the VLM-R1 fine-tuning framework, which provides a GRPO-style training pipeline for vision–language models. By building directly on this open-source library, we ensure a controlled comparison between the original algorithm and our DAPO-inspired enhancements, isolating the effect of each change on convergence speed, stability, and final performance.

5 Results

5.1 Quantitative Evaluation

5.1.1 Training on the Simple vs Complex dataset

The table below shows the overall accuracy of each model on simple and complex tasks, while the bar chart details accuracy for specific tasks.

Table 1: Accuracy (%) on Simple vs. Complex tasks trained using DAPO (1000 samples)

Task	Base	Simple (DAPO)	Complex (DAPO)
Simple Task	61.1	**83.6**	**81.3**
Complex Task	30.4	42.3	**89.0**

Fine-tuning the base model leads to large improvements on both simple and complex reasoning tasks. Training on the Simple dataset improves accuracy on simple tasks from **61.1%** to **83.6%**. It performs up to **95.7%** on edge counts and **95.6%** on node counts, but only **59.1%** on naming nodes and **72.6%** on naming edges. However, it only improves the complex task from **30.4%** to **42.3%**, so it does not work well for the complex question.

Training on the Complex dataset improves complex-task accuracy from **30.4%** to **89.0%** while still doing well on simple tasks it was not trained on: **92.6%** on node counts, **75.2%** on edge counts, **75.8%** on connectivity, and **87.5%** on naming nodes. This shows that training on the complex task teaches the model both global graph structure and low-level features, while training only on simple tasks leaves it struggling with higher-order reasoning.

5.1.2 GRPO vs DAPO

When fine-tuning on the complex "entire listing" task, DAPO-inspired GRPO improves edge-naming accuracy from **81.2%** to **85.9%** and node-naming accuracy from **62.5%** to **80.3%**. Connectivity reasoning also sees a substantial gain, with "nodes connected" accuracy rising from **65.7%** to **88.9%**. This suggests that relaxed clipping and adaptive sampling enable the model to develop more robust label embeddings and to better disambiguate visually similar terms.

Conversely, the model's accuracy on pure counting tasks decreases. Specifically, edge-counting accuracy drops from **69.5%** to **55.3%**, and node-counting accuracy falls from **85.9%** to **60.7%**.

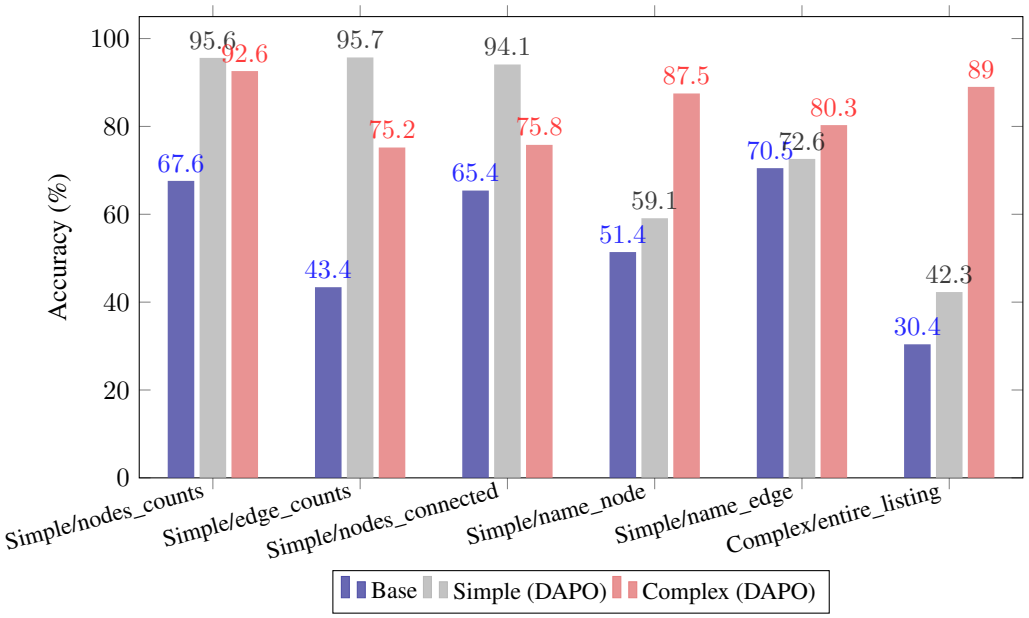

Figure 3: Per-task accuracy for simple tasks and the single complex task.

Table 2: Ablation Study: Training using the base GRPO algorithm vs. our modified DAPO algorithm trained on the Complex dataset (2000 samples)

Task	Base	Complex (GRPO)	Complex (DAPO)
Simple Tasks			
Name edge	70.5%	81.2%	**85.9%**
Name node	51.4%	62.5%	**80.3%**
Edge count	43.4%	**69.5%**	55.3%
Node count	67.6%	**85.9%**	60.7%
Nodes connected	65.4%	65.7%	**88.9%**
Complex Task			
Entire listing	31.1%	91.4%	**95.8%**

5.2 Qualitative Analysis

5.2.1 Effects of Training Data:

Quantitatively, the model achieves **95.7%** accuracy on edge-counting and **95.6%** on node-counting tasks, demonstrating near-perfect performance on basic structural recognition. Most remaining errors stem from "off-by-one" cases: for example, overcounting edges at tight junctions or missing isolated nodes in sparse graphs, as illustrated in Figure 4. These layout-related issues are rare, which is why the miscount rate is below 5%. Additionally, the model got **94.1%** accuracy on connected-node task, indicating that it groups nodes by reachability reliably. Occasional failures occur in cases of overlapping or occluded arrows or nodes, where the model may misinterpret visual cues. However, these results indicate that the model has effectively mastered simple graph reasoning tasks such as counting and connectivity.

Although not as high as the performance for simple question types, when fine-tuned with the complex entire-listing task, it showed a notable improvements compared to the base model. To our surprise, it even performed better on node and edge naming tasks than the fine-tuned model trained with those question types. Regarding that tracking all the names of nodes and edges is crucial to accomplish the complex task, this suggests that mastering global graph structure implicitly teaches lower-level subtasks, pointing to a natural hierarchical learning effect.

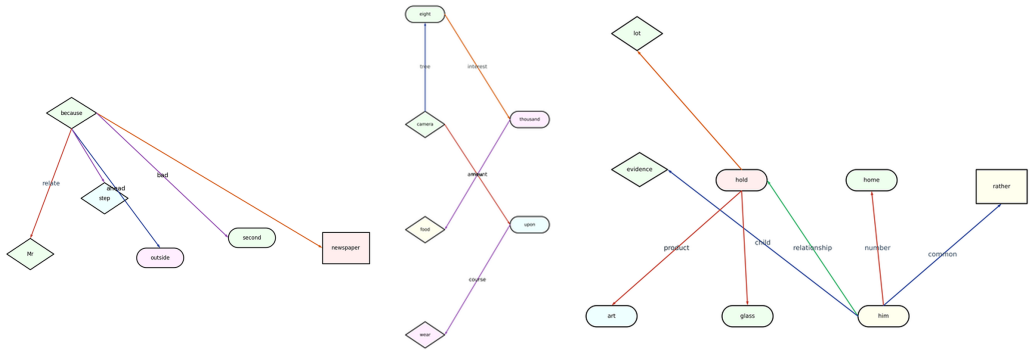

Figure 4: Examples of data with layout overlaps or errors

5.2.2 Effects of GRPO vs DAPO:

For the Complex dataset, the accuracy increased from 30.4% (base) to 91.4% (GRPO). If we add our algorithmic modifications, the accuracy further increases to **95.8%** (DAPO). This result confirms the findings of the DAPO paper, showing that DAPO techniques, such as relaxed clipping and dynamic sampling, significantly improve training speed and are still relatively stable. In our experiments, DAPO enhancements reached GRPO's final accuracy in roughly half the training steps.

On the other hand, the performance for simple question types uncorrelated with the complex entire-listing task was significantly degraded with DAPO enhancements. This decline in performance is likely due to the model's relatively small size of 3 billion parameters and the absence of a KL divergence penalty in our DAPO process. By specializing the model for complex tasks, its learned weights appear to shift, which negatively affects its generalization capabilities for less related out-of-distribution (OOD) tasks like node and edge counting. Without the KL divergence term, this shift is unbounded, leading to the performance degradation of less related tasks.

6 Discussion

6.1 Limitations

Despite substantial gains, our approach has several places for improvement. First of all, our synthetic dataset, while diverse, may not capture the full variety of real-world diagram styles, layouts, or noise patterns. Domain shifts, such as unconventional color schemes or handwritten annotations, could degrade performance. Second, we have only checked the graph structure understanding performance with one basic question type. Questions requiring more sophisticated reasoning and semantic understanding, such as asking for the shortest path between two nodes, can be added to further improve practicality. Finally, we fine-tuned for only one epoch and used a single model size (3B parameters); further training and exploration of larger or ensemble models may yield additional improvements.

6.2 Broader Impact

Enabling VLMs to interpret graphs and diagrams can accelerate research and development in fields ranging from scientific publishing to industrial automation. Automated diagram understanding could streamline tasks such as data extraction from chemical pathway charts, software architecture analysis, or financial flow diagrams. However, misinterpretation of safety-critical diagrams (for example, engineering schematics or medical flowcharts) could lead to incorrect decisions. Careful human oversight and uncertainty quantification must accompany any deployment in high-stakes environments.

6.3 Comments and Difficulties

Balancing model capacity, computational efficiency, and dataset complexity posed several challenges throughout the project. Fine-tuning a 3B-parameter vision–language model necessitated careful configuration of LoRA to prevent GPU memory overflow, and stabilizing GRPO-based training required extensive hyperparameter tuning, particularly for the learning rate and clipping range. Additionally, constructing a high-quality synthetic graph dataset demanded multiple rounds of refinement. Initial graph designs were often too simplistic, enabling the model to overfit rapidly, whereas excessively complex structures hindered effective learning. To overcome these limitations, we incorporated auxiliary techniques described in Section 3, which proved essential for maintaining learning efficacy on more structurally sophisticated graph instances.

7 Conclusions

In this work, we demonstrated that reinforcement learning–based fine-tuning can significantly enhance the ability of a vision–language model to perform graph reasoning. By applying GRPO with LoRA and further incorporating DAPO-inspired enhancements, we raised simple question accuracy from 61.1% to over 83% and complex question accuracy from 31.1% to 95.8%. Qualitative analysis revealed that counting tasks are nearly solved while naming tasks require improved OCR and semantic disambiguation. We also showed that training on a global graph-listing task implicitly teaches lower-level subtasks, suggesting a natural hierarchical learning effect.

Throughout the entire project, we found that dataset quality was crucial. The careful selection of images and the generation of question-action pairs were the most significant factors contributing to high performance. Additionally, the DAPO-inspired method demonstrated meaningful improvements, particularly in enhancing training efficiency.

These findings indicate that, with carefully tailored data and training methods, end-to-end RL fine-tuning can bridge the gap between raw visual inputs and structured graph understanding. Future extensions will focus on real-world diagram domains, uncertainty estimation, and hierarchical RL formulations to further close the remaining naming accuracy gap and ensure robust deployment in practical applications.

8 Team Contributions

- **Mike Zhao**: Performing "warm start" SFT initialization of the VLM for GRPO/DAPO. Designing the reward function so the model won't struggle with extremely sparse rewards. Building a custom training server to accelerate training.
- **Raina Song**: Synthetic dataset generation for simple, complex questions types for both GRPO and DAPO methods, explore other algorithms and design enhancements for the GRPO method such as multi-task and hierarchical reinforcement learning. Poster and report writing.
- **Joonwon Kang**: Building a training framework for GRPO fine-tuning using custom-built reward functions, question-action pair generation for simple question types, evaluating the performance of the models, conducting a literature review, and writing outlines for the documents.

Changes from Proposal In general, there were no notable changes from the proposal.

References

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, et al. 2025. Qwen2. 5-vl technical report. *arXiv preprint arXiv:2502.13923* (2025).

Richard O. Duda and Peter E. Hart. 1972. Use of the Hough transform to detect lines and curves in pictures. *Commun. ACM* 15, 1 (1972), 11–15. https://doi.org/10.1145/361237.361242

Yifan Hou, Buse Giledereli, Yilei Tu, and Mrinmaya Sachan. 2024. Do Vision-Language Models Really Understand Visual Language? *arXiv preprint arXiv:2410.00193* (2024).

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. 2022. Lora: Low-rank adaptation of large language models. *ICLR* 1, 2 (2022), 3.

Frank D. Julca-Aguilar and Nina S. T. Hirata. 2017. Symbol detection in online handwritten graphics using Faster R-CNN. *arXiv preprint arXiv:1712.04833* (2017).

Kushal Kafle, Brian Price, Scott Cohen, and Christopher Kanan. 2018. DVQA: Understanding Data Visualizations via Question Answering. In *Proc. CVPR*.

Aniruddha Kembhavi, Mike Salvato, Eric Kolve, Minjoon Seo, Hannaneh Hajishirzi, and Ali Farhadi. 2016. A Diagram Is Worth A Dozen Images. *arXiv preprint arXiv:1603.07396* (2016).

Ahmed Masry, Mehrad Shahmohammadi, Md Rizwan Parvez, Enamul Hoque, and Shafiq Joty. 2024. ChartInstruct: Instruction Tuning for Chart Comprehension and Reasoning. In *Findings of ACL*. 10387–10409.

Nitesh Methani, Pritha Ganguly, Mitesh M. Khapra, and Pratyush Kumar. 2019. PlotQA: Reasoning over Scientific Plots. *arXiv preprint arXiv:1909.00997* (2019).

Huitong Pan, Qi Zhang, Cornelia Caragea, Eduard Dragut, and Longin Jan Latecki. 2024. Flowlearn: Evaluating large vision-language models on flowchart understanding. In *ECAI 2024*. IOS Press, 73–80.

Aditya Pansare et al. 2019. Extracting Flowchart Features into a Structured Representation. `https://adityapansare.github.io/assets/GilbrethBlackbook.pdf`.

Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. 2016. You Only Look Once: Unified, Real-Time Object Detection. In *Proc. CVPR*.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347* (2017).

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Y Wu, et al. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300* (2024).

Xudong Yang, Yifan Wu, Yizhang Zhu, Nan Tang, and Yuyu Luo. 2024. AskChart: Universal Chart Understanding through Textual Enhancement. *arXiv preprint arXiv:2412.19146* (2024).

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, et al. 2025. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476* (2025).

A Additional Experiments

B Implementation Details

Used GPU NVIDIA H100 for all training experiments

B.1 Experiment 1: Simple vs Complex

LoRA rank: 64
LoRA alpha: 64

All models were trained for one epoch with a learning rate of 2×10^{-5}. For simple tasks, 18,967 question-answer pairs were used for training and 2,080 for testing; for complex tasks, 2,497 pairs were used for training and 272 for testing.

B.2 Experiment 2: GRPO vs DAPO

All models were trained for one epoch with a learning rate of 4×10^{-5}. For complex tasks, 4,235 pairs were used for training and 472 for testing. LoRA rank: 64
LoRA alpha: 64
ema_alpha 0.1
ema_skip_threshold 0.9
ema_skip_max_probability 0.9
ema_skip_scale_factor 9