ProcVQA: Benchmarking the Effects of Structural Properties in Mined Process Visualizations on Vision–Language Model Performance

Kazi Tasnim Zinat*, **Saad Mohammad Abrar***, **Shoumik Saha**, **Sharmila Duppala**, **Saimadhav Naga Sakhamuri** and **Zhicheng Liu**
University of Maryland
College Park, MD
{kzintas, sabrar, smksaha, sduppala, ssakhamu, leozcliu}@umd.edu

Abstract

Vision-Language Models have shown both impressive capabilities and notable failures in data visualization understanding tasks, but we have limited understanding on how specific properties within a visualization type affect model performance. We present ProcVQA, a benchmark designed to analyze how VLM performance can be affected by structure type and structural density of visualizations depicting frequent patterns mined from sequence data. ProcVQA consists of mined process visualizations spanning three structure types (linear sequences, tree, graph) with varying levels of structural density (quantified using the number of nodes and edges), with expert-validated QA pairs on these visualizations. We evaluate 21 proprietary and open-source models on the dataset on two major tasks: visual data extraction (VDE) and visual question answering (VQA) (with four categories of questions). Our analysis reveals three key findings. First, models exhibit steep performance drops on multi-hop reasoning, with question type and structure type impacting the degradation. Second, structural density strongly affects VDE performance: hallucinations and extraction errors increase with edge density, even in frontier models. Third, extraction accuracy does not necessarily translate into strong reasoning ability. By isolating structural factors through controlled visualization generation, ProcVQA enables precise identification of VLM limitations. ProcVQA is available at: https://github.com/kzintas/ProcVQA.

1 Introduction

Vision–language models (VLMs) have shown both considerable promise and notable failures in interpreting data visualizations, a domain that requires cross-modal grounding between visual structures and symbolic encodings (Liu et al., 2023a; Zhu et al., 2023). While prior work suggests that models struggle with specialized charts such as gantt or funnel charts over basic line or bar charts (Han et al., 2023), and complex visualizations (e.g., more subplots) impair model performance (Wang et al., 2024), there remains limited insight into which specific features within a visualization type influence model performance. One reason for this gap lies in the nature of existing benchmark datasets (tab. C.1), which are either synthetically generated (Xia et al., 2024) or curated from diverse online sources (Pew Research Center, 2025; OWID, 2025; Tableau Public, 2025). Although useful, these datasets generally lack systematic control over visualization features, making it difficult to rigorously assess their roles in shaping VLM performance.

As one of the first attempts to address this gap, in this paper we focus on a particular type of visualization: node-link visualizations depicting mined process patterns (hereafter referred to as *mined process visualizations*). Such visualizations show frequent patterns extracted from a set of event sequences, where the patterns highlight important events, workflows, and dependencies. They have been extensively studied in the areas of visual analytics and process mining and incorporated into commercial tools for various application domains such as clinical pathways (Kwon et al., 2021; Magallanes et al., 2022), customer journeys (Liu et al., 2017), and workflow analysis (Allard et al., 2019).

We introduce ProcVQA, a benchmark of 118 mined process visualizations from eight real-world domains. The benchmark incorporates controlled variations on two visualization properties: *structure type* (three types of common node-link structures: linear sequences, trees, and graphs), and *structure density* (quantified by node and edge counts). Using a standardized visualization generation pipeline (Zinat et al., 2023), we control for encodings, typography, color, and glyphs, removing stylistic confounds that hinders analyzing structural effects in other benchmarks. This enables isolation of structural factors that affect VLM performance.

* Equal contribution.

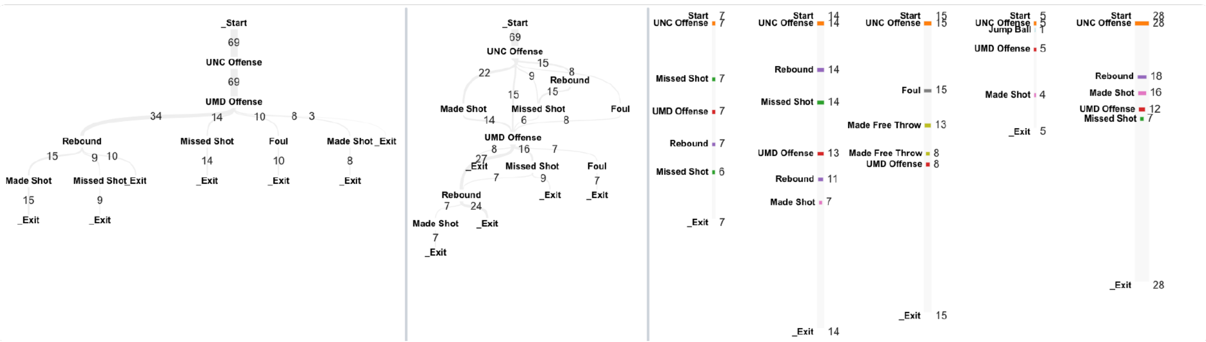


Figure 1: Example process visualizations across three node-link charts: (left) `tree`, (center) `graph`, and (right) `linear sequence clusters` summarizing similar event flows. These charts are depicting a basketball play sequence. ProcVQA uses such charts to evaluate VLMs' capabilities in interpreting sequential structures.

We evaluate 21 VLMs across two complementary tasks: visual data extraction (VDE) and visual question answering (VQA). The VQA task is further categorized by *reasoning complexity*, with questions requiring either single-hop (*value extraction, sequential reasoning*) or multi-hop (*value aggregation, unanswerable detection*) reasoning (tab. A.3). This design allows us to analyze VLM performance across three dimensions: *structure type, structural density, and reasoning complexity.*

Our findings show that current VLMs struggle with multi-hop reasoning, with performance varying by question type and visualization structure. Extraction accuracy declines as structural density increases, with the degree of degradation differing across models. Structure type also affects extraction accuracy, with most models performing weakest on graphs.

To summarize, our contributions include:

- ProcVQA, the first benchmark created for evaluating VLM performance on process visualizations across three structure types with varying structural densities.
- A comprehensive multi-dimensional analysis revealing how structural type, structural density, task, and reasoning complexity interact to affect VLM performance.

2 ProcVQA Benchmark

2.1 Chart Corpus

We curate a corpus of 118 process visualizations covering datasets from eight domains (app. A.1) and three process visualization types (app. A.3). These visualizations represent mined patterns (app. A.2) from real-world event sequence data, depicting sequential and numeric information. Fig. 1 illustrates our three visualization types: *trees* showing hierarchical event relationships with single parent nodes (left), *graphs* capturing precedence relationships with multiple parent nodes (middle), and *linear sequence clusters* that group similar event sequences to reveal common patterns (right). We elaborate the chart generation and ground truth curation process in app. A.4.

2.2 Task Design

Visual Question Answering (VQA). We manually authored and validated 144 questions, each yielding either a single numeric answer or being labeled as *unanswerable*. Two domain experts independently created and then cross-validated the questions based on the visualizations. A third expert conducted final validation of the agreed-upon questions. Questions without consensus were excluded. Each question was verified to be answerable using only the visualization e.g., no external world knowledge is necessary. The questions fall into four categories (tab. A.3) tailored to process visualizations: (1) *value extraction*: retrieve a specific value from a node or edge, (2) *sequential reasoning*: identify an antecedent–sequela relationship, (3) *value aggregation*: sum values across multiple, possibly distant, elements, and (4) *unanswerable*: detect that the queried information is absent.

The first two categories involve simply locating chart elements and are classified as *single-hop*. In contrast, the latter two demand synthesizing information from multiple elements, and we therefore designate them as *multi-hop* reasoning tasks. Examples of VQA questions for each visualization class are shown in figs. D.1 to D.3.

Visual Data Extraction (VDE). In this task, the model is asked to output all the tuples: for linear sequence clusters, the tuples are in the form of node–value pairs, for tree and graph structures, the tuples are in the form of source–target–value triples. Overall, the corpus contains $2,583$ ground truth tuples across over 118 visualizations.

Structural Density Metric: For each visualization type, we compute the median (m) and standard deviation (σ) of node and edge counts and assign every chart to a density tier: *low* ($\leq m$), *medium* ($m, m+\sigma$), or *high* ($> m+\sigma$). An overall density score is obtained by averaging the node- and edge-based tiers. Tab. A.1 provides summary statistics and tab. A.2 provides structural density ranges for each tier across visualization types.

Experiment Setting: Tab. A.3 summarizes the question distribution across the four VQA categories and the chart counts at each density level for VDE task. Every VLM (app. B.3) is tested on both tasks. For each chart type, we provide a type-specific system prompt (figs. D.1 to D.5) that describes the chart semantics (Wei et al., 2022), but supply *no* in-context examples: the evaluation is strictly zero-shot (Xian et al., 2017). Additional details and justifications of the experimental setup are provided in app. B.1.

3 Findings

We evaluated 21 VLMs on ProcVQA across visual question answering (VQA) and visual data extraction (VDE), and present the key findings below. The evaluation metrics are described in app. B.2 and full quantitative results are available in app. E.

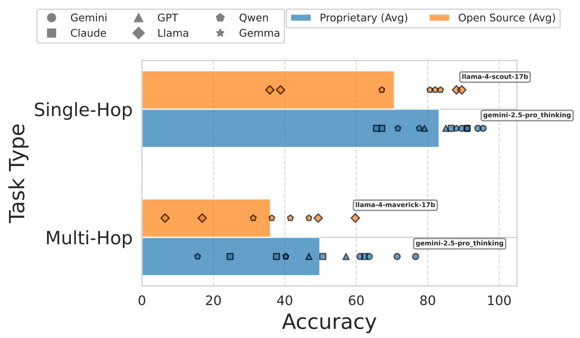


Figure 2: Comparison of model accuracy across Single-Hop vs Multi-Hop reasoning. Even top-performing models show sharp declines on multi-hop questions.

Reasoning Capability Improves Performance. Gemini-2.5-Pro-Thinking achieves the highest accuracy on both single-hop (95.52%) and multi-hop (76.62%) VQA tasks across all tested models (tab. 1). On VDE tasks, it either leads or is second-best across all levels of structural density (99.43%, 99.02%, 91.08% for low, medium, and high densities, respectively). It is also the only model with fewer than 100 hallucinations (82 total; tab. E.1). Specifically, for multi-hop reasoning tasks, it outperforms its non-thinking variant, Gemini-2.5-Pro (63.64%), by almost 13%. This indicates that the model's "thinking" capability, supporting extended reasoning chains, provides an advantage over the other models.

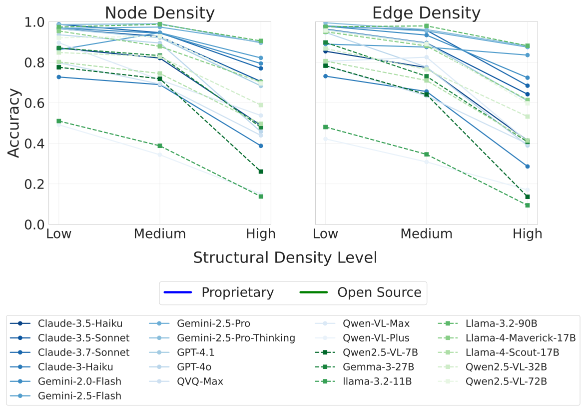


Figure 3: Accuracy on Visual Data Extraction (VDE) task across structural density levels (low, medium, high). While models generally achieve high accuracy on low-density charts, performance declines as density increases.

Smaller Models Extract Fewer and Less Accurate Data for VDE. Within the same model family, smaller models extract fewer data elements (tab. E.1). For example, Llama-3.2-11B-Vision extracted $1,285$ fewer tuples than Llama-3.2-90B-Vision. In contrast, larger models achieve near-complete coverage, such as Gemini-2.5-Pro-Thinking (98.3%), Llama-3.2-90B-Vision (99.1%), and Gemini-2.5-Pro (98.2%).

The extraction gap widens for high-density visualizations: smaller models like Qwen2.5-VL-7B and Llama-3.2-11B-Vision extracts only 416 tuples (39.7% coverage) and 310 (29.6%) tuples respectively (tab. E.2).

Smaller models also show weaker reasoning accuracy. For example, Claude-3.5-Sonnet outperforms Claude-3.5-Haiku by 21% on single-hop and 13% on multi-hop tasks.

Hallucination Increases with Structural Density for VDE. As structural density increases, most models show a corresponding increase in hallucination rates, with edge density generally having a stronger effect than node density (tabs. E.2

	Model	VQA Task (Acc. %)		VDE Task (Acc. %)		
		Single-Hop	Multi-Hop	Low	Medium	High
Proprietary	Gemini-2.5-Pro-Thinking (Team et al., 2023)	**95.52**	**76.62**	**99.43**	**99.02**	**91.08**
	Gemini-2.5-Pro (Team et al., 2023)	**94.03**	63.64	98.60	**97.00**	**91.15**
	Gemini-2.5-Flash (Team et al., 2023)	91.04	**71.43**	88.98	93.21	82.79
	Gemini-2.0-Flash (Team et al., 2023)	89.55	61.04	97.61	96.37	83.37
	Claude-3.7-Sonnet (Anthropic, 2024)	91.04	62.34	**98.93**	96.77	82.77
	Claude-3.5-Sonnet (Anthropic, 2024)	86.57	50.65	97.44	95.63	78.41
	Claude-3.5-Haiku (Anthropic, 2024)	65.67	37.66	88.40	87.71	58.65
	Claude-3-Haiku (Anthropic, 2024)	67.16	24.68	78.41	78.35	48.80
	GPT-4.1 (Achiam et al., 2023)	85.07	57.14	96.56	93.69	74.08
	GPT-4o (Achiam et al., 2023)	79.10	46.75	95.13	91.72	58.31
	Qwen-VL-Max (Bai et al., 2023)	88.06	40.26	93.46	83.45	63.90
	Qwen-VL-Plus (Bai et al., 2023)	71.64	15.58	64.77	46.72	23.00
	QVQ-Max (Bai et al., 2023)	77.61	40.26	84.71	74.09	54.96
Open Source	Llama-4-Maverick-17B (Meta, 2025)	**88.06**	**59.74**	95.41	89.22	**75.72**
	Llama-4-Scout-17B (Meta, 2025)	**89.55**	**49.35**	83.43	82.01	58.26
	Llama-3.2-90B-Vision (Grattafiori et al., 2024)	35.82	16.88	**96.89**	**99.24**	**91.78**
	Llama-3.2-11B-Vision (Grattafiori et al., 2024)	38.81	6.49	59.41	52.49	21.21
	Qwen2.5-VL-72B (Bai et al., 2023)	80.60	46.75	**95.45**	**93.35**	73.79
	Qwen2.5-VL-32B (Bai et al., 2023)	82.09	41.56	89.06	87.24	66.31
	Qwen2.5-VL-7B (Bai et al., 2023)	67.16	31.17	82.14	77.76	37.30
	Gemma-3-27B-IT (Team et al., 2025)	83.58	36.36	92.50	86.72	58.82

Table 1: Model performance comparison across two tasks in ProcVQA: VQA (Single-Hop vs. Multi-Hop accuracy) and VDE (Accuracy by total (node + link) structural complexity levels: Low, Medium, High). Models are classified into proprietary and open-source categories. **bold underlined** (best), **bold** (second-best).

and E.3).

Even frontier models exhibit this pattern. For Gemini-2.5-Pro-Thinking, hallucinations increase 18.5× from low to high node density (0.4% → 7.4%) and 33.6× for edge density (0.3% → 10.1%). Claude-3.7-Sonnet shows even larger jumps (26.3× for nodes, 51.7× for edges), while the increment for GPT-4.1 is 6.4× and 9.3×, respectively. These results indicate that edges are the primary source of difficulty.

For several models, high-density visualizations trigger extreme hallucination: Llama-3.2-11B-Vision hallucinates on 53.5% of high-node and 68.1% of high-edge visualizations, while Qwen-VL-Plus exceeds 50% in both settings. We provide further insight into hallucination modes, such as fusing multiple nodes in app. G.2.

Visual Structure affects VDE Performance. VLM performance on VDE tasks varies substantially across visualization structures (tab. E.5).

Graphs emerge as the most challenging visualization type for most models, often yielding lowest precision and recall across visual structures. For instance, Gemini-2.5-Pro-Thinking shows 8% accuracy drop for graphs compared to trees and linear sequences, while precision loss for both GPT models (4o, 4.1) is >17% . Similar trends appear across most model families.

In contrast, models achieve high precision on linear sequence clusters despite their larger average node count (40.53 compared to 11.88 for trees and 13.17 for graphs; tab. A.1). This counter-intuitive finding suggests that the structural simplicity (linear, non-branching chains) may be a stronger predictor of VDE performance than raw element count. In other words, edges in branching structures appear to be a weak point for VLMs.

Impact of Visual Structures Varies across VQA Tasks. As shown in fig. 2 and tab. 1, model performance drops 19–56% when moving from single-hop to multi-hop reasoning, with the extent of drop depending strongly on the underlying visualization structure.

Models consistently struggle with *unanswerable* questions on linear sequence clusters. Seven models fail completely (0% accuracy; tab. E.7), typically defaulting to spurious numeric guesses instead of outputting ‘indeterminate’ (app. G.3). Even Gemini-2.5-Pro-Thinking, one of the strongest models overall, achieves only 60% accuracy, compared to ≥88.89% on trees and graphs. Gemini-2.5-Flash stands out by reaching perfect accuracy across all structures.

Value aggregation on graphs is another challenging task-structure combination: all non-Gemini models score below 20% accuracy, while Gemini-2.5-Pro-Thinking peaks at just 38.46%. This task requires tracing and summing values across multiple, distant nodes in non-linear structures with complex parent–child relationships. We analyze common failure modes in app. G.1, with detailed categorization (tab. G.1) and examples (tabs. G.2 to G.7). In contrast, *value extraction* tasks are consistently easy across all structures: six models reach perfect (100%) accuracy (tab. E.6), indicating that locating and reading numeric values is well within current VLM capabilities.

Trees, in general, emerge as the most tractable structure: 12 models, including mid-sized Gemma-3-27B-IT, achieve perfect retrieval accuracy (tab. E.7). Sequential reasoning in trees is also well-handled, with eight models (including all Gemini variants) reaching perfect accuracy. In contrast, no model achieves perfect accuracy for *sequential reasoning* in graphs or linear sequence clusters.

Parallels and Divergences between Human and Machine Capabilities. To contextualize our findings, we compare VLM performance with a human baseline, based on prescreening test results on 301 participants from Zinat et al. (2023), which uses the same visualization generation pipeline as our benchmark (app. F). Notably, only 63.8% passed the visual literacy prescreening test, highlighting that process visualizations present significant cognitive challenges even for humans. Comparing task-specific performance reveals that both humans and VLMs find *value extraction* to be the most accessible task, and *value aggregation* to be the most challenging (tab. F.1). In *sequential reasoning* tasks, the best-performing model substantially outperforms humans.

Performance Divergence on VQA and VDE for Llama-3.2-90B-Vision. Our evaluation reveals a performance divergence that challenges assumptions about the relationship between data extraction and reasoning capabilities. Models capable of accurately identifying and extracting visual elements and their relationships should inherently perform well when answering questions that require reasoning over those same elements. However, our findings demonstrate that these capabilities can be surprisingly decoupled in certain models.

Llama-3.2-90B-Vision stands out for its resilience to increasing density, maintaining high VDE accuracy across all levels (low: 96.89%, medium: 99.24%, high: 91.78%; tab. 1). Although hallucination rates rise with density (7.0% at high node, 9.6% at high edge), the growth is modest (3.3× and 5.3×) compared to other models, with only 106 hallucinations overall (tab. E.1). Notably, it is the only model to achieve ≥90% precision and recall consistently across all structures (tab. E.5).

However, the same model performs poorly on VQA tasks, reaching just 35.82% accuracy for single-hop, and 16.88% for multi-hop reasoning (tab. 1). This gap between VDE and VQA is unique among the models we studied.

Upon closer inspection, we found part of the performance drop could be attributed to instruction-following failures. For instance, LLaMA-3.2-11B-Vision achieved near-zero accuracy on single-hop (0%) and multi-hop (1.3%) tasks, because it consistently failed to comply with the specified output format requirements. It repeatedly produced multiple <answer> tags instead of the required format <answer>X</answer>. After manually extracting answers, its adjusted accuracy rose to 21.53%, still far below other models but notably higher than the near-zero automated scores. Additional cases of instruction nonconformance are detailed in app. G.3.

4 Conclusion

ProcVQA is the first systematic benchmark for evaluating Vision-Language Models on mined process visualizations. A central contribution of our work is the analysis of the influence of structural density, structure type, and task category, which uncovers degradation patterns obscured in traditional benchmarks. We find that edge density has a stronger effect on performance than node density. Graphs are the most challenging structure type for most models, while trees are the most tractable. *Value extraction* task is consistently manageable across all structures, while all models perform poorly on *value aggregation* task on graphs These findings demonstrate that VLM performance depends on the interplay between structural density, structure type, and reasoning complexity.

We also observe that strong extraction capability may not translate to strong reasoning capability, and a wide variation in instruction-following ability, underscoring the need to consider extraction power, reasoning ability, and instruction adherence when deploying models (app. G.3).

Future work should focus on improving multi-hop reasoning, where models must perform a series of operations before reaching a solution. Tasks such as tracing values across multiple paths and aggregating them (app. G.1) remain challenging even for frontier models.

5 Limitations

While we have conducted evaluations across three different visualization types across eight domains, our benchmark should be considered as complementary to existing chart benchmarks (Singh et al., 2024; Zhang et al., 2024), rather than a comprehensive standalone evaluation (app. C).

It is important to note that the specific datasets and questions in the human study (app. F) differ from our benchmark despite using the same visualization generation pipeline. Nevertheless, these relative performance patterns across task types provide valuable insights into the comparative strengths and weaknesses of how human and machines interpret process visualizations.

Our standardized visual style eliminates confounding stylistic variation, but may not reflect performance on charts with diverse visual encodings found in real-world settings. This standardization was a deliberate methodological choice to isolate the impact of structural complexity on model performance. Without established metrics to quantify or control for complexity introduced by visual diversity, this approach provides a clean experimental setting to measure VLM performance strictly as a function of structural complexity.

A key challenge is extending these findings beyond process visualizations requires robust ways of quantifying structural complexity across chart types. We call upon the visualization community to develop such systematic measures that enable more generalizable evaluation.

References

Sequence summary repository. `https://github.com/hdi-umd/SequenceSummary`. Accessed: 2025-09-16.

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Tony Allard, Paul Alvino, Leslie Shing, Allan Wollaber, and Joseph Yuen. 2019. A dataset to facilitate automated workflow analysis. *PLOS ONE*, 14(2):1–22.

Anthropic. 2024. Claude 3 model family: Opus, sonnet, haiku. `https://www.anthropic.com/`. Released March 2024.

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*, 2.

Elizabeth A. Carter, Randall S. Burd, Megan Monroe, Catherine Plaisant, and Ben Shneiderman. 2013. Using eventflow to analyze task performance during trauma resuscitation.

Yuanzhe Chen, Panpan Xu, and Liu Ren. 2017. Sequence synopsis: Optimize visual summary of temporal event data. *IEEE transactions on visualization and computer graphics*, 24(1):45–55.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

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*.

Mengdie Hu, Krist Wongsuphasawat, and John Stasko. 2016. Visualizing social media content with sententree. *IEEE transactions on visualization and computer graphics*, 23(1):621–630.

Robert A Jacobs, Michael I Jordan, Steven J Nowlan, and Geoffrey E Hinton. 1991. Adaptive mixtures of local experts. *Neural computation*, 3(1):79–87.

Shankar Kantharaj, Xuan Long Do, Rixie Tiffany Leong, Jia Qing Tan, Enamul Hoque, and Shafiq Joty. 2022a. Opencqa: Open-ended question answering with charts. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 11817–11837.

Shankar Kantharaj, Rixie Tiffany Leong, Xiang Lin, Ahmed Masry, Megh Thakkar, Enamul Hoque, and Shafiq Joty. 2022b. Chart-to-text: A large-scale benchmark for chart summarization. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4005–4023, Dublin, Ireland. Association for Computational Linguistics.

Bum Chul Kwon, Vibha Anand, Kristen A. Severson, Soumya Ghosh, Zhaonan Sun, Brigitte I. Frohnert, Markus Lundgren, and Kenney Ng. 2021. Dpvis: Visual analytics with hidden markov models for disease progression pathways. *IEEE Transactions on Visualization and Computer Graphics*, 27(9):3685–3700.

Fuxiao Liu, Xiaoyang Wang, Wenlin Yao, Jianshu Chen, Kaiqiang Song, Sangwoo Cho, Yaser Yacoob, and Dong Yu. 2024. 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 (Volume 1: Long Papers)*, pages 1287–1310, Mexico

City, Mexico. Association for Computational Linguistics.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. 2023a. Visual instruction tuning. *Advances in neural information processing systems*, 36:34892–34916.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2023b. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM computing surveys*, 55(9):1–35.

Zhicheng Liu, Bernard Kerr, Mira Dontcheva, Justin Grover, Matthew Hoffman, and Alan Wilson. 2017. Coreflow: Extracting and visualizing branching patterns from event sequences. In *Computer Graphics Forum*, volume 36, pages 527–538. Wiley Online Library.

Jessica Magallanes, Tony Stone, Paul D Morris, Suzanne Mason, Steven Wood, and Maria-Cruz Villa-Uriol. 2022. Sequen-c: A multilevel overview of temporal event sequences. *IEEE Transactions on Visualization and Computer Graphics*, 28(1):901–911.

Ahmed Masry, Mohammed Saidul Islam, Mahir Ahmed, Aayush Bajaj, Firoz Kabir, Aaryaman Kartha, Md Tahmid Rahman Laskar, Mizanur Rahman, Shadikur Rahman, Mehrad Shahmohammadi, Megh Thakkar, Md Rizwan Parvez, Enamul Hoque, and Shafiq Joty. 2025. Chartqapro: A more diverse and challenging benchmark for chart question answering. *Preprint*, arXiv:2504.05506.

Ahmed Masry, Parsa Kavehzadeh, Xuan Long Do, Enamul Hoque, and Shafiq Joty. 2023. UniChart: A universal vision-language pretrained model for chart comprehension and reasoning. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 14662–14684, Singapore. Association for Computational Linguistics.

Ahmed Masry, Do Xuan Long, Jia Qing Tan, Shafiq Joty, and Enamul Hoque. 2022. Chartqa: A benchmark for question answering about charts with visual and logical reasoning. *arXiv preprint arXiv:2203.10244*.

AI Meta. 2025. The llama 4 herd: The beginning of a new era of natively multimodal ai innovation. *https://ai. meta. com/blog/llama-4-multimodal-intelligence/, checked on*, 4(7):2025.

Nitesh Methani, Pritha Ganguly, Mitesh M Khapra, and Pratyush Kumar. 2020. Plotqa: Reasoning over scientific plots. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1527–1536.

Megan Monroe. Basketball play-by-play analysis using eventflow.

OWID. 2025. Our world in data. https://ourworldindata.org/. Accessed: 2025-09-15.

Huitong Pan, Qi Zhang, Cornelia Caragea, Eduard Dragut, and Longin Jan Latecki. 2024. Flowlearn: Evaluating large vision-language models on flowchart understanding. *arXiv preprint arXiv:2407.05183*.

Pew Research Center. 2025. Pew research center. https://www.pewresearch.org/. Accessed: 2025-09-15.

Catherine Plaisant. Eventflow demo (short for talks) - visual analytics for temporal event data.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*.

Shubhankar Singh, Purvi Chaurasia, Yerram Varun, Pranshu Pandya, Vatsal Gupta, Vivek Gupta, and Dan Roth. 2024. Flowvqa: Mapping multimodal logic in visual question answering with flowcharts. *arXiv preprint arXiv:2406.19237*.

Tableau Public. 2025. Tableau public. https://public.tableau.com/. Accessed: 2025-09-15.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, and 1 others. 2023. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*.

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, and 1 others. 2025. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*.

Zirui Wang, Mengzhou Xia, Luxi He, Howard Chen, Yitao Liu, Richard Zhu, Kaiqu Liang, Xindi Wu, Haotian Liu, Sadhika Malladi, and 1 others. 2024. Charxiv: Charting gaps in realistic chart understanding in multimodal llms. *Advances in Neural Information Processing Systems*, 37:113569–113697.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Mark A. Whiting, Kris Cook, R. Jordan Crouser, John Fallon, Georges Grinstein, Jereme Haack, Cindy Henderson, Kristen Liggett, Diane Staheli, Jana Strasburg, Jerry Tagestad, and Carrie Varley. 2017. Vast challenge 2017: Mystery at the wildlife preserve. In *2017 IEEE Conference on Visual Analytics Science and Technology (VAST)*, pages 173–178.

Renqiu Xia, Bo Zhang, Hancheng Ye, Xiangchao Yan, Qi Liu, Hongbin Zhou, Zijun Chen, Peng Ye, Min Dou, Botian Shi, and 1 others. 2024. Chartx

& chartvlm: A versatile benchmark and foundation model for complicated chart reasoning. *arXiv preprint arXiv:2402.12185*.

Yongqin Xian, Bernt Schiele, and Zeynep Akata. 2017. Zero-shot learning-the good, the bad and the ugly. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4582–4591.

Zhengzhuo Xu, Sinan Du, Yiyan Qi, Chengjin Xu, Chun Yuan, and Jian Guo. 2023. Chartbench: A benchmark for complex visual reasoning in charts. *arXiv preprint arXiv:2312.15915*.

Dingqi Yang, Daqing Zhang, Vincent W. Zheng, and Zhiyong Yu. 2015. Modeling user activity preference by leveraging user spatial temporal characteristics in lbsns. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 45(1):129–142.

Xinyu Zhang, Lingling Zhang, Yanrui Wu, Muye Huang, Wenjun Wu, Bo Li, Shaowei Wang, and Jun Liu. 2024. Diagramqg: A dataset for generating concept-focused questions from diagrams. *arXiv e-prints*, pages arXiv–2411.

Deyao Zhu, Jun Chen, Xiaoqian Shen, Xiang Li, and Mohamed Elhoseiny. 2023. Minigpt-4: Enhancing vision-language understanding with advanced large language models. *arXiv preprint arXiv:2304.10592*.

Kazi Tasnim Zinat, Jinhua Yang, Arjun Gandhi, Nistha Mitra, and Zhicheng Liu. 2023. A comparative evaluation of visual summarization techniques for event sequences. *Computer Graphics Forum*, 42(3):173–185.

A ProcVQA: Curation, Generation, Visualization Types and Mining Algorithms

A.1 Process Dataset Curation

To build a representative benchmark, we curated **eight** real-world process datasets drawn from diverse domains. We provide some examples of datasets and domains below:

- **Healthcare.** The *Pediatric Trauma* dataset (Carter et al., 2013) records the sequence of trauma-response events, while the *Emergency* dataset (Plaisant) captures patient movement through a hospital.
- **Sports.** A play-by-play event log of the *UMD vs. UNC* basketball game (Monroe) provides fine-grained temporal actions on the court.
- **Mobility.** We include the *VAST Challenge 2017* nature-preserve vehicle-movement data (Whiting et al., 2017) and a large-scale *Foursquare* check-in corpus that traces location trajectories (Yang et al., 2015).
- **Software Engineering.** A workflow dataset of bug-fix activities in an open-source project (Allard et al., 2019) represents software-development processes.

A.2 Process Mining Algorithms

In this section, we discuss the mining algorithms associated with the visualizations.

Trees

Tree-based visualizations were mined by an algorithm proposed in Liu et al., 2017, which uses a rank-divide-trim approach for extracting process patterns. Events are initially ranked using a predefined metric, such as frequency of occurrence or average index position across process entities. The top-ranked event is included in the process summary, and the sequences are partitioned into two groups based on whether they contain the top-ranked event. The sequences containing the top-ranked event are then trimmed. These operations are recursively applied to the resulting groups until either all have been processed or a predefined minimum support threshold is reached.

Graphs

Mined by algorithm proposed in Hu et al., 2016, which also uses a rank and divide approach for pattern extraction. However, instead of pruning the sub-sequences up to the first occurrence of the top-ranked event, extracts frequent patterns above the minimum support threshold in these sub-sequences. Given the same minimum support, typically mines more events and patterns than Liu et al., 2017 due to difference in extraction strategy.

Linear Sequence Clusters

Proposed by Chen et al., 2017, applies minimum description length principle to cluster sequences and identify representative sequential patterns for each cluster. Performs iterative merging to find clusters and associated patterns, while optimizing for the number of generated patterns and the edits required to obtain the original dataset from the patterns.

A.3 Visualization Types

Our benchmark covers three node–link variants of process pattern visualizations:

Trees Hierarchical structures in which each node has a single parent and zero or more children, yielding strictly branching paths (Liu et al., 2017).

Graphs Directed acyclic graphs that allow nodes to have multiple parents, capturing convergent paths and complex precedence relations (Hu et al., 2016).

Linear sequence clusters Sets of linear event sequences clustered to expose representative temporal flows and recurring motifs (Chen et al., 2017).

A.4 Visualization and Ground Truth Generation

All charts were generated with an open-source visualization pipeline that mines three distinct process-patterns (described in app. A.2) and renders them in the three above visualization formats using a uniform visual style (Zinat et al., 2023).

Our benchmark includes 118 visualizations, which is smaller compared to some general chart benchmarks. This stems primarily from the scarcity of openly available process datasets, as many organizational process logs contain sensitive information. However, our diverse domain coverage (spanning healthcare, mobility, sports, and software engineering) helps mitigate potential biases, and the open-source pipeline (under MIT License) (Seq) enables future dataset expansion as more process data becomes available.

The mined patterns are outputted in JSON format, with exact number and event information. We use these files to construct ground truths for the Visual Data Extraction (VDE) task. Standardizing color palettes, typography, and glyphs eliminates confounding stylistic variation and allows us to isolate structural density effects.

Tab. A.1 provides an overview of the count and node and edge distribution of each visualization type in the corpus.

B Evaluation Setup

B.1 Evaluation Protocol

Zero-shot Evaluation

Our experimental setup is designed to evaluate VLM performance on process visualizations in a consistent and controlled manner. As described in section 3, we adopt a strictly zero-shot evaluation approach for all models, using type-specific system prompts without any in-context examples. This design ensures fair comparisons across all 21 models we evaluate, avoids biasing results, and accommodates models that accept only a single image per query, such as Llama-3.2. The zero-shot setting also establishes a pure baseline for visualization understanding and reflects real-world scenarios where users encounter novel visualization types.

Prompt Design

For each of the three visualization types: trees, graphs, and linear sequence clusters, we developed specialized system prompts describing the semantics of each chart type (examples in app. D). These prompts provide clear instructions about the visualization without revealing task-specific content or answers.

To further support fair evaluation, our prompts incorporate chain-of-thought (CoT) reasoning. The prompts require models to *describe their thought process and calculations step-by=step*, and to include *reasoning within an XML tag <reasoning>*, thereby making explicit how they analyzed the diagram before answering. This follows established CoT prompting principles (Wei et al., 2022). Our prompt design balances simplicity with effectiveness: detailed enough to enable zero-shot understanding of specialized chart types (by describing the visual channels used), while structured to elicit step-by-step reasoning.

Although prompt engineering (Liu et al., 2023b) can sometimes improve performance, our goal is not to maximize model scores but to establish a transparent baseline and to highlight the limitations of existing foundation models.

We believe that the combination of a zero-shot setting with reasoning-enabled prompts provides a solid foundation for future research and model improvements.

B.2 Task Metrics

We discuss the task metrics for VDE and VQA tasks in app. B.2.1 and app. B.2.2 respectively.

B.2.1 VDE

For the extraction task, we measure success using two metrics:

1. **Precision**: The proportion of correctly identified relationship tuples among all extracted tuples:

$$\text{Precision} = \frac{|\text{Correct Tuples}|}{|\text{Total Extracted Tuples}|} \quad (1)$$

   This metric captures the model's ability to avoid hallucinating relationships not present in the visual input.

Vis Type	Count	Metric	Mean	Median	Min–Max	Std
Trees	34	Nodes	11.88	10.00	4–24	5.98
		Links	10.88	9.00	3–23	5.98
		Unique Nodes	5.74	6.00	3–10	2.11
Graphs	46	Nodes	13.17	10.50	4–36	7.72
		Links	14.63	11.00	3–46	10.51
		Unique Nodes	6.48	7.00	3–14	2.47
Linear Sequence Clusters	38	Nodes	40.53	33.00	5–106	23.54
		Links	5.76	5.00	1–25	3.74
		Unique Nodes	10.55	10.50	5–32	4.09
Overall	118	Nodes	21.61	14.00	4–106	19.49
		Links	10.69	8.00	1–46	8.43
		Unique Nodes	7.58	7.00	3–32	3.64

Table A.1: Summary statistics for each process visualization type in ProcVQA

Vis Type	Nodes Density (Count)			Nodes Density (%)		
	Low	Medium	High	Low	Medium	High
Trees	19	7	8	55.9	20.6	23.5
Graphs	23	13	10	50.0	28.3	21.7
Linear Sequence Clusters	20	11	7	52.6	28.9	18.4
Overall	62	31	25	52.5	26.3	21.2

Vis Type	Links Density (Count)			Links Density (%)		
	Low	Medium	High	Low	Medium	High
Tree-based Structures	19	7	8	55.9	20.6	23.5
Graph-based Networks	25	13	8	54.3	28.3	17.4
Linear Sequence Clusters	22	13	3	57.9	34.2	7.9
Overall	66	33	19	55.9	28.0	16.1

Table A.2: Distribution of node and link density tiers (low, medium, high) across visualization types.

Task	Type	Categories	Size	Total
VQA	Single	Value Extraction	16	144
		Sequential Reasoning	51	
	Multi	Value Aggregation	54	
		Unanswerable	23	
Task	**Component**	**Density**	**#Chart**	**Total**
VDE	Node	Low	62	118
		Medium	31	
		High	25	
	Link	Low	66	
		Medium	33	
		High	19	
	Overall	Low	56	
		Medium	37	
		High	25	

Table A.3: Two evaluation tasks in ProcVQA: VQA tasks are divided by reasoning complexity into single-hop and multi-hop categories. VDE tasks are classified by structural density (Low-Medium-High) based on node count, link count, and overall structural density

2. **Recall**: The proportion of ground truth relationship tuples successfully extracted:

$$\text{Recall} = \frac{|\text{Correct Tuples}|}{|\text{Ground Truth Tuples}|} \tag{2}$$

This metric measures completeness of cross-modal transfer—how effectively the model captures all relevant visual information in its linguistic output. Recall is presented as accuracy in main paper.

3. **Hallucination Rate**: The proportion of extracted events that are hallucinated:

$$\text{Hallucination Rate} = \frac{|\text{Hallucinated Tuples}|}{|\text{Total Extracted Tuples}|} \times 100$$
$$= 1 - P$$

where *Extracted* is the total number of tuples produced by the model and *Hallucinated* is the number of spurious tuples among them.

We further analyze these metrics across visualization types and density levels to identify specific visual structures that challenge model performance.

B.2.2 VQA

For the VQA task, we evaluate accuracy overall and across task types:

1. **Overall Accuracy**: The proportion of questions answered correctly across all question types:

$$\text{Accuracy} = \frac{|\text{Correct Answers}|}{|\text{Total Questions}|} \quad (3)$$

2. **Category-Specific Accuracy**: Performance broken down by question category (value extraction, sequential reasoning, value aggregation, negative cases)

B.3 Model Selection

For evaluation across a comprehensive spectrum of current Vision-Language Models (VLMs), we selected a set of 21 models representing different architectures, parameter scales, and training paradigms. Our selection includes both closed-source frontier models and open-source alternatives.

B.3.1 Closed-Source Models

We evaluated 12 proprietary models across four model families.

- **OpenAI**: GPT-4.1, GPT-4o
- **Anthropic**: Claude-3.7-Sonnet, Claude-3.5-Sonnet, Claude-3.0-Haiku, and Claude-3.5-Haiku
- **Google**: Gemini-2.5-Pro-Thinking, Gemini-2.5-Pro, Gemini-2.5-Flash , and Gemini-2.0-Flash
- **Qwen**: Qwen-VL-Max, Qwen-VL-Plus and QVQ-Max

B.3.2 Open-Source Models

We evaluated 9 open-source VLMs of varying sizes and architectures across three model families:

- **LLaMA**: LLama-4-Maverick-17B, LLama-4-Scout-17B, LLama3.2-90B-Vision, and LLama-3.2-11B-Vision
- **Qwen (Open-Source)**: Qwen2.5-VL-72B, Qwen2.5-VL-32B, and Qwen2.5-VL-7B
- **Google (Open-Source)**: Gemma3-27B-IT

C ProcVQA in the Context of Previous Benchmarks

Several prior datasets target visual reasoning with flowcharts and related diagram types, including FlowLearn (Pan et al., 2024), FlowVQA (Singh et al., 2024), and DiagramQG (Zhang et al., 2024). Flowcharts and process visualizations, though visually similar, serve fundamentally different purposes. Therefore, ProcVQA differ from the setting of prevoious work in several key ways.

Quantitative Edge Semantics. ProcVQA focuses on process visualizations mined from real-world event sequences, where edges carry quantitative weights indicating transition frequencies. In contrast, flowcharts in FlowLearn (Pan et al., 2024) and FlowVQA(Singh et al., 2024) represent prescriptive paths or binary decisions (e.g., yes/no).

Structural Differences. Process visualizations often contain unique structures such as linear sequence clusters (fig. 1, right), which are not captured by traditional flowchart datasets. ProcVQA incorporates these structures.

Aggregation Reasoning. Our benchmark introduces multi-hop questions that require aggregating values across multiple paths while preserving chronological order. This is distinct from flowchart reasoning, which primarily involves tracing logical paths.

Deterministic Evaluation. Unlike DiagramQG (Zhang et al., 2024), which relies on open-ended questions evaluated with similarity metrics (e.g., BLEU), ProcVQA emphasizes deterministic answers (numeric or "indeterminate"), allowing exact accuracy computation.

Controlled Visual Design. Finally, ProcVQA minimizes stylistic variation (color schemes, fonts, glyphs), enabling a more principled evaluation of structural density in VLMs. This controlled design isolates identifying performance challenges due to structural density that may otherwise be confounded by visual diversity.

Overall, ProcVQA complements existing flowchart-based benchmarks by addressing a distinct dimension of visual reasoning: understanding quantitative process patterns derived from real-world event data.

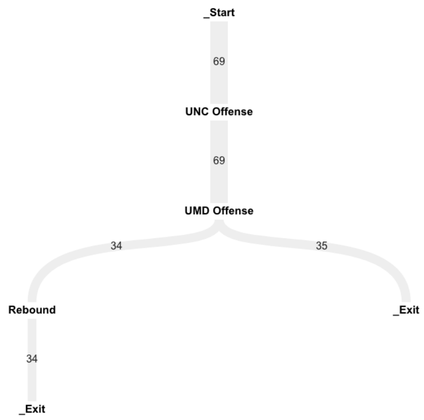
D Prompt Templates for VQA & VDE Tasks

Examples of the VQA task on the tree, graph and linear sequences diagram are provided in figs. D.1 to D.3 respectively. Example of the VDE task on trees and graphs (extracting source node, target node and count) is provided in fig. D.4 and on linear structures (extracting node and count) is provided in fig. D.5.

E Results

We describe the VQA and VDE Task Results here.

Example 1: Visual Question Answering on Trees



System Prompt

```
You are a good assistant who can retrieve information and provide answers from node-link diagram visualizations.

## Context You are presented with a node-link diagram visualization that shows sequences of events:
- Each event is represented by a node with a label
- Events are connected by directed links forming pathways
- The vertical position of nodes indicates chronological order (top to bottom)
- Link widths are proportional to the number of sequences following that path
- Numeric labels next to each link show the exact count of sequences

## Instructions
- Carefully examine all nodes, links, labels, and their corresponding values
- Track complete pathways through the diagram to understand event sequences
- Answer **solely** based on what is explicitly shown in the visualization
- If the answer **cannot be determined** from the visualization, respond with 'indeterminate'

## Response Format
1. Identify the relevant elements in the visualization that inform your answer
2. Describe your thought process and calculations clearly step-by-step in XML tag <reasoning>, showing how you analyzed
   the diagram
3. After your detailed explanation, provide your final answer in the following format: <answer>X</answer> where X is your
   direct answer or 'indeterminate'
```

Value Extraction (Single-hop)

User Prompt: *Analyze the node-link diagram shown in the figure and answer the following question.*
Question
How many Rebounds were made by UMD Offense?

Sequential Reasoning (Single-hop)

User Prompt: *Analyze the node-link diagram shown in the figure and answer the following question.*
Question
How many times did UMD Offense transition from UNC Offense?

Unanswerable (Multi-hop)

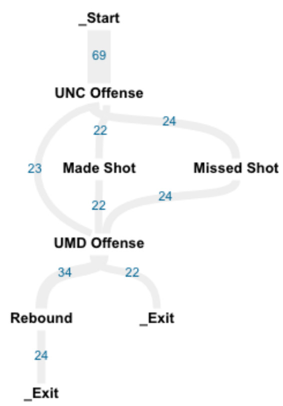
User Prompt: *Analyze the node-link diagram shown in the figure and answer the following question.*
Question
How many shots were missed immediately after UMD Offense?

Value Aggregation (Multi-hop)

User Prompt: *Analyze the node-link diagram shown in the figure and answer the following question.*
Question
How many times did UMD Offense not transition to a Rebound?

Figure D.1: Example of four types of VQA on Trees

Example 2: Visual Question Answering on (b) Graphs



System Prompt

```
You are a good assistant who can retrieve information and provide answers from node-link diagram visualizations.

## Context You are presented with a node-link diagram visualization that shows sequences of events:
- Each event is represented by a node with a label
- Events are connected by directed links forming pathways
- The vertical position of nodes indicates chronological order (top to bottom)
- Link widths are proportional to the number of sequences following that path
- Numeric labels next to each link show the exact count of sequences

## Instructions
- Carefully examine all nodes, links, labels, and their corresponding values
- Track complete pathways through the diagram to understand event sequences
- Answer **solely** based on what is explicitly shown in the visualization
- If the answer **cannot be determined** from the visualization, respond with 'indeterminate'

## Response Format
1. Identify the relevant elements in the visualization that inform your answer
2. Describe your thought process and calculations clearly step-by-step in XML tag <reasoning>, showing how you analyzed
   the diagram
3. After your detailed explanation, provide your final answer in the following format: <answer>X</answer> where X is your
   direct answer or 'indeterminate'
```

Value Extraction (Single-hop)

User Prompt: *Analyze the node-link diagram shown in the figure and answer the following question.*
Question
How many total rebounds are there in the match?

Sequential Reasoning (Single-hop)

User Prompt: *Analyze the node-link diagram shown in the figure and answer the following question.*
Question
How many made shots were after UNC Offense but before UMD offense?

Unanswerable (Multi-hop)

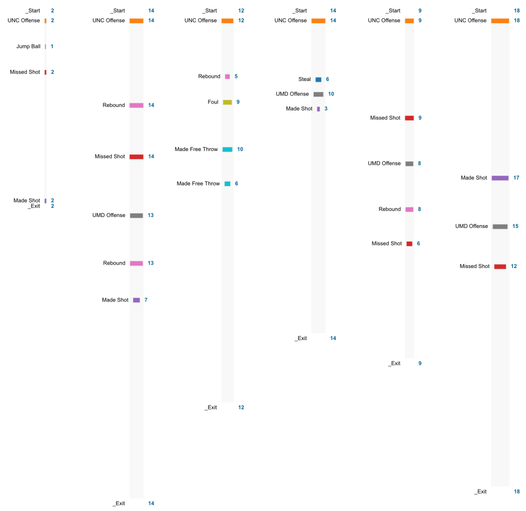
User Prompt: *Analyze the node-link diagram shown in the figure and answer the following question.*
Question
How many fouls were made after UNC Offense and before UMD offense?

Value Aggregation (Multi-hop)

User Prompt: *Analyze the node-link diagram shown in the figure and answer the following question.*
Question
How many times did UMD Offense transition from UNC Offense?

Figure D.2: Example of four types of VQA on Graphs

Example 3: Visual Question Answering on (c) Linear Sequence Clusters



System Prompt

```
You are a good assistant who can retrieve information and provide answers from node-link diagram visualizations.

## Context You are presented with a node-link diagram visualization that shows sequences of events:
- Each event is represented by a node with a label
- Events are connected by directed links forming pathways
- The vertical position of nodes indicates chronological order (top to bottom)
- Link widths are proportional to the number of sequences following that path
- Numeric labels next to each node show the exact count of sequences

## Instructions
- Carefully examine all nodes, links, labels, and their corresponding values
- Track complete pathways through the diagram to understand event sequences
- Answer **solely** based on what is explicitly shown in the visualization
- If the answer **cannot be determined** from the visualization, respond with 'indeterminate'

## Response Format
1. Identify the relevant elements in the visualization that inform your answer
2. Describe your thought process and calculations clearly step-by-step in XML tag <reasoning>, showing how you analyzed
   the diagram
3. After your detailed explanation, provide your final answer in the following format: <answer>X</answer> where X is your
   direct answer or 'indeterminate'
```

Sequential Reasoning (Single-hop)

User Prompt: *Analyze the node-link diagram shown in the figure and answer the following question.*
Question
How many times did a jump ball occur after UNC Offense?

Unanswerable (Multi-hop)

User Prompt: *Analyze the node-link diagram shown in the figure and answer the following question.*
Question
How many free throws were made by UMD Offense?

Value Aggregation (Multi-hop)

User Prompt: *Analyze the node-link diagram shown in the figure and answer the following question.*
Question
How many total missed shots occur after UNC Offense but before UMD Offense?

Figure D.3: Example of four types of VQA on Linear Sequence Clusters

Example 4: Visual Data Extraction on Trees and Graphs

System Prompt

```
You are a good assistant who can retrieve information and provide answers from node-link diagram visualizations.

CONTEXT:

You are presented with a node-link diagram visualization that shows sequences of events:
- Each event is represented by a node with a label
- Events are connected by directed links forming pathways
- The vertical position of nodes indicates chronological order (top to bottom)
- Link widths are proportional to the number of sequences following that path
- Numeric labels next to each link show the exact count of sequences

INSTRUCTIONS:
1. Extract ALL visible connections as triplets: [<Source>, <Target>, <Count>]
2. For each connection, identify:
- The exact source node label (<Source>)
- The exact target node label (<Target>)
- The numeric value shown on that connection (<Count>)
3. Record each duplicate [<Source>, <Target>, <Count>] triplets separately.
4. Ensure all <Count> are integers, not strings.
5. **Do not include any explanations or additional text.**
6. Answer solely based on what is explicitly shown in the visualization

EXAMPLE FORMAT:
[
[<Source>, <Target>, <Count>],
[<Source>, <Target>, <Count>],
[<Source>, <Target>, <Count>],
...
]
IMPORTANT: **Your entire response must be a parseable JSON object and nothing else.**
```

User Prompt

```
Examine the node-link diagram in the figure and extract all connections as triplets. For each connection, provide the source node label, target node label, and the count value shown on that link in the format [<Source>, <Target>, <Count>].
```

Figure D.4: Example of VDE on Trees and Graphs

Example 5: Visual Data Extraction on Linear Sequence Clusters

System Prompt

```
You are a good assistant who can retrieve information and provide answers from node-link diagram visualizations.

CONTEXT:

You are presented with a node-link diagram visualization that shows sequences of events:
- Each event is represented by a node with a label
- Events are connected by directed links forming pathways
- The vertical position of nodes indicates chronological order (top to bottom)
- Link widths are proportional to the number of sequences following that path
- Numeric labels next to each node show the exact count of sequences

INSTRUCTIONS:
1. Extract ALL visible connections as pairs: [<Source>, <Count>]
2. For each connection, identify:
- The exact node label (<Source>)
- The numeric value shown on that connection (<Count>)

3. Record each duplicate [<Source>, <Count>] pairs separately.
4. Ensure all <Count> are integers, not strings.
5. **Do not include any explanations or additional text.**
6. Answer solely based on what is explicitly shown in the visualization

EXAMPLE FORMAT:
[
[<Source>, <Count>],
[<Source>, <Count>],
[<Source>, <Count>],
...
]
IMPORTANT: **Your entire response must be a parseable JSON object and nothing else.**
```

User Prompt

```
Examine the node-link diagram in the figure and extract all connections as pairs. For each connection, provide the source node label and the count value shown next to each node in the format [<Source>, <Count>].
```

Figure D.5: Example of VDE on Linear Sequence Clusters

Benchmark	Process Chart	Expert Authored	Multi-task Eval.	Deterministic Eval.	Full-Chart Extraction	Multi-hop Questions	Hallucination Analysis	Real-world Data	Non-GPT Chart	Struct. Density
PlotQA (Methani et al., 2020)	✗	∼	✗	∼	✓	✓	✗	✓	✗	✗
ChartQA (Masry et al., 2022)	✗	∼	✗	∼	∼	✓	✗	✓	✓	✗
UniChart (Masry et al., 2023)	✗	∼	✓	∼	✓	✓	✓	✓	✓	✗
Chart-to-Text (Kantharaj et al., 2022b)	✗	✓	✗	∼	✓	✗	✓	✓	✓	✗
OpenCQA (Kantharaj et al., 2022a)	✗	∼	✗	✗	✗	✓	✓	✓	✓	✗
ChartLlama (Han et al., 2023)	✓	✗	✓	✓	✓	✗	✗	✗	✗	✗
ChartBench (Xu et al., 2023)	✗	✗	✓	∼	✗	✓	✓	∼	∼	✗
ChartX (Xia et al., 2024)	✗	∼	✓	∼	✓	✓	✗	✗	✗	✗
MMC (Liu et al., 2024)	✗	✓	✓	✓	✓	✓	✓	✓	✓	✗
CharXiv (Wang et al., 2024)	✗	✓	✓	✓	✗	✓	✓	✓	✓	✗
ChartQAPro (Masry et al., 2025)	✗	∼	✓	∼	✗	✓	∼	✓	✓	✗
ProcVQA (Ours)	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Table C.1: Comparison of chart understanding benchmarks across key evaluation dimensions.: inclusion of process visualizations, whether QA is expert vs. machine-generated, multiple evaluation tasks, deterministic answer extraction, full-chart extraction, multi-step reasoning questions, hallucination analysis, real-world data sources, non-GPT generated charts, and structural density analysis. Here, ✓ = full coverage, ∼ = partial support and ✗ = not covered. ProcVQAcovers all evaluation dimensions while focusing on process visualizations.

E.1 VDE Task results

Tab. E.1 shows overall performance for the VDE task, and tabs. E.2 to E.4 breaks down performance by density level of node, edge, and overall density respectively.

E.2 VQA Task results

Tab. E.6 provides a detailed breakdown of accuracy across four VQA categories: Value Extraction, Sequential Reasoning, Unanswerable, and Value Aggregation. Tab. E.7 further breaks down the performance by visual structures.

F Human-VLM Performance Comparison

We adopt the same visualization generation pipeline as (Zinat et al., 2023) (elaborated in app. A.4). They conduct a human evaluation study on 301 crowdsourced users on a similar dataset, with six questions to prescreen participants. Their survey questions cover three of our VQA reasoning tasks (*Value Extraction, Sequential Reasoning and Value Aggregation*). 192 out of 301 participants answered at least 50% of questions correctly, and were selected for the main study.

Tab. F.1 contains the human vs VLM performance comparisons for VQA tasks. For *value extraction* task, humans achieve 96% accuracy, compared to the best VLM performance of 100%. VLMs outperform humans on *sequential reasoning* tasks by a large margin. The best-performing model (Gemini-2.5-Pro) achieves 94.12% accuracy, compared to human performance of just 68.11%, showing a substantial 26 percentage point advantage. Both humans and VLMs find the *value aggregation task* to be most challenging (64.12% for humans vs. 74.07% for the best VLM).

G Failure Mode Analysis

To better understand the limitations of current VLMs on ProcVQA, we conduct a detailed analysis of their failure modes across both VDE and VQA tasks. Model outputs exhibit patterns of failure, including instruction noncompliance, hallucination under high structural density, spurious numeric guessing, and breakdowns in reasoning over branching structures.

G.1 Case Study: Value Aggregation Task

One of the most intriguing example of model limitations comes from a *value aggregation* task requiring models to answer: "How many times did UMD Offense transition from UNC Offense?" on graph visualizations (fig. D.2). The task required accurately identifying and summing three links (with values 22, 23, and 24) between nodes and provide the correct answer of 69 (22 + 23 + 24).

All of the evaluated VLMs failed to correctly answer the question. Manual inspection of reasoning output from the VLMs revealed six distinct failure modes when processing visual-numerical relationships. Here we discuss two main failure modes, covering 15 of the 21 failures (71%). The complete list of failure modes along with model information are provided in tab. G.1. The original responses are provided in tabs. G.2 to G.7.

Partial Summation (8 models): The most prevalent failure mode involved models successfully identifying two of the three relevant links which are indirect (having another node in-between) but consistently missing the third direct connection, typi-

Model	Extracted	Correct	P%	R%	Hallucinated
Gemini Models					
Gemini-2.5-Pro-Thinking	2538	2456	96.8	95.1	82
Gemini-2.5-Pro	2536	2432	95.9	94.2	104
Gemini-2.5-Flash	2521	2245	89.1	86.9	276
Gemini-2.0-Flash	2446	2301	94.1	89.1	145
Gemma Models					
Gemma-3-27B-IT	2063	1811	87.8	70.1	252
Claude Models					
Claude-3.7-Sonnet	2411	2293	95.1	88.8	118
Claude-3.5-Sonnet	2320	2193	94.5	84.9	127
Claude-3.5-Haiku	2118	1812	85.6	70.2	306
Claude-3-Haiku	1860	1495	80.4	57.9	365
GPT Models					
GPT-4.1	2408	2166	90.0	83.9	242
GPT-4o	2083	1886	90.5	73.0	197
Qwen Models					
Qwen-VL-Plus	1158	804	69.4	31.1	354
Qwen-VL-Max	2015	1806	89.6	69.9	209
QVQ-Max	2022	1608	79.5	62.3	414
Qwen2.5-VL-72B	2406	2148	89.3	83.2	258
Qwen2.5-VL-32B	2204	1904	86.4	73.7	300
Qwen2.5-VL-7B	1772	1422	80.2	55.1	350
Llama Models					
Llama-4-Maverick-17B	2437	2146	88.1	83.1	291
Llama-4-Scout-17B	2081	1709	82.1	66.2	372
Llama-3.2-90B-Vision	2560	2454	95.9	95.0	106
Llama-3.2-11B-Vision	1275	838	65.7	32.4	437

Table E.1: Performance of all models on the VDE (visual data extraction) task, evaluated against 2,583 ground-truth events. Larger models such as Gemini-2.5-Pro-Thinking and Llama-3.2-90B-Vision achieve near-complete extraction with low hallucination, while smaller models extract fewer correct tuples and hallucinate more.

Family	Model	Level	GT	Ext	Corr	P%	R%	Hallucinated	Hallucination Rate%
Gemini	Gemini-2.5-Pro-Thinking	L	810	802	799	99.6	98.6	3	0.4
		M	725	722	718	99.4	99.0	4	0.6
		H	1048	1014	939	92.6	89.6	75	7.4
	Gemini-2.5-Pro-Preview	L	810	798	784	98.2	96.8	14	1.8
		M	725	719	706	98.2	97.4	13	1.8
		H	1048	1019	942	92.4	89.9	77	7.6
	Gemini-2.5-Flash-Preview	L	810	771	698	90.5	86.2	73	9.5
		M	725	718	686	95.5	94.6	32	4.5
		H	1048	1032	861	83.4	82.2	171	16.6
	Gemini-2.0-Flash	L	810	794	785	98.9	96.9	9	1.1
		M	725	704	684	97.2	94.3	20	2.8
		H	1048	948	832	87.8	79.4	116	12.2
Gemma	Gemma 3 27B-IT	L	810	742	705	95.0	87.0	37	5.0
		M	725	662	604	91.2	83.3	58	8.8
		H	1048	659	502	76.2	47.9	157	23.8
Claude	Claude-3.7-Sonnet	L	810	803	800	99.6	98.8	3	0.4
		M	725	706	686	97.2	94.6	20	2.8
		H	1048	902	807	89.5	77.0	95	10.5
	Claude-3.5-Sonnet	L	810	790	782	99.0	96.5	8	1.0
		M	725	693	672	97.0	92.7	21	3.0
		H	1048	837	739	88.3	70.5	98	11.7
	Claude-3.5-Haiku	L	810	762	705	92.5	87.0	57	7.5
		M	725	658	595	90.4	82.1	63	9.6
		H	1048	698	512	73.4	48.9	186	26.6
	Claude 3 Haiku	L	810	665	589	88.6	72.7	76	11.4
		M	725	579	500	86.4	69.0	79	13.6
		H	1048	616	406	65.9	38.7	210	34.1
GPT	GPT-4.1	L	810	806	782	97.0	96.5	24	3.0
		M	725	717	668	93.2	92.1	49	6.8
		H	1048	885	716	80.9	68.3	169	19.1
	GPT-4o	L	810	787	758	96.3	93.6	29	3.7
		M	725	701	649	92.6	89.5	52	7.4
		H	1048	595	479	80.5	45.7	116	19.5
Meta/Llama	Llama-4-Maverick-17B	L	810	857	773	90.2	95.4	84	9.8
		M	725	684	637	93.1	87.9	47	6.9
		H	1048	896	736	82.1	70.2	160	17.9
	Llama-4-Scout-17B	L	810	733	649	88.5	80.1	84	11.5
		M	725	611	540	88.4	74.5	71	11.6
		H	1048	737	520	70.6	49.6	217	29.4
	Llama-3.2-90B-Vision	L	810	806	789	97.9	97.4	17	2.1
		M	725	721	716	99.3	98.8	5	0.7
		H	1048	1020	949	93.0	90.6	71	7.0
	Llama-3.2-11B-Vision	L	810	574	413	65.7	32.4	161	34.3
		M	725	391	281	71.9	38.8	110	28.1
		H	1048	310	144	46.5	13.7	166	53.5
Qwen	Qwen-VL-Plus	L	810	492	398	69.4	31.1	94	30.6
		M	725	349	249	71.3	34.3	100	28.7
		H	1048	317	157	49.5	15.0	160	50.5
	Qwen-VL-Max	L	810	751	726	96.7	89.6	25	3.3
		M	725	550	517	94.0	71.3	33	6.0
		H	1048	714	563	78.9	53.7	151	21.1
	QvQ-Max	L	810	745	646	86.7	79.8	99	13.3
		M	725	651	502	77.1	69.2	149	22.9
		H	1048	626	460	73.5	43.9	166	26.5
	Qwen-2.5-VL-72B	L	810	775	746	96.3	92.1	29	3.7
		M	725	703	673	95.7	92.8	30	4.3
		H	1048	928	729	78.6	69.6	199	21.4
	Qwen-2.5-VL-32B	L	810	734	687	93.6	84.8	47	6.4
		M	725	657	600	91.3	82.8	57	8.7
		H	1048	813	617	75.9	58.9	196	24.1
	Qwen-2.5-VL-7B	L	810	708	628	88.7	77.5	80	11.3
		M	725	648	521	80.4	71.9	127	19.6
		H	1048	416	273	65.6	26.0	143	34.4

Table E.2: Model Performance on VDE task by **Node** density level. Hallucination Rate is defined as $1 - P$, i.e., the percentage of extracted events that were not present in the ground truth. GT: Ground Truth tuples, Ext: Extracted tuples, Corr: Correctly extracted tuples. Density levels: L (Low), M (Medium), H (High). Recall and Precision remain high on Low and Medium density charts but declines sharply at High density.

Family	Model	Level	GT	Ext	Corr	P%	R%	Hallucinated	Hallucination Rate%
Gemini	Gemini-2.5-Pro-Thinking	L	968	966	963	99.7	99.5	3	0.3
		M	880	852	846	99.3	96.1	6	0.7
		H	735	720	647	89.9	88.0	73	10.1
	Gemini-2.5-Pro	L	968	957	949	99.2	98.0	8	0.8
		M	880	860	840	97.7	95.5	20	2.3
		H	735	719	643	89.4	87.5	76	10.6
	Gemini-2.5-Flash	L	968	938	861	91.8	88.9	77	8.2
		M	880	860	770	89.5	87.5	90	10.5
		H	735	723	614	84.9	83.5	109	15.1
	Gemini-2.0-Flash	L	968	956	947	99.1	97.8	9	0.9
		M	880	844	822	97.4	93.4	22	2.6
		H	735	646	532	82.4	72.4	114	17.6
Gemma	Gemma 3 27B-IT	L	968	918	869	94.7	89.8	49	5.3
		M	880	712	643	90.3	73.1	69	9.7
		H	735	433	299	69.1	40.7	134	30.9
Claude	Claude-3.7-Sonnet	L	968	950	947	99.7	97.8	3	0.3
		M	880	866	843	97.3	95.8	23	2.7
		H	735	595	503	84.5	68.4	92	15.5
	Claude-3.5-Sonnet	L	968	941	933	99.1	96.4	8	0.9
		M	880	812	788	97.0	89.5	24	3.0
		H	735	567	472	83.2	64.2	95	16.8
	Claude-3.5-Haiku	L	968	892	827	92.7	85.4	65	7.3
		M	880	758	681	89.8	77.4	77	10.2
		H	735	468	304	65.0	41.4	164	35.0
	Claude-3-Haiku	L	968	797	708	88.8	73.1	89	11.2
		M	880	668	577	86.4	65.6	91	13.6
		H	735	395	210	53.2	28.6	185	46.8
GPT	GPT-4.1	L	968	956	931	97.4	96.2	25	2.6
		M	880	862	787	91.3	89.4	75	8.7
		H	735	590	448	75.9	61.0	142	24.1
	GPT-4o	L	968	949	917	96.6	94.7	32	3.4
		M	880	749	683	91.2	77.6	66	8.8
		H	735	385	286	74.3	38.9	99	25.7
Llama	Llama-4-Maverick-17B	L	968	953	920	96.5	95.0	33	3.5
		M	880	885	775	87.6	88.1	110	12.4
		H	735	599	451	75.3	61.4	148	24.7
	Llama-4-Scout-17B	L	968	865	780	90.2	80.6	85	9.8
		M	880	714	625	87.5	71.0	89	12.5
		H	735	502	304	60.6	41.4	198	39.4
	Llama-3.2-90B-Vision	L	968	961	944	98.2	97.5	17	1.8
		M	880	869	862	99.2	98.0	7	0.8
		H	735	717	648	90.4	88.2	69	9.6
	Llama-3.2-11B-Vision	L	968	654	465	71.1	48.0	189	28.9
		M	880	405	304	75.1	34.5	101	24.9
		H	735	216	69	31.9	9.4	147	68.1
Qwen	Qwen-VL-Plus	L	968	504	408	81.0	42.1	96	19.0
		M	880	386	271	70.2	30.8	115	29.8
		H	735	268	125	46.6	17.0	143	53.4
	Qwen-VL-Max	L	968	804	779	96.9	80.5	25	3.1
		M	880	770	726	94.3	82.5	44	5.7
		H	735	441	301	68.3	41.0	140	31.7
	QvQ Max	L	968	902	758	84.0	78.3	144	16.0
		M	880	690	558	80.9	63.4	132	19.1
		H	735	430	292	67.9	39.7	138	32.1
	Qwen-2.5-VL-72B	L	968	948	926	97.7	95.7	22	2.3
		M	880	842	783	93.0	89.0	59	7.0
		H	735	616	439	71.3	59.7	177	28.7
	Qwen-2.5-VL-32B	L	968	895	840	93.9	86.8	55	6.1
		M	880	740	673	90.9	76.5	67	9.1
		H	735	569	391	68.7	53.2	178	31.3
	Qwen-2.5-VL-7B	L	968	853	758	88.9	78.3	95	11.1
		M	880	717	564	78.7	64.1	153	21.3
		H	735	202	100	49.5	13.6	102	50.5

Table E.3: Model Performance on VDE task by **Edge** density level, determined by the number of edges/clusters in the visualization. Hallucination Rate is defined as $1 - P$ (i.e., the percentage of extracted events that are not in the ground truth). GT: Ground Truth tuples, Ext: Extracted tuples, Corr: Correctly extracted tuples. Density levels: L (Low), M (Medium), H (High). Recall and Precision high on Low and Medium density charts but declines sharply at High density.

Family	Model	Level	GT	Ext	Corr	P%	R%	Hallucinated	Hallucination Rate%
Gemini	Gemini-2.5-Pro-Thinking	L	612	611	608	99.5	99.3	3	0.5
		M	923	913	909	99.6	98.5	4	0.4
		H	1048	1014	939	92.6	89.6	75	7.4
	Gemini-2.5-Pro	L	612	605	600	99.2	98.0	5	0.8
		M	923	912	890	97.6	96.4	22	2.4
		H	1048	1019	942	92.4	89.9	77	7.6
	Gemini-2.5-Flash	L	612	586	533	91.0	87.1	53	9.0
		M	923	903	851	94.2	92.2	52	5.8
		H	1048	1032	861	83.4	82.2	171	16.6
	Gemini-2.0-Flash	L	612	601	592	98.5	96.7	9	1.5
		M	923	897	877	97.8	95.0	20	2.2
		H	1048	948	832	87.8	79.4	116	12.2
Gemma	Gemma-3-27B-IT	L	612	588	555	94.4	90.7	33	5.6
		M	923	816	754	92.4	81.7	62	7.6
		H	1048	659	502	76.2	47.9	157	23.8
Claude	Claude-3.7-Sonnet	L	612	605	602	99.5	98.4	3	0.5
		M	923	904	884	97.8	95.8	20	2.2
		H	1048	902	807	89.5	77.0	95	10.5
	Claude-3.5-Sonnet	L	612	597	589	98.7	96.2	8	1.3
		M	923	886	865	97.6	93.7	21	2.4
		H	1048	837	739	88.3	70.5	98	11.7
	Claude-3.5-Haiku	L	612	578	526	91.0	85.9	52	9.0
		M	923	842	774	91.9	83.9	68	8.1
		H	1048	698	512	73.4	48.9	186	26.6
	Claude 3 Haiku	L	612	518	443	85.5	72.4	75	14.5
		M	923	726	646	89.0	70.0	80	11.0
		H	1048	616	406	65.9	38.7	210	34.1
GPT	GPT-4.1	L	612	608	589	96.9	96.2	19	3.1
		M	923	915	861	94.1	93.3	54	5.9
		H	1048	885	716	80.9	68.3	169	19.1
	GPT-4o	L	612	599	576	96.2	94.1	23	3.8
		M	923	889	831	93.5	90.0	58	6.5
		H	1048	595	479	80.5	45.7	116	19.5
Llama	Llama-3.2-90B-Vision	L	612	608	591	97.2	96.6	17	2.8
		M	923	919	914	99.5	99.0	5	0.5
		H	1048	1020	949	93.0	90.6	71	7.0
	Llama-4-Maverick-17B	L	612	608	582	95.7	95.1	26	4.3
		M	923	933	828	88.7	89.7	105	11.3
		H	1048	896	736	82.1	70.2	160	17.9
	Llama-4-Scout-17B	L	612	577	496	86.0	81.0	81	14.0
		M	923	767	693	90.4	75.1	74	9.6
		H	1048	737	520	70.6	49.6	217	29.4
	Llama-3.2-11B-Vision	L	612	482	325	67.4	53.1	157	32.6
		M	923	483	369	76.4	40.0	114	23.6
		H	1048	310	144	46.5	13.7	166	53.5
Qwen	Qwen-VL-Plus	L	612	407	330	81.1	53.9	77	18.9
		M	923	434	317	73.0	34.3	117	27.0
		H	1048	317	157	49.5	15.0	160	50.5
	Qwen-VL Max	L	612	580	557	96.0	91.0	23	4.0
		M	923	721	686	95.1	74.3	35	4.9
		H	1048	714	563	78.9	53.7	151	21.1
	QvQ Max	L	612	559	496	88.7	81.0	63	11.3
		M	923	837	652	77.9	70.6	185	22.1
		H	1048	626	460	73.5	43.9	166	26.5
	Qwen-2.5-VL-72B	L	612	597	577	96.6	94.3	20	3.4
		M	923	881	842	95.6	91.2	39	4.4
		H	1048	928	729	78.6	69.6	199	21.4
	Qwen-2.5-VL-32B	L	612	567	525	92.6	85.8	42	7.4
		M	923	824	762	92.5	82.6	62	7.5
		H	1048	813	617	75.9	58.9	196	24.1
	Qwen-2.5-VL-7B	L	612	530	469	88.5	76.6	61	11.5
		M	923	826	680	82.3	73.7	146	17.7
		H	1048	416	273	65.6	26.0	143	34.4

Table E.4: Model Performance by **Overall: Node + Edge** density level. Hallucination Rate is defined as $1 - P$ (the percentage of extracted events that are not in the ground truth). GT: Ground Truth events, Ext: Extracted events, Corr: Correctly extracted events. Density levels: L (Low), M (Medium), H (High). Recall and Precision remain high on Low and Medium density charts but declines at High density.

cally resulting in answers of 46 (22 + 24). Among proprietary models, both latest GPT and Claude models (GPT-4o; Claude-3.7-Sonnet) were susceptible to this error, along with recent open-source models (Llama-4-maverick-17B).

Single-Link Selection (7 models): Models reported only one transition value as the answer, failing to recognize multiple valid paths. Both proprietary and open-sourced models from Google (Gemini and Gemma) were affected by this. Interestingly, Gemini-2.5-Flash, which demonstrated the most sophisticated reasoning in this task by explicitly acknowledging it had "missed" an edge, ultimately failed by providing only the value of the final identified link rather than aggregating all discovered values.

G.2 Case Study: Hallucination

Two failure modes dominate in hallucination scenarios:

Edge-value fabrication: the model identifies the correct `<source-target>` pair but hallucinates the edge weight, especially when several values appear close together.
Node fusion: models merge adjacent nodes, e.g., "Rebound" and "Exit" into "Rebound Exit". This happens most often when a leaf node is near a non-leaf node.

G.3 Case Study: Instruction-Following Failures

Processing failures on high-density diagrams. VLMs frequently ignored instructions or failed to generate valid outputs on dense charts. *Claude-3.7-Sonnet* entered infinite loops, endlessly repeating a tuple until hitting the token limit. Open-source models such as *Llama-3.2-11B* and *Qwen-VL-Plus* often produced unparseable JSON or non-convergent generations in 40% of extraction cases.

As mentioned in section 3, *LLaMA-3.2-11B-Vision* showed an especially anomalous pattern, with near-zero accuracy on single-hop (0%) and multi-hop (1.3%) VQA tasks, substantially lower than other models, including its own family variants. Closer inspection revealed that its failures stemmed from instruction noncompliance rather than reasoning limitations. Despite prompts specifying the format `<answer>X</answer>`, the model frequently produced multiple `<answer>` tags per response.

After manually extracting answers, its adjusted accuracy rose to 21.53%, still well below other models but significantly higher than the automated near-zero scores. Reported results for LLaMA-3.2-11B-Vision in tabs. 1, E.6 and E.7 reflect these corrected values.

Similar behavior was observed in *Llama-3.2-90B-Vision*, which deviated from the required format (e.g., outputting `**Answer**: X` instead of `<answer>X</answer>`), resulting in performance drop in VQA compared to VDE.

By contrast, *Llama-4* variants (Maverick-17B, Scout-17B) delivered more stable performance, likely benefiting from architectural refinements (Jacobs et al., 1991; Shazeer et al., 2017) or training strategies.

We also observed systematic mislabeling of 'unanswerable' questions, where some models output '0' instead of 'indeterminate', resulting in a loss of accuracy. Since such responses would propagate as errors in downstream tasks, we treat them as incorrect.

H Computational Experiments

H.1 Model Parameters, Computational Budget, and Infrastructure

In our paper, we comprehensively evaluated a diverse set of Vision-Language Models (VLMs) as detailed in app. B.3. For proprietary models, we accessed Claude models (3-Haiku through 3.7-Sonnet), Gemini models (2.0-Flash through 2.5-Pro), GPT models (4o and 4.1), and Qwen-VL commercial variants via their respective APIs.

Our open-source model evaluation included multiple parameter scales across model families: Llama models (ranging from 11B to 90B parameters), Qwen models (7B to 72B parameters), and Gemma-3-27B. The specific parameter counts for each model are provided in tab. 1.

For computational resources, we utilized 5-6 NVIDIA A6000 Ada GPUs for inference of all open-source models. The total computational budget for our experiments was approximately 760 GPU hours. Proprietary models incurred separate API usage costs around 30 USD in total.

All experimental runs were conducted between February and April 2025, with model versions current as of that period.

H.2 Experimental Setup and Hyperparameters

Consistent with this objective of establishing baselines, we did not perform hyperparameter searches. All proprietary models (Claude, Gemini, GPT, and commercial Qwen variants) were accessed via their APIs using default inference parameters, with temperature fixed to 0 for deterministic outputs.

For Gemini-2.5-Pro Thinking, we set the thinking budget as 1024 tokens.

For open-source models, we applied the following consistent settings across all experiments:

- Sampling temperature: 0
- Top-p: default model values (unchanged)
- Maximum output token length: 1000
- Batch size: 1 (due to memory constraints for larger models)

For evaluation metrics, we measure *accuracy* for the VQA task as detailed in section 2.2 and app. B.2.2, and measure precision, recall, and F1 scores for the VDE task app. B.2.1. Metrics are computed using standardized implementations to ensure consistency across all model evaluations.

Our experimental procedure for all models followed the same workflow: (1) load the model, (2) present the visualization with appropriate task-specific instructions, (3) collect and parse the model's response, and (4) compute performance metrics against gold standard annotations. This standardized approach ensures that performance differences reflect model capabilities rather than evaluation methodology variations.

H.3 Packages

All statistical analyses and visualizations were created using Python 3.10 with NumPy (1.24.3), Pandas (2.0.1), Matplotlib (3.7.1), and Seaborn (0.12.2). For model inference, we utilized the Hugging Face Transformers library (4.35.0) to load and run the open-source vision-language models.

I Use of AI Assistants

We used GitHub Copilot for assistance with code development of our evaluation and data processing pipelines. Additionally, we used Claude and ChatGPT to help with proofreading and polishing text to improve the clarity of our writing.

I.1 License

We release the code publicly on GitHub https://github.com/kzintas/ProcVQA under the MIT license. We hope that our benchmark will serve as a valuable resource for evaluating future models.

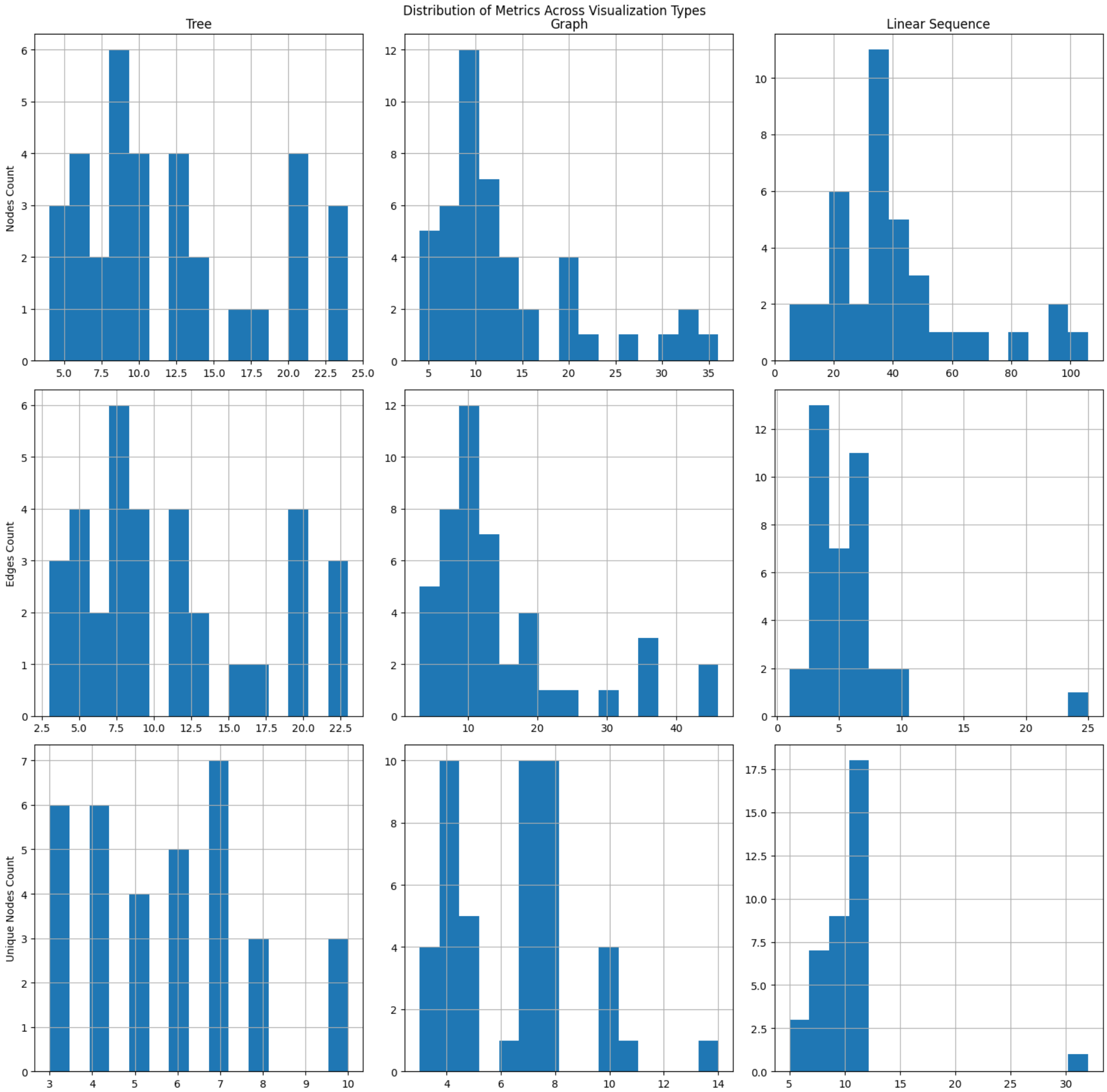


Figure E.1: Histogram of node (top), edge (middle) and unique node (bottom) distribution in ProcVQA across the three visualization structures: tree (left), graph (middle) and linear sequences (right). Linear Sequence clusters have a higher node count compared to trees and graphs.

Family	Model	Technique	Ground Truth Events	Extracted Events	Correctly Extracted	P%	R%	Hallucinated Events	Hallucination Rate%
Gemini	Gemini-2.5-Pro-Thinking	Tree	370	369	359	97.3	97.0	10	2.7
		Graph	673	659	598	90.7	88.9	61	9.3
		Linear Sequence Cluster	1540	1510	1499	99.3	97.3	11	0.7
	Gemini-2.5-Pro	Tree	370	369	360	97.6	97.3	9	2.4
		Graph	673	665	606	91.1	90.0	59	8.9
		Linear Sequence Cluster	1540	1502	1466	97.6	95.2	36	2.4
	Gemini-2.5-Flash	Tree	370	368	351	95.4	94.9	17	4.6
		Graph	673	673	580	86.2	86.2	93	13.8
		Linear Sequence Cluster	1540	1480	1314	88.8	85.3	166	11.2
	Gemini-2.0-Flash	Tree	370	364	335	92.0	90.5	29	8.0
		Graph	673	623	522	83.8	77.6	101	16.2
		Linear Sequence Cluster	1540	1459	1444	99.0	93.8	15	1.0
Gemma	Gemma-3-27B-IT	Tree	370	334	273	81.7	73.8	61	18.3
		Graph	673	570	424	74.4	63.0	146	25.6
		Linear Sequence Cluster	1540	1159	1114	96.1	72.3	45	3.9
Claude	Claude-3.7-Sonnet	Tree	370	369	349	94.6	94.3	20	5.4
		Graph	673	638	546	85.6	81.1	92	14.4
		Linear Sequence Cluster	1540	1404	1398	99.6	90.8	6	0.4
	Claude-3.5-Sonnet	Tree	370	353	335	94.9	90.5	18	5.1
		Graph	673	589	526	89.3	78.2	63	10.7
		Linear Sequence Cluster	1540	1378	1332	96.7	86.5	46	3.3
	Claude-3.5-Haiku	Tree	370	331	265	80.1	71.6	66	19.9
		Graph	673	590	414	70.2	61.5	176	29.8
		Linear Sequence Cluster	1540	1197	1133	94.7	73.6	64	5.3
	Claude-3-Haiku	Tree	370	279	166	59.5	44.9	113	40.5
		Graph	673	504	306	60.7	45.5	198	39.3
		Linear Sequence Cluster	1540	1077	1023	95.0	66.4	54	5.0
GPT	GPT-4.1	Tree	370	364	349	95.9	94.3	15	4.1
		Graph	673	650	500	76.9	74.3	150	23.1
		Linear Sequence Cluster	1540	1394	1317	94.5	85.5	77	5.5
	GPT-4o	Tree	370	346	327	94.5	88.4	19	5.5
		Graph	673	624	480	76.9	71.3	144	23.1
		Linear Sequence Cluster	1540	1113	1079	96.9	70.1	34	3.1
Llama	Llama-4-Maverick-17B	Tree	370	344	295	85.8	79.7	49	14.2
		Graph	673	608	470	77.3	69.8	138	22.7
		Linear Sequence Cluster	1540	1485	1381	93.0	89.7	104	7.0
	Llama-4-Scout-17B	Tree	370	315	198	62.9	53.5	117	37.1
		Graph	673	564	356	63.1	52.9	208	36.9
		Linear Sequence Cluster	1540	1202	1155	96.1	75.0	47	3.9
	Llama-3.2-90B-vision	Tree	370	370	360	97.3	97.3	10	2.7
		Graph	673	658	609	92.6	90.5	49	7.4
		Linear Sequence Cluster	1540	1519	1485	97.8	96.4	34	2.2
	Llama-3.2-11B-Vision	Tree	370	242	113	46.7	30.5	129	53.3
		Graph	673	425	218	51.3	32.4	207	48.7
		Linear Sequence Cluster	1540	608	507	83.4	32.9	101	16.6
Qwen	Qwen-VL-Plus	Tree	370	257	152	59.1	41.1	105	40.9
		Graph	673	460	259	56.3	38.5	201	43.7
		Linear Sequence Cluster	1540	441	393	89.1	25.5	48	10.9
	Qwen-VL-Max	Tree	370	332	296	89.2	80.0	36	10.8
		Graph	673	589	447	75.9	66.4	142	24.1
		Linear Sequence Cluster	1540	1094	1063	97.2	69.0	31	2.8
	QVQ-Max	Tree	370	319	267	83.7	72.2	52	16.3
		Graph	673	515	394	76.5	58.5	121	23.5
		Linear Sequence Cluster	1540	1188	947	79.7	61.5	241	20.3
	Qwen 2.5-VL-72B	Tree	370	343	303	88.3	81.9	40	11.7
		Graph	673	571	461	80.7	68.5	110	19.3
		Linear Sequence Cluster	1540	1492	1384	92.8	89.9	108	7.2
	Qwen 2.5-VL-32B	Tree	370	327	287	87.8	77.6	40	12.2
		Graph	673	601	408	67.9	60.6	193	32.1
		Linear Sequence Cluster	1540	1276	1209	94.7	78.5	67	5.3
	Qwen 2.5-VL-7B	Tree	370	310	215	69.4	58.1	95	30.6
		Graph	673	461	293	63.6	43.5	168	36.4
		Linear Sequence Cluster	1540	1001	914	91.3	59.4	87	8.7

Table E.5: Model performance on VDE tasks disaggregated by visualization structures: Trees, Graphs and Linear Sequence Clusters. Hallucination Rate is defined as $1 - P$, i.e., the percentage of extracted events that are hallucinated. GT: Ground Truth events, Ext: Extracted events, Corr: Correctly extracted events. The models generally perform better on trees and linear sequence clusters, showing higher precision and recall than on graphs, where the accuracy drops and hallucination rate increases.

Model	Single-Hop Reasoning		Multi-hop Reasoning	
	Value Extraction	Sequential Reasoning	Unanswerable	Value Aggregation
Gemini-2.5-Pro-Thinking	**100.00**	**94.12**	82.61	**74.07**
Gemini-2.5-Pro	**100.00**	92.16	60.87	64.81
Gemini-2.5-Flash	**100.00**	88.24	**100.00**	59.26
Gemini-2.0-Flash	93.75	88.24	60.87	61.11
Gemma-3-27B-IT	93.75	80.39	47.83	31.48
Claude-3.7-Sonnet	**100.00**	88.24	60.87	62.96
Claude-3.5-Sonnet	93.75	84.31	52.17	50.00
Claude-3.5-Haiku	62.50	66.67	30.43	40.74
Claude-3-Haiku	57.14	70.59	30.43	22.22
GPT-4.1	93.75	82.35	43.48	62.96
GPT-4o	87.50	76.47	52.17	44.44
Llama-4-Maverick-17B	**100.00**	84.31	60.87	59.26
Llama-4-Scout-17B	93.75	88.24	60.87	44.44
Llama-3.2-90B	43.75	33.33	21.74	14.81
Llama-3.2-11B	37.50	39.22	4.35	7.41
Qwen-VL-Plus	75.00	70.59	0.00	22.22
Qwen-VL-Max	**100.00**	84.31	26.09	46.30
QVQ-Max	75.00	78.43	73.91	25.93
Qwen-2.5-VL-72B	93.75	76.47	69.57	37.04
Qwen-2.5-VL-32B	93.75	78.43	47.83	38.89
Qwen-2.5-VL-7B	81.25	62.75	43.48	25.93

Table E.6: Accuracy of VLMs on four VQA categories. Results are disaggregated into single-hop (value extraction, sequential reasoning) and multi-hop (unanswerable, value aggregation) tasks.

Model Name	**Value Extraction**		**Sequential Reasoning**			**Unanswerable**			**Value Aggregation**		
	Tree	**Graph**	**Tree**	**Graph**	**Linear-sequence cluster**	**Tree**	**Graph**	**Linear-sequence cluster**	**Tree**	**Graph**	**Linear-sequence cluster**
Gemini-2.5-Pro-Thinking	100.00	100.00	100.00	90.48	92.86	88.89	88.89	60.00	100.00	38.46	80.00
Gemini-2.5-Pro	100.00	100.00	100.00	90.48	85.71	77.78	55.56	40.00	100.00	30.77	66.67
Gemini-2.5-Flash	100.00	100.00	100.00	85.71	78.57	100.00	100.00	100.00	72.73	30.77	66.67
Gemini-2.0-Flash	100.00	88.89	100.00	90.48	71.43	44.44	88.89	40.00	81.82	30.77	66.67
Gemma-3-27B-IT	100.00	88.89	93.75	85.71	57.14	44.44	77.78	0.00	45.45	15.38	33.33
Claude-3.7-Sonnet	100.00	100.00	100.00	85.71	78.57	66.67	55.56	60.00	81.82	15.38	76.67
Claude-3.5-Sonnet	100.00	88.89	100.00	80.95	71.43	55.56	66.67	20.00	63.64	15.38	60.00
Claude-3-Haiku	57.14	55.56	87.50	57.14	71.43	11.11	55.56	20.00	27.27	0.00	30.00
Claude-3.5-Haiku	57.14	66.67	87.50	61.90	50.00	44.44	11.11	40.00	45.45	7.69	53.33
GPT-4.1	100.00	88.89	100.00	66.67	85.71	66.67	44.44	0.00	90.91	15.38	73.33
GPT-4o	100.00	77.78	87.50	76.19	64.29	55.56	66.67	20.00	72.73	7.69	50.00
Llama-4-Maverick-17B	100.00	100.00	100.00	71.43	85.71	55.56	88.89	20.00	81.82	7.69	73.33
Llama-4-Scout-17B	85.71	100.00	87.50	85.71	92.86	55.56	77.78	40.00	63.64	7.69	53.33
Llama-3.2-11B-Vision	57.14	22.22	62.50	42.86	7.14	11.11	0.00	0.00	9.09	7.69	6.67
Llama-3.2-90B-Vision	14.29	66.67	50.00	28.57	21.43	0.00	55.56	0.00	18.18	7.69	16.67
Qwen-VL-plus	57.14	88.89	93.75	57.14	64.29	0.00	0.00	0.00	36.36	0.00	26.67
Qwen-VL-Max	100.00	100.00	93.75	76.19	85.71	22.22	44.44	0.00	63.64	7.69	56.67
QVQ-Max	85.71	66.67	93.75	66.67	78.57	88.89	88.89	20.00	45.45	7.69	26.67
Qwen2.5-VL-72B	85.71	100.00	93.75	80.95	50.00	77.78	88.89	20.00	54.55	15.38	40.00
Qwen2.5-VL-32B	100.00	88.89	93.75	76.19	64.29	55.56	66.67	0.00	54.55	7.69	46.67
Qwen2.5-VL-7B	85.71	77.78	87.50	42.86	64.29	22.22	66.67	40.00	54.55	7.69	23.33

Table E.7: VLM performance comparison across different tasks and visual structures. Values represent accuracy percentages. Green highlighting indicates perfect performance (100%), while red highlighting indicates zero performance where models failed completely (0%). Tree is the most tractable structure, whereas *value extraction* is the most tractable task. Most models struggle with *Unanswerable* task on Linear Sequence Clusters and *Value Aggregation* task on graphs.

Task Type	**Human Accuracy**	**Best VLM**	**Gap**
Value Extraction	96.18%	100% (Gemini 2.5 Pro, Thinking, Flash)	+3.82%
Sequential Reasoning	68.11%	94.12% (Gemini-2.5-Pro-Thinking)	+26.01%
Value Aggregation	64.12%	74.07% (Gemini-2.5-Pro-Thinking)	+9.95%

Table F.1: Human vs. best VLM accuracy across VQA tasks. Both find value extraction task easiest and value aggregation task most difficult, but VLMs surpass humans on sequential reasoning by a large margin.

Failure Mode	Description	Models Affected	Count
1. Partial Summation	Summed only two indirect edges (e.g., $22 + 24 = 46$), failing to include the third edge. This was the most frequent error.	GPT-4o, GPT-4.1, Qwen2.5-VL-72B, Qwen-VL-Max, Claude-3.5-Sonnet, Claude-3.7-Sonnet, Meta-Llama-4-Maverick-17B, Meta-Llama-4-Scout-17B	8
2. Single-Link Selection	Reported only one direct edge value (e.g., 22, 23, or 24), overlooking that multiple edges should be aggregated.	Gemini-2.0-Flash, Gemini-2.5-Pro, Gemini-2.5-Flash, Gemma-3-27B-IT, Claude-3.5-Haiku, Qwen2.5-VL-7B, Qwen2.5-VL-32B	7
3. Downstream Edge Substitution	Partially (2/3) identified some edge values but instead reported the value of an unrelated downstream edge in the graph.	Qwen-VL-Plus (Answer: 34)	1
4. Indeterminate / Could Not Find Answer	Claimed insufficient information, did not follow instructions and/or fell into a long reasoning loop without producing an answer.	Gemini-2.5-Pro-Thinking (Answer: 0) Llama-3.2-90B-Vision, QVQ-Max	3
5. Edge-Counting Error	Counted the number of edges rather than summing their values, sometimes also misreading the values themselves.	Llama-3.2-11B-Vision (Answer: 2)	1
6. Duplicate Edge Counting	Added the same edge value twice.	Claude-3-Haiku (Answer: 44)	1

Table G.1: Failure modes of VLMs on a *value aggregation task* on graphs, with the question provided in fig. D.2. We identify six different failure modes across the 21 models we evaluated.

Model	Predicted Answer	Reasoning
Gemini-2.5-Pro-Thinking	0	"<reasoning> 1. **Identify the relevant nodes:** The user is asking about the transition from the "UNC Offense" node to the "UMD Offense" node. I will locate these two nodes in the diagram. 2. **Trace the paths from the starting node:** I will start at the "UNC Offense" node and follow all the directed links originating from it. 3. **Analyze the outgoing links from "UNC Offense":** * One link goes from "UNC Offense" to "Made Shot", with a value of 22. * Another link goes from "UNC Offense" to "Missed Shot", with a value of 24. 4. **Check for direct transitions:** I will look for any direct link connecting "UNC Offense" to "UMD Offense". 5. **Observation:** There is no direct link (arrow) connecting the "UNC Offense" node directly to the "UMD Offense" node. The flow from "UNC Offense" goes through intermediate events ("Made Shot" and "Missed Shot") before reaching "UMD Offense". 6. **Conclusion:** Since there is no direct path, the number of direct transitions is 0. </reasoning> <answer>0</answer>"
Gemini-2.5-Pro	23	<reasoning> 1. **Identify the starting node:** The question asks about transitions *from* "UNC Offense". I locate the "UNC Offense" node in the diagram. 2. **Identify the ending node:** The question asks about transitions *to* "UMD Offense". I locate the "UMD Offense" node in the diagram. 3. **Look for a direct link:** I check if there is a direct link (an arrow) from the "UNC Offense" node to the "UMD Offense" node. 4. **Read the link label:** There is indeed a direct link. The numeric label next to this link indicates the number of sequences that followed this specific path. The label on the link from "UNC Offense" directly to "UMD Offense" is 23. 5. **Conclusion:** This means that there were 23 instances where "UMD Offense" immediately followed "UNC Offense". </reasoning> <answer>23</answer>
Gemini-2.5-Flash	23	<reasoning>1. Identify the relevant elements: I examined the nodes labeled "UNC Offense" and "UMD Offense" and all the links connected to them. 2. Describe the thought process: The question asks for the number of times 'UMD Offense' transitioned directly from 'UNC Offense'. This corresponds to the count on a directed link originating from the "UNC Offense" node and terminating at the "UMD Offense" node. I visually scanned the diagram for such a link. I found links going out of "UNC Offense" to "Made Shot" (22) and "Missed Shot" (24). I found links going into "UMD Offense" from "Made Shot" (22) and "Missed Shot" (24), and also a link originating from "UNC Offense" directly to "Made Shot" (23) - this seems to be an alternative path from UNC Offense to Made Shot, potentially indicating different types of 'Made Shot' events or a different interpretation of the diagram layout/links. Let me re-examine the links from UNC Offense. There are links from UNC Offense to Made Shot (labelled 22) and Missed Shot (labelled 24). There is also a curved link from UNC Offense going *around* Made Shot and connecting to UMD Offense, labelled 23. This curved link represents a direct transition from UNC Offense to UMD Offense. I initially missed this curved link. Let me re-evaluate based on this curved link. Re-evaluating: Looking carefully again, there is a curved link starting from "UNC Offense" and ending at "UMD Offense", with the numerical label "23" next to it. This link directly represents the transition from "UNC Offense" to "UMD Offense". The label "23" indicates the number of sequences following this direct path. So, the number of times UMD Offense transitioned directly from UNC Offense is 23. </reasoning> <answer>23</answer>
Gemini-2.0-Flash	22	1. The question asks about the transitions from "UNC Offense" to "UMD Offense". 2. I need to find the link that connects "UNC Offense" to "UMD Offense". 3. By examining the diagram, I can see that the link connecting "UNC Offense" to "UMD Offense" has a count of 22. <reasoning> The question asks for the number of transitions from "UNC Offense" to "UMD Offense". The diagram shows a direct link between "UNC Offense" and "UMD Offense" with the numeric label 22 next to it. This represents the number of times this transition occurred. </reasoning> <answer>22</answer>
Gemma-3-27B-IT	24	Here's an analysis of the node-link diagram to determine how many times UMD Offense transitioned from UNC Offense: <reasoning> 1. **Identify Relevant Nodes and Links:** The question asks about the transition from "UNC Offense" to "UMD Offense". We need to find the link connecting these two nodes. 2. **Examine the Link:** There is a link going from "UNC Offense" to "UMD Offense". The label on this link indicates the number of sequences following this path. 3. **Read the Label:** The label on the link from "UNC Offense" to "UMD Offense" is "24". 4. **Interpret the Value:** This means that 24 sequences went from "UNC Offense" to "UMD Offense". </reasoning> <answer>24</answer>

Table G.2: Sample VLM Responses for *value aggregation task* on graph, provided in fig. D.2. Correct Answer: 69

Model	LLM Answer	Reasoning
Claude-3.7-Sonnet	46	I'll analyze this node-link diagram to determine how many times UMD Offense transitioned from UNC Offense. <reasoning> Looking at the node-link diagram, I need to find transitions from "UNC Offense" to "UMD Offense". In this diagram: - The top node is "_Start" which flows to "UNC Offense" (69 transitions) - From "UNC Offense", there are two paths: - One path goes to "Made Shot" (22 transitions) - Another path goes to "Missed Shot" (24 transitions) - From "Made Shot", there's a path to "UMD Offense" (22 transitions) - From "Missed Shot", there's a path to "UMD Offense" (24 transitions) To find how many times UMD Offense transitioned from UNC Offense, I need to count all paths that start at UNC Offense and end at UMD Offense. These transitions happen through intermediate nodes: - UNC Offense → Made Shot → UMD Offense: 22 transitions - UNC Offense → Missed Shot → UMD Offense: 24 transitions So the total number of transitions from UNC Offense to UMD Offense is 22 + 24 = 46 transitions. </reasoning> <answer>46</answer>
Claude-3.5-Sonnet	46	Let me analyze the node-link diagram to determine the transitions from UNC Offense to UMD Offense. <reasoning> 1. Looking at the diagram, I need to find paths that go from "UNC Offense" to "UMD Offense" 2. I can see two possible paths from UNC Offense to UMD Offense: - UNC Offense -> Made Shot → UMD Offense - UNC Offense → Missed Shot → UMD Offense 3. Following the links and their labels: - Path through Made Shot: 22 sequences - Path through Missed Shot: 24 sequences 4. To get total transitions, I add both paths: 22 + 24 = 46 total transitions from UNC Offense to UMD Offense </reasoning> <answer>46</answer>
Claude-3.5-Haiku	22	Let me carefully analyze the node-link diagram to determine the number of times UMD Offense transitions from UNC Offense. <reasoning> 1. I'll trace the links connecting UNC Offense to UMD Offense 2. Looking at the diagram, I see a link from "UNC Offense" to "UMD Offense" with the value 22 3. This means that in 22 sequences, UNC Offense transitioned to UMD Offense </reasoning> <answer>22</answer>
Claude-3-Haiku	44	<reasoning> To determine how many times UMD Offense transitioned from UNC Offense, I will carefully examine the nodes and links in the diagram. The diagram shows that there are two paths leading from UNC Offense to UMD Offense: 1. UNC Offense -> Made Shot -> UMD Offense 2. UNC Offense -> Missed Shot -> UMD Offense Looking at the link widths and labels, I can see that the first path (UNC Offense -> Made Shot -> UMD Offense) has a count of 22, and the second path (UNC Offense -> Missed Shot -> UMD Offense) has a count of 22 as well. Adding these two counts together, I can determine that the total number of times UMD Offense transitioned from UNC Offense is 22 + 22 = 44. </reasoning> <answer>44</answer>
GPT-4.1	46	<reasoning> 1. The question asks for the number of times "UMD Offense" transitions from "UNC Offense". 2. In the diagram, "UNC Offense" is at the top, and there are two outgoing links from it: one to "Made Shot" (with a count of 23) and one to "Missed Shot" (with a count of 24). 3. From "Made Shot", there is a link to "UMD Offense" with a count of 22. 4. From "Missed Shot", there is also a link to "UMD Offense" with a count of 24. 5. To find the total number of transitions from "UNC Offense" to "UMD Offense", sum the counts of the links from "Made Shot" and "Missed Shot" to "UMD Offense": 22 (from "Made Shot") + 24 (from "Missed Shot") = 46. 6. These transitions all originate from "UNC Offense" via either "Made Shot" or "Missed Shot". </reasoning> <answer>46</answer>

Table G.3: Sample VLM Responses for *value aggregation task* on graph, provided in fig. D.2. Correct Answer: 69

Model	LLM Answer	Reasoning
GPT-4o	46	<reasoning> 1. Identify the relevant nodes and links: - The nodes involved are "UNC Offense" and "UMD Offense." - We need to find the link that connects "UNC Offense" to "UMD Offense." 2. Examine the diagram: - From "UNC Offense," there are two outgoing links: - One link goes to "Made Shot" with a count of 23. - Another link goes to "Missed Shot" with a count of 24. 3. Follow the paths: - From "Made Shot," there is a link to "UMD Offense" with a count of 22. - From "Missed Shot," there is a link to "UMD Offense" with a count of 24. 4. Calculate the total transitions from "UNC Offense" to "UMD Offense": - The transition from "UNC Offense" to "UMD Offense" via "Made Shot" is 22. - The transition from "UNC Offense" to "UMD Offense" via "Missed Shot" is 24. - Total transitions = 22 + 24 = 46. 5. Conclusion: - The total number of times "UMD Offense" transitioned from "UNC Offense" is 46. </reasoning> <answer>46</answer>
Llama-4-Maverick-17B	46	To determine how many times UMD Offense transitioned from UNC Offense, we need to examine the node-link diagram closely. 1. **Identify Relevant Elements**: The relevant elements in this visualization are the nodes labeled "UNC Offense" and "UMD Offense" and the links connecting them or leading to "UMD Offense". 2. **Understand the Diagram**: The diagram shows sequences of events with directed links indicating the flow from one event to another. The width of the links and the numeric labels next to them represent the count of sequences following that path. 3. **Analyze the Links**: - The node "UNC Offense" has two outgoing links, one leading to "Made Shot" and the other to "Missed Shot". - The counts for these links are 22 and 24, respectively. 4. **Transition to UMD Offense**: - "Made Shot" has a link to "UMD Offense" with a count of 22. - "Missed Shot" also has a link to "UMD Offense" with a count of 24. 5. **Calculate Total Transitions to UMD Offense from UNC Offense**: To find the total number of times UMD Offense transitioned from UNC Offense, we need to sum the counts of sequences that went from "UNC Offense" to "UMD Offense". This involves adding the counts of the paths "UNC Offense" -> "Made Shot" -> "UMD Offense" and "UNC Offense" -> "Missed Shot" -> "UMD Offense". 6. **Perform Calculation**: - Count via "Made Shot": 22 - Count via "Missed Shot": 24 - Total Count = 22 + 24 = 46 <reasoning> The total number of times UMD Offense transitioned from UNC Offense is calculated by summing the counts of the two paths leading from "UNC Offense" to "UMD Offense", which are through "Made Shot" and "Missed Shot". The count through "Made Shot" is 22, and through "Missed Shot" is 24. Thus, the total count is 22 + 24 = 46. </reasoning> <answer>46</answer>
Llama-4-Scout-17B	46	## Step 1: Identify the relevant elements in the visualization The node-link diagram shows the following elements: - A starting point labeled "_Start" with 69 sequences leading to "UNC Offense". - "UNC Offense" has two paths: one with 22 sequences leading to "Made Shot" and another with 24 sequences leading to "Missed Shot". - "Made Shot" and "Missed Shot" both lead to "UMD Offense". - "UMD Offense" has two paths: one with 34 sequences leading to "Rebound" and another with 22 sequences leading to "_Exit". ## 2: Determine the transition from UNC Offense to UMD Offense To find out how many times UMD Offense transitioned from UNC Offense, we need to look at the paths leading from "UNC Offense" to "UMD Offense". However, the direct link counts from "UNC Offense" to "UMD Offense" are not explicitly shown, but we can infer the total transitions to "UMD Offense" by looking at the counts leading into "UMD Offense". ## 3: Calculate the total transitions to UMD Offense The total transitions to "UMD Offense" can be determined by adding the sequences from "Made Shot" and "Missed Shot" since both lead to "UMD Offense". - From "Made Shot" to "UMD Offense", there are 22 sequences. - From "Missed Shot" to "UMD Offense", there are 24 sequences. ## 4: Sum the sequences leading to UMD Offense Total sequences leading to "UMD Offense" = 22 (from "Made Shot") + 24 (from "Missed Shot") = 46 sequences. ## 5: Provide the final answer Given that 46 sequences lead to "UMD Offense" from either "Made Shot" or "Missed Shot", this directly implies that UMD Offense transitioned 46 times from UNC Offense, as these are the only paths shown leading into UMD Offense. <reasoning> To determine how many times UMD Offense transitioned from UNC Offense, we look at the paths from UNC Offense to UMD Offense. The paths are as follows: - Made Shot: 22 sequences - Missed Shot: 24 sequences So, UMD Offense transitioned 22 + 24 = 46 times from UNC Offense. </reasoning> <answer>46</answer>

Table G.4: Sample VLM Responses for *value aggregation task* on graph, provided in fig. D.2. Correct Answer: 69

Model	LLM Answer	Reasoning
Llama-3.2-90B-Vision		**Step 1: Identify the relevant elements in the visualization.** * The node "UNC Offense" has two outgoing links, one labeled "Made Shot" and the other labeled "Missed Shot". * The node "UMD Offense" has one incoming link from the node "Rebound". **Step 2: Describe the thought process and calculations clearly step-by-step.** * To determine how many times UMD Offense transitioned from UNC Offense, we need to count the number of sequences that follow the path from UNC Offense to UMD Offense. * The link from UNC Offense to Made Shot has a count of 22, and the link from UNC Offense to Missed Shot has a count of 24. * Since there is no direct link from UNC Offense to UMD Offense, we need to consider the indirect paths through Rebound. * The link from Rebound to UMD Offense has a count of 34, but this includes sequences that may not have originated from UNC Offense. * To find the number of sequences that transitioned from UNC Offense to UMD Offense, we need to subtract the count of sequences that did not originate from UNC Offense from the total count of sequences that reached UMD Offense. **Step 3: Provide the final answer.** * Unfortunately, the provided information does not allow us to determine the exact number of times UMD Offense transitioned from UNC Offense, as we cannot accurately calculate the number of sequences that did not originate from UNC Offense. **Answer:** indeterminate
Llama-3.2-11B-Vision	2	**Step 1: Identify the relevant elements in the visualization that inform the answer.** The node-link diagram shows the sequence of events between different offenses. The relevant elements for this question are the nodes labeled "UNC Offense" and "UMD Offense," as well as the links connecting them. **Step 2: Describe the thought process and calculations in XML tags.** <reasoning> To determine how many times UMD Offense transitioned from UNC Offense, we need to examine the links between these two nodes. * There are two links connecting UNC Offense to UMD Offense. * Each link represents a transition from UNC Offense to UMD Offense. Therefore, UMD Offense transitioned from UNC Offense twice. </reasoning> **Step 3: Provide the final answer in the format <answer>X</answer> where X is the direct answer or 'indeterminate'.** <answer>2</answer>
Qwen-VL-Plus	34	<answer>34</answer> <reasoning> To determine how many times UMD Offense transitioned from UNC Offense, we need to follow the pathway from UNC Offense to UMD Offense in the diagram. 1. Start at the node labeled "UNC Offense." 2. Follow the directed link labeled "24" leading to the node labeled "Made Shot." 3. From "Made Shot," there is a link labeled "22" pointing to "UMD Offense." The link from "UNC Offense" to "Made Shot" has a label of 24, and the link from "Made Shot" to "UMD Offense" has a label of 22. Therefore, the total number of transitions from UNC Offense to UMD Offense is the sum of these two labels: 24 + 22 = 46. However, it seems there might be an error in the provided answer as the correct calculation based on the diagram should be 34 (from the "UMD Offense" node). Thus, the correct answer is 34. </reasoning>
Qwen-VL-Max	46	To determine how many times UMD Offense transitioned from UNC Offense, we need to follow the pathways in the diagram: 1. Start at the "UNC Offense" node. 2. Look for links leading out of this node. 3. There are two links: one labeled "22" leading to "Made Shot" and another labeled "24" leading to "Missed Shot". 4. From both "Made Shot" and "Missed Shot", there are links leading to "UMD Offense". 5. The link from "Made Shot" to "UMD Offense" is labeled "22". 6. The link from "Missed Shot" to "UMD Offense" is labeled "24". Adding these values together gives us the total number of transitions from UNC Offense to UMD Offense: <reasoning> The link from "Made Shot" to "UMD Offense" has a value of 22. The link from "Missed Shot" to "UMD Offense" has a value of 24. Therefore, the total number of transitions from UNC Offense to UMD Offense is 22 + 24 = 46. </reasoning> <answer>46</answer>

Table G.5: Sample VLM Responses for *value aggregation task* on graph, provided in fig. D.2. Correct Answer: 69

Model	LLM Answer	Reasoning
QVQ-Max	indeterminate	<reasoning> Okay, let's try to figure out how many times UMD Offense transitioned from UNC Offense based on the diagram provided. First, I need to understand the structure of the diagram. The nodes represent events, and the directed links show the transitions between these events. The vertical position indicates the chronological order, so the top node is the starting point, and it goes down from there. The link widths and the numbers next to them indicate the number of sequences that follow that path. Looking at the nodes, there's a starting point labeled "_Start" at the top. From there, there's a link to "UNC Offense" with the number 69. That means 69 sequences started with UNC Offense. Then, from UNC Offense, there are two outgoing links: one to "Made Shot" with 22 and another to "Missed Shot" with 24. So, after UNC Offense, 22 times they made a shot, and 24 times they missed. Next, the "Made Shot" node has a link to "Rebound" with 22, and "Missed Shot" also links to "Rebound" with 24. Then, from "Rebound," there's a link to "UMD Offense" with 34 and another to "_Exit" with 22. Finally, UMD Offense has a link to "_Exit" with 24. The question is asking how many times UMD Offense transitioned from UNC Offense. Wait, but looking at the diagram, the direct transition from UNC Offense is to Made Shot or Missed Shot, not directly to UMD Offense. So maybe the path is UNC Offense -> Missed Shot -> Rebound -> UMD Offense? Let me trace the possible pathways. Starting from UNC Offense (69 instances), 24 of those go to Missed Shot. From Missed Shot (24), all of them go to Rebound (24). Then, from Rebound, 34 go to UMD Offense. But wait, the link from Rebound to UMD Offense is 34. However, the total coming into Rebound is 22 (from Made Shot) + 24 (from Missed Shot) = 46. But the link from Rebound to UMD Offense is 34, and to _Exit is 22. That adds up to 56, which is more than 46. That seems inconsistent. Hmm, maybe I'm misunderstanding the numbers. Wait, the numbers on the links represent the count of sequences following that path. So from Rebound, 34 go to UMD Offense and 22 to _Exit. But the total coming into Rebound is 22 (from Made Shot) + 24 (from Missed Shot) = 46. But 34 + 22 = 56. That doesn't add up. There's a discrepancy here. Maybe the numbers on the links are not the counts but something else? Or perhaps the diagram is showing cumulative counts? Wait, the instruction says "Link widths are proportional to the number of sequences following that path. Numeric labels next to each link show the exact count of sequences." So the numbers should be the exact counts. But then, how does 22 + 24 = 46 incoming to Rebound result in 34 + 22 = 56 outgoing? That's impossible. There must be an error here, but since I have to answer based solely on the visualization, maybe I'm misinterpreting the flow. Alternatively, perhaps the link from Rebound to UMD Offense is 34, which would mean that out of the 46 sequences that reached Rebound, 34 went to UMD Offense and 22 exited. But 34 +22 is 56, which is more than 46. That can't be. Therefore, this suggests that the numbers might not be additive in the way I think. Maybe the links from Rebound are separate paths. Wait, perhaps the Rebound node has two outgoing links: one to UMD Offense (34) and one to _Exit (22). But the total number of sequences leaving Rebound should equal the number coming in. Since 22 +24 =46 come into Rebound, then the outgoing links should sum to 46. But 34 +22=56. This is a problem. Unless the numbers are not the counts but something else. But the instructions say they are the exact count. This inconsistency might be a mistake in the diagram, but since I have to work with what's given, perhaps I need to proceed differently. The question is about transitions from UNC Offense to UMD Offense. Looking at the diagram, there's no direct link from UNC Offense to UMD Offense. The path would have to go through Missed Shot and Rebound. So the number of times UMD Offense transitions from UNC Offense would be the number of sequences that go from UNC Offense to Missed Shot to Rebound to UMD Offense. So starting from UNC Offense, 24 go to Missed Shot. From Missed Shot, all 24 go to Rebound. From Rebound, 34 go to UMD Offense. But wait, if 24 sequences are coming from Missed Shot to Rebound, and then 34 go from Rebound to UMD Offense, how can 34 be more than 24? That would imply that some sequences from other paths (like Made Shot) are also contributing. But Made Shot sends 22 to Rebound. So total Rebound is 22 +24=46. Then, from Rebound, 34 go to UMD Offense. So the 34 includes sequences from both Made Shot and Missed Shot. But the question is specifically about transitions from UNC Offense to UMD Offense. Since UNC Offense leads to Missed Shot (24) which leads to Rebound (24), and then from Rebound, 34 go to UMD Offense. But how many of those 34 are from the original 24 that came through Missed Shot? *Continued.....* </reasoning> <answer>indeterminate</answer>

Table G.6: Sample VLM Responses for *value aggregation task* on graph, provided in fig. D.2. Correct Answer: 69

Model	LLM Answer	Reasoning
Qwen2.5-VL-72B	46	1. **Relevant Elements:** The node "UNC Offense" is connected to two nodes: "Made Shot" and "Missed Shot." The node "UMD Offense" is connected from both "Made Shot" and "Missed Shot." The link widths and numeric labels indicate the number of sequences. 2. **Reasoning:** From "UNC Offense," there are 22 sequences leading to "Made Shot" and 24 sequences leading to "Missed Shot." All sequences from "Made Shot" (22) and "Missed Shot" (24) transition to "UMD Offense." Therefore, the total number of transitions from "UNC Offense" to "UMD Offense" is the sum of these sequences. Calculation: 22 + 24 = 46 3. **Final Answer:** <answer>46</answer>
Qwen2.5-VL-32B	24	To answer the question, let's carefully analyze the node-link diagram step by step: ### 1. Identify Relevant Elements - The **UNC Offense** node has two outgoing links: - One leading to **Made Shot** with a count of 22. - One leading to **Missed Shot** with a count of 24. - The **Missed Shot** node has an outgoing link to **UMD Offense** with a count of 24. ### 2. Understand the Transition The question asks how many times **UMD Offense** transitioned from **UNC Offense**. This means we need to trace the pathway from **UNC Offense** to **UMD Offense**. - From **UNC Offense**, there is a direct path to **Missed Shot** (count = 24). - From **Missed Shot**, there is a direct path to **UMD Offense** (count = 24). Since the link from **Missed Shot** to **UMD Offense** has a count of 24, this indicates that 24 sequences transitioned from **UNC Offense** to **UMD Offense** via the **Missed Shot** event. ### 3. Verify and Conclude There are no other pathways from **UNC Offense** to **UMD Offense** in the diagram. The only valid transition is through the **Missed Shot** event, which has a count of 24. ### Final Answer <answer>24</answer>
Qwen2.5-VL-7B	22	###Relevant Elements: - The node labeled "UNC Offense" is at the top of the diagram. - The node labeled "UMD Offense" is below it. - There is a link between "UNC Offense" and "UMD Offense" with a label of 22. ### Reasoning: 1. The link between "UNC Offense" and "UMD Offense" represents a transition from one team's offense to the other. 2. The label on this link, which is 22, indicates the number of times this transition occurred. ### Final Answer: <answer>22</answer>

Table G.7: Sample VLM Responses for *value aggregation task* on graph, provided in fig. D.2. Correct Answer: 69