SEETHEFLOW: STRUCTURE-GUIDED FLOWCHART PARSING

Anonymous authors
Paper under double-blind review

ABSTRACT

Flowchart parsing requires vision-language models (VLMs) to recover not only textual and visual content but also the underlying topology. VLMs performance degrades sharply as diagrams become larger and more densely connected. Existing approaches primarily rely on visual reasoning or additional VLM inference, while the benefit of explicitly supplying externally extracted structure remains less explored. We introduce **SeeTheFlow**, a VLM-training-free framework that extracts an explicit structural prior from flowchart images using deterministic Computer Vision and OCR, then conditions a VLM on this prior through a lightweight, cost-aware router that intervenes only on diagrams likely to benefit. Across six VLM backbones, SeeTheFlow improves connection weighted F1 by up to 30 percentage points on large, structurally complex diagrams and by 11.2 points on real-world flowcharts, while invoking a second VLM inference on only roughly one-third of inputs rather than applying structural guidance unconditionally. We further introduce a 1,200-instance benchmark with controlled structural complexity, isolating diagram topology from lexical and rendering variation. Our results show that explicit, selectively applied structural evidence can improve robustness to structural complexity while avoiding the cost of additional inference on every diagram. Our code and data are publicly available at `https://anonymous.4open.science/r/SeeTheFlow_ICLR-0502`.

1 INTRODUCTION

Flowcharts are visual artifacts for representing procedural logic and workflows across domains such as software engineering, business process modeling, and education Arbaz et al. (2024); Deka & Devereux (2025). Unlike natural images, their meaning depends not only on recognizing textual labels and visual symbols but also on recovering how these elements are connected. A model can correctly recognize every node yet produce an incorrect interpretation if it misses a connection or follows the wrong branch. Thus, flowchart understanding requires recovering the underlying topology together with its visual and textual content.

Despite their visual reasoning capabilities, Vision-Language Models (VLMs) struggle with flowcharts as structural complexity increases Kahou et al. (2018); Masry et al. (2022); Peng et al. (2023); Yang et al. (2023); Ye et al. (2024). Existing benchmarks such as FlowLearn Pan et al. (2024) and FlowVQA Singh et al. (2024) show that strong VLMs perform well on simpler diagrams but degrade as nodes, connections, and structural dependencies increase. This suggests a limitation beyond visual recognition: *image-only VLMs have difficulty recovering global structure as diagrams become larger and denser*. The output may remain plausible and contain correctly recognized labels while its edges and control flow are wrong. This motivates our question: *Can we provide a model with structural information that is difficult to recover directly from the image?*

We introduce *SeeTheFlow*, a VLM-training-free framework that combines an off-the-shelf VLM with a deterministic CV/OCR-derived structural prior capturing candidate nodes and their spatial and connectivity structure, and optional graph verification. Simple diagrams may be handled adequately by image-only inference, whereas complex diagrams can benefit from structural guidance. Because the prior can be unreliable and an additional VLM call is not always beneficial, a lightweight SVM router uses inexpensive CV/OCR features to decide whether to retain the image-only result or invoke a structure-conditioned call (Section 5.2), avoiding unnecessary inference and its associated cost.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

When invoked, the VLM receives both the image and structural guidance, while a verifier identifies structural inconsistencies in the predicted graph and provides targeted feedback for refinement. We compare SeeTheFlow-Always, which always provides the prior, with SeeTheFlow-ImageCritique, which performs image-only iterative refinement, to isolate the contribution of structural guidance (Section 7).

Our central finding is that structural guidance is particularly effective on large, highly connected flowcharts. For Claude and GPT-5.5, SeeTheFlow improves CW-F1 by approximately 30 pp on these diagrams. The gains persist when node labels are randomized or permuted, indicating that they are not chiefly driven by textual semantics. On real-world diagrams, however, inaccurate structural guidance can reduce performance, motivating selective routing. Ground-truth routing reaches 48.8 CW-F1 compared with 44.7 for the learned router, indicating remaining headroom in predicting when structural guidance will help. We make the following contributions:

- We introduce SeeTheFlow, a VLM-training-free framework combining a CV/OCR-derived structural prior, optional graph verification, and adaptive routing with an off-the-shelf VLM.
- We show that structural guidance improves CW-F1 by about 30 pp for Claude and GPT-5.5 on large, highly connected flowcharts.
- We introduce a controlled benchmark of 1,200 flowcharts spanning 200 domains and six structural conditions, with graph-structured ground truth for evaluating diagram size and structural complexity.
- We show that the benefits of structural guidance persist under controlled label perturbations, indicating that the gains are not primarily due to textual semantics.
- We show that adaptive routing concentrates intervention on complex diagrams, from 4% of small diagrams to 79% of large diagrams, achieving 44.7 CW-F1 with an average of 1.8 VLM calls per diagram, compared with 48.8 for a ground-truth oracle that selects the better inference strategy.

2 Related Work

Flowchart understanding has progressed from segmentation- and OCR-based pipelines Arbaz et al. (2024); Shukla et al. (2023) to VLM-based approaches. Benchmarks such as FlowchartQA Tannert et al. (2023), FlowVQA Singh et al. (2024), and FlowLearn Pan et al. (2024) examine flowchart interpretation and reasoning, with studies reporting performance degradation as diagram complexity increases Shi et al. (2026). Related challenges arise in broader diagram-understanding tasks, including ChartQA Masry et al. (2022) and FigureQA Kahou et al. (2018).

General-purpose VLMs OpenAI (2023); Team et al. (2023); Liu et al. (2023); Bai et al. (2023) provide a foundation for diagram understanding, but recovering relationships among visual elements remains challenging. Recent approaches address this problem through region-based reasoning Li et al. (2025), arrow-guided interpretation Omasa et al. (2025), attention-based grounding Esmaeilkhani & Latecki (2026), textual intermediate representations Ye et al. (2025), diagram-to-Mermaid conversion Deka & Devereux (2025), and edge-map augmentation Dou et al. (2026). These methods introduce different forms of guidance: TEXTFLOW represents diagram information in text for subsequent reasoning, while EdgeFlow and arrow-guided approaches emphasize visual cues for recovering connections.

A complementary line of research uses tools and iterative feedback to improve model outputs, from tool-using agents Schick et al. (2023); Yao et al. (2023) to self-refinement approaches that identify and correct generated errors Shinn et al. (2023); Madaan et al. (2023); Gupta & Kembhavi (2023); Gou et al. (2023). In diagram understanding, such feedback can support repeated interpretation or revision of an initially predicted structure, motivating our use of external structural evidence and verification alongside visual inference.

SeeTheFlow builds on these directions by providing the VLM with an explicit graph-structured prior extracted from the flowchart image. Unlike visual edge or arrow cues, the prior represents the recovered topology as a graph that can be inspected and corrected before being supplied to the VLM. Rather than applying structural guidance to every diagram, SeeTheFlow also determines when an additional structure-conditioned inference call is warranted. Our evaluation examines how the benefit of this guidance varies with diagram complexity and prior reliability, and whether adaptive routing can reduce the additional inference cost. The framework uses off-the-shelf VLMs without task-specific fine-tuning.

3 DATASET

Existing Benchmarks and Their Limitations. We consider two widely used flowchart datasets: *Flowchart2Mermaid* Deka & Devereux (2025) (1,800 flowcharts) and *FlowGen* Shi et al. (2026) (12,000 diagrams). *Flowchart2Mermaid* consists primarily of simple, near-linear diagrams with a limited set of node shapes, and models can score well without being stress-tested under high structural complexity. *FlowGen* contains structurally complex diagrams, but node labels are randomly assigned within broad domains and lack semantic coherence, conflating structural difficulty with lexical inconsistency. We also evaluated *FlowVQA* Singh et al. (2024) across commercial models on instances with more than 30 nodes; however, the evaluated models achieved near-perfect scores approaching 1.0. This performance sharply contrasts with results on datasets of comparable structural complexity, suggesting potential benchmark contamination. Consequently, we exclude *FlowVQA* from our primary benchmark suite. Full results are reported at Appendix C.2.

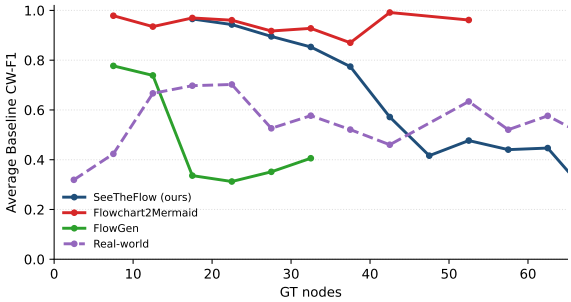


Figure 1: Baseline F1 by gold node count, micro-averaged across Claude-4.6, GPT-5.5, and Gemini-3.5 on SeeTheFlow, Flowchart2Mermaid, and FlowGen.

We also evaluate transfer to real-world flowcharts from FlowSight herbit2004 (2025), FC Detection Galirage Inc. (2025); Omasa et al. (2025), and FlowCE Zhang et al. (2024). We manually filter each source to retain node-and-edge flowcharts, excluding icon and emoji collages. Automatic CW-F1 evaluation uses approximately 1,000 FlowSight images with author-provided Mermaid diagrams as ground truth. The 182 FC-Detection and FlowCE files contain candidate Mermaid representations and are used only for diagnostics and release, not quantitative evaluation. These real-world images also introduce photographic noise and difficult OCR conditions, under which inaccurate structural priors may mislead the VLM.

Our synthetic dataset (SeeTheFlow). We introduce a synthetic 1200-instances suite with coherent labels and controlled topology. Each of 200 domains is instantiated under six structural conditions (Easy/Medium/Hard $\times$ with/without subgraphs), so difficulty variants are *controlled structural perturbations of shared domains*, not independent semantic samples. Diagrams are generated with GPT-5 Singh et al. (2025) under explicit node/connectivity constraints; Figure 1 compares size coverage to Flowchart2Mermaid/FlowGen and includes a size-stratified real Baseline curve.

We define three difficulty levels: **Easy** ($\geq$15 nodes/edges, near-linear), **Medium** ($\geq$25 nodes/edges, moderate branching), and **Hard** ($\geq$35 nodes/edges, dense nested control flow), plus terminals, cylinders, and hierarchical subgraphs underrepresented in prior benchmarks. The difficulty levels also differ structurally: harder diagrams contain more loops, nested branches, and re-convergent paths, rather than differing only in node count.

Dataset diagnostics. The Easy, Medium, and Hard conditions differ in structural complexity, not merely diagram size. As difficulty increases, the average number of loops rises from 0.92 to 3.85, branches from 3.97 to 7.16, nested control-flow structures from 7.79 to 25.64, and reconvergent paths from 2.68 to 5.46. Sampled SBERT and Levenshtein similarities remain low (≈ 0.08 and 0.20, respectively), indicating variation across diagram instances. Detailed results appear in Appendix B.

4 EVALUATION METRICS

Prior work evaluated flowchart understanding using triplet-based metrics. A flowchart is represented as a set of triplets encoding labeled edges (A, X, B), connectivity $(A, \texttt{connectedTo}, B)$, and subgraph membership $(A, \texttt{partOf}, B)$. However, these metrics do not fully capture structural errors, such as missing node shapes or incorrect connectivity. We nonetheless report triplet-based results in Appendix C for a comprehensive comparison with prior work.

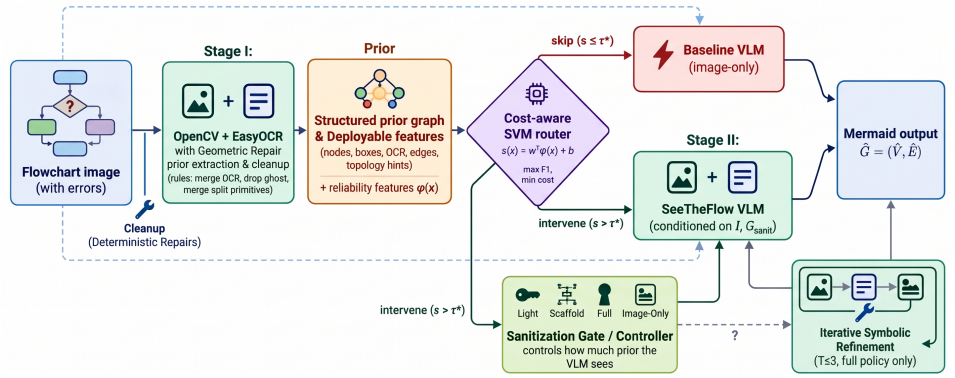


Figure 2: SeeTheFlow pipeline. A CV/OCR pass extracts structural prior and routing features. The router selects image-only or structure-conditioned inference, with optional verification.

We instead propose *Coverage-Weighted F1 (CW-F1)* as our primary metric for evaluating graph recovery. To compare predicted and ground-truth graphs, we construct a weighted bipartite graph between their nodes using fuzzy label similarity and apply the Hungarian algorithm to obtain a one-to-one correspondence map M. *Node Coverage* (NC $= |M|/|\mathcal{V}_{\text{gt}}|$) measures the proportion of ground-truth nodes recovered. We also report *Structural F1* over edges (ignoring labels). CW-F1 scales edge precision and recall by node coverage:

$$\text{CW-F1} = \frac{2 \cdot \text{P}_{\text{cw}} \cdot \text{R}_{\text{cw}}}{\text{P}_{\text{cw}} + \text{R}_{\text{cw}}}, \quad \text{P}_{\text{cw}} = \frac{|\mathcal{E}_{\text{correct}}|}{|\mathcal{E}_{\text{pred}}|} \cdot \text{NC}, \quad \text{R}_{\text{cw}} = \frac{|\mathcal{E}_{\text{correct}}|}{|\mathcal{E}_{\text{gt}}|} \cdot \text{NC} \tag{1}$$

This simplifies to CW-F1 $=$ NC $\times$ Edge-F1, with $\text{P}_{\text{cw}} = 0$ when $|\mathcal{E}_{\text{pred}}| = 0$. By incorporating node coverage, CW-F1 penalizes missing nodes in addition to errors in the recovered connections. Because node alignment M is based on label similarity, CW-F1 does not directly measure shape or subgraph recovery. Node matching uses edit-similarity thresholds of 0.90 for labels and 0.70 for assignment. These thresholds were selected to balance strict label fidelity against tolerance for minor OCR noise, consistent with common practice in fuzzy graph-matching evaluation. We report NC, Edge-F1, and shape and subgraph metrics in Appendix C.9, while Section 6 examines sensitivity to node labels by perturbing them without changing the diagram layout.

5 Methodology

In this section, we present the three components of *SeeTheFlow* (Figure 2). Given a flowchart image $\mathcal{I}$, the framework predicts its graph representation $\hat{\mathcal{G}} = (\hat{\mathcal{V}}, \hat{\mathcal{E}})$, comprising the diagram's nodes and directed connections. First, deterministic CV/OCR operations construct an approximate structural prior $\mathcal{G}_{\text{prior}}$. Second, an adaptive router determines whether to retain the image-only prediction or invoke an additional, structure-conditioned VLM call. Third, when full structural guidance is selected, optional verification and refinement address potential inconsistencies in the predicted graph.

To isolate the source of any gain, we define an ablation ladder used throughout Section 7: Baseline (single-pass image-only decoding), SeeTheFlow-ImageCritique (an inference-control ablation: the VLM iteratively revises its own prior prediction using only the image, with no external structural representation), SeeTheFlow-Always (the structural prior exposed unconditionally, no routing or verification), and **SeeTheFlow** (routed Prior+Verify). SeeTheFlow-ImageCritique and SeeTheFlow both use iterative calls; comparing them isolates whether gains come from extra inference or from the structural prior itself. Full prompts and protocol appear in Appendix C.4.

5.1 Structural Prior Construction

Given an image $\mathcal{I}$, SeeTheFlow constructs an approximate graph $\mathcal{G}_{\text{prior}} = (\mathcal{V}_{\text{prior}}, \mathcal{E}_{\text{prior}})$ using deterministic visual operations. OpenCV Bradski (2000) detects graphical elements, EasyOCR AI (2020) extracts text, and spatial heuristics associate labels with nodes and recover their connections.

The extracted graph may contain errors caused by fragmented shapes, overlapping contours, or incorrect associations between text and graphical elements. We apply three rules to correct common

extraction errors: (i) merge horizontally overlapping OCR tokens separated by a vertical gap below δ_y; (ii) remove spurious contours without associated text that overlap more than two neighboring contours; and (iii) merge fragmented shapes, such as cylinders detected as separate rectangles and ellipses. The resulting graph approximates the diagram's topology but may still contain missing or incorrect elements and connections. Thus, we treat $\mathcal{G}_{\text{prior}}$ as supplementary evidence rather than ground truth. Before supplying the prior to the VLM, a sanitization policy determines how much structural information to expose. The policy supports four settings: *light*, *scaffold*, *full*, and *image-only*. These settings vary the amount of structural guidance, including the option to retain image-only inference. The VLM is instructed to prefer the image when it conflicts with the supplied prior, although this does not guarantee that incorrect structural information will be disregarded. The same CV/OCR pass produces the features used by the adaptive router. Both the structural prior and the routing features are obtained without access to the ground-truth graph.

5.2 Adaptive Structural Guidance

Structural guidance is not universally beneficial: simple flowcharts may already parse correctly from the image alone, while an inaccurate prior can introduce errors, particularly on noisy real-world images, and applying an extra structure-conditioned VLM call to every input adds cost and may reduce accuracy. We address this tradeoff with an adaptive router that chooses between the image-only prediction and an additional structure-conditioned call.

Routing features. For each input image $\mathcal{I}$, the CV/OCR pass produces a feature vector $\phi(\mathcal{I})$. Its features include estimated node and edge counts, the edge-to-node ratio, source and sink counts, chain structure, the fraction of nodes without recognized text, and the isolation rate. These features characterize the extracted structure and provide inexpensive indicators of diagram complexity and prior quality. Ground-truth node counts, topology, evaluation annotations, and corpus identity are not used as routing inputs.

Learning the routing decision. To train the router, we evaluate both inference strategies on each training diagram: image-only inference and structure-conditioned inference. Let their respective CW-F1 scores on image $\mathcal{I}_i$ be $F1_{\text{base}}(\mathcal{I}_i)$ and $F1_{\text{agent}}(\mathcal{I}_i)$. The observed improvement from structural guidance is $\Delta_i = F1_{\text{agent}}(\mathcal{I}_i) - F1_{\text{base}}(\mathcal{I}_i)$. We train a linear support vector machine (SVM) to predict whether this improvement exceeds a threshold $\varepsilon = 0.02$. The training label is $y_i = \mathbf{1}[\Delta_i > \varepsilon]$. Thus, the router classifies whether structural guidance is likely to improve CW-F1 beyond this threshold rather than predicting the exact improvement. We also evaluate an RBF-kernel SVM, SVR on the predicted improvement, logistic regression, and a simple size threshold.

The routing features do not depend on the VLM, although the observed improvements do. Hence, we train one router using paired outcomes collected across models rather than fitting a separate router for each model. Training, validation, and test sets use a group-aware 60/20/20 split by underlying diagram. Different renderings or perturbations of the same diagram are kept within the same split. We repeat the evaluation over 30 splits; further details appear in Appendix C.4.

Cost-aware routing. The trained SVM assigns each image a score $s(\mathcal{I}) = w^\top \phi(\mathcal{I}) + b$. A threshold τ determines whether structural guidance is invoked:

$$r_\tau(\mathcal{I}) = \begin{cases} \text{SeeTheFlow}, & s(\mathcal{I}) > \tau, \\ \text{Baseline}, & \text{otherwise.} \end{cases} \tag{2}$$

Each diagram first receives an image-only VLM prediction. If the router selects SeeTheFlow, an additional structure-conditioned call is made; otherwise, the image-only result is retained. For a set of n diagrams, the resulting average CW-F1 is

$$F1_{\text{routed}}(\tau) = \frac{1}{n} \sum_{i=1}^{n} \left(\mathbf{1}[s(\mathcal{I}_i) > \tau] F1_{\text{agent}}(\mathcal{I}_i) + \mathbf{1}[s(\mathcal{I}_i) \leq \tau] F1_{\text{base}}(\mathcal{I}_i) \right). \tag{3}$$

The fraction of diagrams receiving the additional structure-conditioned call is $C(\tau) = \frac{1}{n} \sum_{i=1}^{n} \mathbf{1}[s(\mathcal{I}_i) > \tau]$. We select the routing threshold on the validation set by balancing prediction quality against the rate of structure-conditioned intervention:

Table 1: Inference cost and CW-F1 improvement (percentage points) over Baseline, averaged across Claude-4.6, GPT-5.5, and Gemini-3.5. Latency is estimated from VLM calls, except for SeeTheFlow, which uses measured latency.

	Cost Metrics		Synthetic Δ		Real-World Δ	
	VLM Calls	Latency(s)	Pooled	Large (>40)	Pooled	Large (>30)
SeeTheFlow-Always (ours)	2.0	≈ 42	+21.0	+27.9	+2.8	+0.6
SeeTheFlow-ImageCritique (ours)	2.0	≈ 28	+11.9	+16.5	-1.5	-0.7
SeeTheFlow (ours)	1–2 (avg 1.8)	21–42 (avg 35)	+14.2	**+26.1**	**+11.2**	**+11.6**
Baseline	1.0	≈ 14	—	—	—	—
CoR Li et al. (2025)	16	≈ 224	−1.6	−21.6	+6.8	+1.4
Arrow-Guided (ours) Omasa et al. (2025)	1.0	≈ 14	−5.2	+11.1	+7.6	+10.0
EdgeCSC diagnostic	5.6	≈ 78	**+25.9**	+25.1	+3.2	+6.2

$$\tau^\star = \arg\max_{\tau} \left[F1_{\text{routed}}(\tau) - \lambda C(\tau) \right], \qquad \lambda = 0.05. \tag{4}$$

Here, λ assigns a relative cost to invoking an additional structure-conditioned call; it is not a monetary cost. We evaluate the quality–cost tradeoff across routing thresholds in Figure 4. The measure $C(\tau)$ captures the intervention rate, while optional refinement calls, described next, are included in the total inference cost reported for the deployed framework.

5.3 Verification and Refinement

When full structural guidance is selected, SeeTheFlow can further refine the predicted graph using feedback from deterministic structural checks. At iteration t, the verifier compares the current prediction $\hat{\mathcal{G}}^{(t)}$ with the structural prior $\mathcal{G}_{\text{prior}}$, checking for five common flowchart-convention violations: missing nodes, isolated nodes, decision nodes with fewer than two outgoing edges, unusual source/sink counts, and bidirectional edges. These checks reflect common conventions rather than universal constraints, and a deviation does not necessarily indicate an error, since the prior itself may be inaccurate.

When potential inconsistencies are detected, the verifier provides targeted feedback to the VLM, which generates a revised graph. A revision is accepted only if it can be parsed, removes no edges, deletes at most two nodes, and introduces no additional check violations; otherwise, the previous prediction is retained. This conservative rule protects recovered structure but may reject valid corrections that require removing incorrect nodes or edges. Refinement is limited to $T = 1$ additional VLM calls and is enabled only under full structural guidance. Across the deployed framework, including image-only inference, structure-conditioned inference, and any refinement calls, the measured average is 1.8 VLM calls per diagram (Table 1).

6 Results and Analysis

Our central finding is that structural guidance becomes more valuable as flowcharts grow in structural complexity. Image-only VLM performance declines as diagrams become larger and more connected, while SeeTheFlow reduces this decline by providing an explicit structural prior. However, the benefit depends on the reliability of the prior and the VLM's ability to use it. These findings motivate selective structural guidance rather than applying the prior to every diagram.

We first examine how structural guidance affects performance across diagram complexity and inference cost, followed by differences across VLMs and real-world diagrams. We then use label perturbations and error analysis to investigate the sources of these gains.

Structural guidance and diagram complexity. Across Claude-4.6, GPT-5.5, and Gemini-3.5, SeeTheFlow improves CW-F1 by 14.2 pp on synthetic diagrams, rising to 26.1 pp on large diagrams with more than 40 nodes (Table 1). Its win rate increases from 57% on small diagrams to 72% on large ones. Thus, structural guidance does not produce a uniform improvement; its benefit increases with diagram complexity. The comparison methods provide additional context. CoR and Arrow-Guided do not improve pooled synthetic performance over Baseline (-1.6 and -5.2 pp, respectively), although

Arrow-Guided gains 11.1 pp on large diagrams. These results suggest that additional inference or visual guidance alone does not reproduce the benefits of an explicit graph-structured prior. EdgeCSC achieves a larger pooled synthetic gain (25.9 pp), but follows a different protocol averaging 5.6 VLM calls. Thus, we use it as a diagnostic comparison rather than a directly comparable deployment strategy. Under the secondary triplet-based protocol (Appendix C), SeeTheFlow also improves mean Macro F1 over Baseline on synthetic diagrams, from 0.472 to 0.554 pooled across models (0.326→0.439 on Large diagrams).

Adaptive routing and inference cost. Structural prior extraction takes less than 3 seconds on CPU, while an additional VLM call accounts for most of the intervention cost. The router thus aims to invoke structural guidance where its expected benefit justifies that cost. Using features extracted by CV/OCR, it routes only 4% of small diagrams and 10% of medium diagrams, compared with 79% of large diagrams (Figure 4). These decisions use estimated structural properties rather than ground-truth diagram information. On large synthetic diagrams, SeeTheFlow-Always improves CW-F1 by 27.9 pp, compared with 26.1 pp for SeeTheFlow. The routed framework approaches the large-diagram benefit of always-on intervention while averaging 1.8 VLM calls per diagram instead of applying the additional call to every input. It also limits unnecessary intervention on smaller diagrams, where the image-only prediction may already be correct. By limiting intervention on diagrams unlikely to benefit, the router reduces unnecessary VLM calls and their associated computational cost.

Variation across VLMs. The benefit of structural guidance also depends on the VLM receiving the prior (Table 2): Claude and GPT-5.5 improve modestly (3.5 and 4.9 pp), Gemini and Gemma show larger gains (22.8 and 26.7 pp, though Gemma starts near-floor at 1.3 CW-F1), while Llama-4 and Qwen-3.5 regress (14.0 and 7.8 pp). This shows that supplying an explicit graph does not guarantee better predictions, the VLM must also interpret the prior alongside the image, and our experiments do not isolate whether regressions stem from copying incorrect structural information, output-format errors, or other model-specific behavior; incorporating VLM identity into routing is a possible extension of the current model-independent router.

Table 2: Baseline and SeeTheFlow CW-F1 across models, pooled over scored datasets. Bold values indicate the highest CW-F1 score and largest gain (Δ). Win rate excludes ties; Gemma has a near-floor baseline.

Backbone	Baseline	SeeTheFlow	Δ (pp)	Win %
Claude-4.6	64.3	67.8	+3.5	34
GPT-5.5	65.4	**70.3**	+4.9	49
Gemini-3.5	39.9	62.7	+22.8	71
Gemma-3	1.3	27.9	**+26.7**	91
Llama-4	26.2	12.2	−14.0	27
Qwen-3.5	37.1	29.2	−7.8	15

Transfer to real-world flowcharts. On the pooled real-world datasets, SeeTheFlow improves CW-F1 by 11.2 pp overall and 11.6 pp on large diagrams (Table 1). On FlowSight dataset with verified ground truth, the improvement is smaller (from 54.0 to 60.8 CW-F1, +6.7 pp, 95% CI $[-2.1, +17.8]$). The label-strict triplet variant shows the same pattern on FlowSight ($0.552 \rightarrow 0.608$, +5.6 pp), driven mainly by node coverage (+6.8 pp; Appendix C).

Prior reliability matters most on real-world diagrams: on the hard subset, an unrepaired prior reduces CW-F1 from 66.7 to 59.1, below baseline, while structural repair raises it to 67.0, and a ground-truth oracle reaches 69.5. This shows both the cost of inaccurate structural information and that repairing the prior recovers most of the loss, with the remaining gap suggesting room for better decisions about when to apply structural guidance.

Sensitivity to node labels. An alternative explanation for the observed gains is that the structural prior helps the VLM recognize node labels rather than recover the diagram's topology. To examine this possibility, we randomize or permute node labels while preserving the diagram layout. Under these perturbations, image-only CW-F1 decreases by 44.4 pp for GPT-5.5 and 31.6 pp for Claude, whereas SeeTheFlow is comparatively stable. The extracted visual prior also changes little (20.2 to 18.8), consistent with its reliance on visual layout and connectivity. The same pattern appears on FlowGen and a random split of Flowchart2Mermaid. For example, Claude's image-only CW-F1 falls to 0.006 under label perturbation, while structure-conditioned inference remains comparatively

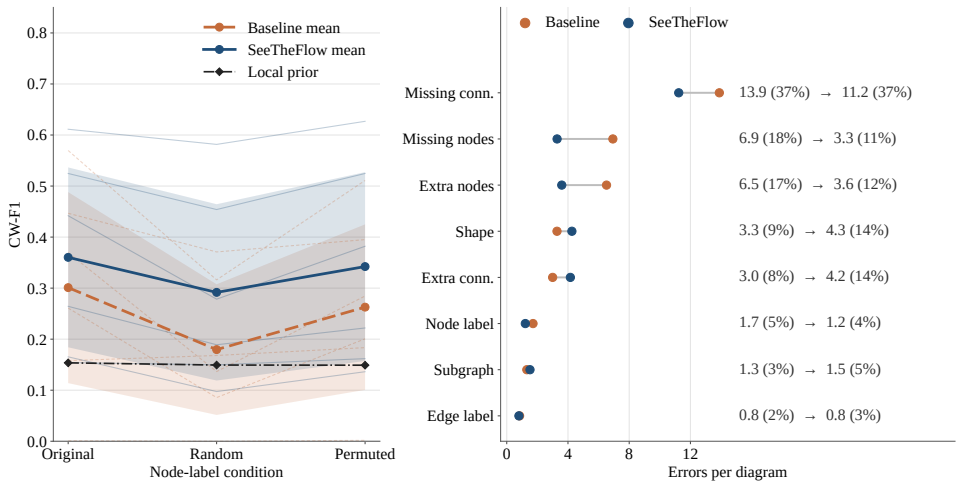


Figure 3: Left: CW-F1 under node-label perturbations. Baseline performance declines sharply, while SeeTheFlow is comparatively stable. Right: Error distributions for Baseline and SeeTheFlow. Structural guidance reduces missing nodes and connections but increases spurious connections.

robust (Figure 3, Appendix C). These results indicate that the gains from structural guidance are not explained solely by improved recognition or use of node-label semantics.

Structural error analysis. The error analysis provides further evidence of how structural guidance affects graph recovery. On synthetic diagrams, SeeTheFlow reduces the proportion of missing connections from 36.3% to 28.3% and missing nodes from 20.5% to 12.2%, while spurious connections increase from 19.5% to 30.3% (Figure 3). On hard diagrams, connection F1 increases from 0.32 to 0.48. These results indicate that structural guidance primarily reduces omissions in the recovered graph, but can also introduce connections absent from the source diagram. Its benefit depends on the balance between missing and spurious structure, the reliability of the extracted prior, and the VLM's ability to use that prior appropriately. Adaptive routing addresses the first consideration by selecting diagrams likely to benefit from intervention; improving prior reliability and model-specific use of structural guidance remain directions for future work (Section 8).

7 Ablation Studies

The previous section established a trade-off: structural guidance recovers topology missed by image-only decoding, but an unreliable prior can introduce errors. We examine three questions: how much of the gain comes from the prior rather than additional inference, how prior unreliability affects performance, and if selective routing can retain the benefits without intervening on every diagram.

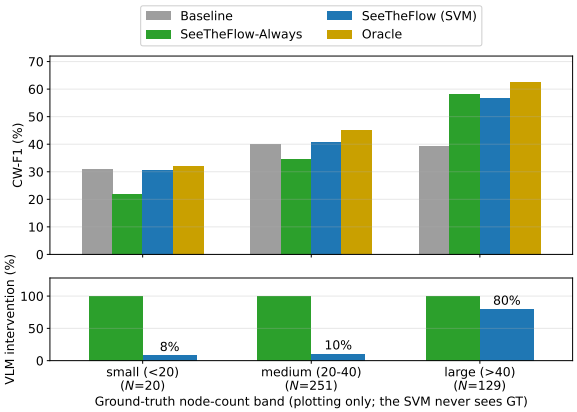


Figure 4: Routing by diagram size. SeeTheFlow-Always scores 58.3 on large diagrams but 22.1 on small ones (Baseline: 31.1). The SVM invokes structural guidance on 4%, 10%, and 79% of small, medium, and large diagrams, respectively.

Does the structural prior drive the gain? On large synthetic diagrams (Appendix C.6), SeeTheFlow-Always raises CW-F1 from 49.6 to 81.1 (+31.5 pp) over Baseline. This is our primary evidence that exposing an explicit structural prior recovers topology that image-only decoding misses.

Does external structure help beyond additional visual inference? SeeTheFlow-ImageCritique is an inference-control ablation: it tests whether SeeTheFlow's benefit can be reproduced by spending

an additional VLM call on image-only self-critique, without supplying any external structural representation. SeeTheFlow-ImageCritique reaches 83.4 CW-F1 on the same clean synthetic split – matching or exceeding SeeTheFlow-Always. Both explicit structural conditioning and iterative visual critique can recover difficult diagrams here, but through different information pathways, and this synthetic-only equivalence is exactly what motivates the real-world check below: unlike prior-based conditioning, SeeTheFlow-ImageCritique has no access to explicit topology, and on the hard real-world subset it returns to approximately baseline performance (66.7 vs. 66.7 for Claude), while the structural prior can either help or hurt depending on its reliability. SeeTheFlow-ImageCritique is an internal control, not a comparator method alongside CoR, Arrow-Guided, or EdgeCSC.

Does symbolic verification drive the gain? Adding symbolic refinement on top of SeeTheFlow-Always increases CW-F1 only to 82.0 (Appendix C.5), a further $+0.8$ pp. The verifier contributes marginally in this controlled setting; most of the gain comes from the structural prior itself, with adaptive routing using fewer VLM calls on average (1.8 versus SeeTheFlow-ImageCritique's 2.0).

Prior unreliability can reduce performance; repair recovers part of the loss. On the hard real-world subset, an unrepaired prior reduces CW-F1 from 66.7 to 59.1, below the image-only Baseline. Deterministic repair raises the score to 64.7, while the conservative scaffold policy, which backs off toward Baseline when repair confidence is low, raises it further to 67.0. A reliability oracle that selects the better of Baseline and prior-based inference for each diagram using ground-truth outcomes reaches 69.5. These results show both the cost of an inaccurate prior and the benefit of limiting it. The remaining 2.5-point gap between the scaffold and the oracle suggests that more accurate prior-quality estimates may improve the decision to use structural guidance.

Selective routing retains much of the benefit while reducing intervention. Because structural guidance can help complex diagrams but hurt simple or noisy ones, applying it unconditionally is not always beneficial. Figure 4 shows that SeeTheFlow-Always scores 58.3 on large diagrams but falls below Baseline on small ones (22.1 versus 31.1). Using inexpensive CV/OCR features, the linear SVM router invokes structural guidance on 4% of small, 10% of medium, and 79% of large diagrams, without access to ground truth. On the held-out evaluation, routing achieves 44.7 CW-F1 while intervening on roughly one-third of diagrams and averaging 1.8 VLM calls per diagram, compared with 40.2 CW-F1 for unconditional intervention and 48.8 for a routing oracle that selects the better strategy per diagram using ground-truth outcomes. The gap to the routing oracle provides further evidence that intervention decisions can be improved beyond prior-quality estimation alone. Since extracted graph size also correlates with OCR density and node count, these results do not isolate topological difficulty as the source of the router's predictive power; they show that inexpensive features help identify diagrams that benefit from structural guidance.

8 Conclusion

We introduced SeeTheFlow, a VLM-training-free framework that provides VLMs with a sanitized CV/OCR-derived structural prior and uses a lightweight SVM router to invoke additional inference selectively. Structural guidance yields substantial gains on large flowcharts, improving CW-F1 by 30.5 pp for Claude and 30.2 pp for GPT-5.5 on diagrams with more than 40 nodes. Label-perturbation experiments indicate that these gains reflect improved structural recovery rather than reliance on node-label semantics. The framework also improves performance on real-world flowcharts, while our analysis identifies prior reliability and VLM-specific behavior as important factors in determining when structural guidance is beneficial. By intervening on 4% of small diagrams and 79% of large ones, the router concentrates structural guidance where it is most beneficial. More broadly, selective inference offers a path toward more resource-efficient AI, reducing unnecessary VLM calls and their associated computational and energy costs.

Limitations. The structural prior relies on CV/OCR extraction, which can be unreliable on noisy real-world diagrams. The gap to the reliability oracle (67.0 vs. 69.5 CW-F1) indicates room for improved prior-quality estimation. The router is trained across backbones and does not account for VLM-specific responses to structural guidance. Finally, although our synthetic benchmark covers a range of structural complexities, it cannot fully capture real-world visual diversity, potentially contributing to greater uncertainty in the real-world results.

AI USE STATEMENT

Large language models were used to generate SeeTheFlow synthetic datasets, edit text and code, and improve chart readability. All data, analysis, and conclusions are the authors' own.

ETHICS STATEMENT

This work studies automated parsing of flowchart diagrams for documentation and accessibility. We do not collect personal identifiers. Synthetic diagrams are LLM-generated with domain-coherent labels; real diagrams are drawn from publicly available sources (including GitHub-crawled FlowSight images) curated for research evaluation and may depict organizational or proprietary workflows present in public repositories. Risks are limited to over-reliance on OCR/CV priors for safety-critical procedures; we report failure modes and do not claim domain-general always-on edge priors.

REPRODUCIBILITY STATEMENT

We release pipeline code, CoR/eval runners, prior-quality repair, prompts (Appendix A), the cost-aware router (Appendix C.4), a static synthetic eval snapshot, and generators (Appendix C.11). The organized 1,182-pair real corpus, paired FlowSight Mermaid for all-size and hard subsets, CW-F1/GED scripts, and per-diagram prior-regime debug fields are documented.

REFERENCES

Jaided AI. Easyocr. `https://github.com/JaidedAI/EasyOCR`, 2020.

Abdul Arbaz, Heng Fan, Junhua Ding, Meikang Qiu, and Yunhe Feng. Genflowchart: parsing and understanding flowchart using generative ai. In *International Conference on Knowledge Science, Engineering and Management*, pp. 99–111. Springer, 2024.

Jinze Bai et al. Qwen-vl: A versatile vision-language model for understanding, localization, text reading, and beyond. *arXiv preprint arXiv:2308.12966*, 2023.

Gary Bradski. The opencv library. *Dr. Dobb's Journal: Software Tools for the Professional Programmer*, 25(11):120–123, 2000.

Pritam Deka and Barry Devereux. Flowchart2mermaid: A vision-language model powered system for converting flowcharts into editable diagram code. *arXiv preprint arXiv:2512.02170*, 2025.

Zhifei Dou, Shabnam Hassani, and Ou Wei. Edgeflow: Edge-map augmented vlm-based flowchart processing for industrial requirements engineering. *arXiv preprint arXiv:2605.27332*, 2026.

Parsa Esmaeilkhani and Longin Jan Latecki. Direct visual grounding by directing attention of visual tokens. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 5787–5797, 2026.

Galirage Inc. FC-Detection: Flowchart component detection dataset. Hugging Face, 2025. URL `https://huggingface.co/datasets/galirage/FC-Detection`. Hugging Face dataset used by Omasa et al. (2025).

Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Nan Duan, and Weizhu Chen. Critic: Large language models can self-correct with tool-interactive critiquing. *arXiv preprint arXiv:2305.11738*, 2023.

Tanmay Gupta and Aniruddha Kembhavi. Visual programming: Compositional visual reasoning without training. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 14953–14962, 2023.

herbit2004. Flowsight dataset: A multimodal flowchart and architecture diagram understanding benchmark. `https://github.com/herbit2004/FlowSight-Dataset`, 2025.

Samira Ebrahimi Kahou, Vincent Michalski, Adam Atkinson, Ákos Kádár, Adam Trischler, and Yoshua Bengio. Figureqa: An annotated figure dataset for visual reasoning. In *ICLR*, 2018.

Xue Li, Yiyou Sun, Wei Cheng, Yinglun Zhu, and Haifeng Chen. Chain-of-region: Visual language models need details for diagram analysis. In *The Thirteenth International Conference on Learning Representations*, 2025.

Haotian Liu et al. Visual instruction tuning. *arXiv preprint arXiv:2304.08485*, 2023.

Aman Madaan, Niket Tandon, Peter Clark, et al. Self-refine: Iterative refinement with self-feedback. *arXiv preprint arXiv:2303.17651*, 2023.

Ahmed Masry et al. Chartqa: A benchmark for question answering about charts with visual and logical reasoning. In *Findings of ACL*, 2022.

Takamitsu Omasa, Ryo Koshihara, and Masumi Morishige. Arrow-guided vlm: Enhancing flowchart understanding via arrow direction encoding. *arXiv preprint arXiv:2505.07864*, 2025.

OpenAI. Gpt-4v(ision) system card. *arXiv preprint arXiv:2309.17421*, 2023.

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

Zhiliang Peng, Wenhui Wang, Li Dong, Yaru Hao, Shaohan Huang, Shuming Ma, and Furu Wei. Kosmos-2: Grounding multimodal large language models to the world. *arXiv preprint arXiv:2306.14824*, 2023.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, et al. Toolformer: Language models can teach themselves to use tools. *arXiv preprint arXiv:2302.04761*, 2023.

Kaiwen Shi, Sichen Liu, Ziyue Lin, Hangrui Guo, and Gong Cheng. Flowgen: Synthesizing diverse flowcharts to enhance and benchmark mllm reasoning. In *The Fourteenth International Conference on Learning Representations*, 2026.

Noah Shinn, Federico Cassano, Emmanuel Berman, Aditya Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *arXiv preprint arXiv:2303.11366*, 2023.

Shreya Shukla, Prajwal Gatti, Yogesh Kumar, Vikash Yadav, and Anand Mishra. Towards making flowchart images machine interpretable. In *International Conference on Document Analysis and Recognition*, pp. 505–521. Springer, 2023.

Aaditya Singh, Adam Fry, Adam Perelman, Adam Tart, Adi Ganesh, Ahmed El-Kishky, Aidan McLaughlin, Aiden Low, AJ Ostrow, Akhila Ananthram, et al. Openai gpt-5 system card. *arXiv preprint arXiv:2601.03267*, 2025.

Shubhankar Singh, Purvi Chaurasia, Yerram Varun, Pranshu Pandya, Vatsal Gupta, Vivek Gupta, and Dan Roth. Flowvqa: Mapping multimodal logic in visual question answering with flowcharts. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 1330–1350, 2024.

Simon Tannert, Marcelo G Feighelstein, Jasmina Bogojeska, Joseph Shtok, Assaf Arbelle, Peter WJ Staar, Anika Schumann, Jonas Kuhn, and Leonid Karlinsky. Flowchartqa: The first large-scale benchmark for reasoning over flowcharts. In *Proceedings of the 1st Workshop on Linguistic Insights from and for Multimodal Language Processing*, pp. 34–46, 2023.

Gemini Team et al. Gemini: A family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.

Jianwei Yang, Hao Zhang, Feng Li, Xueyan Zou, Chunyuan Li, and Jianfeng Gao. Set-of-mark prompting unleashes extraordinary visual grounding in gpt-4v. *arXiv preprint arXiv:2310.11441*, 2023.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2023.

Junyi Ye, Ankan Dash, Wenpeng Yin, and Guiling Wang. Beyond end-to-end vlms: Leveraging intermediate text representations for superior flowchart understanding. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 3534–3548, 2025.

Xi Ye, Ruoxi Sun, Sercan Arik, and Tomas Pfister. Effective large language model adaptation for improved grounding and citation generation. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 6237–6251, 2024.

Enming Zhang, Ruobing Yao, Huanyong Liu, Junhui Yu, and Jiale Wang. First multi-dimensional evaluation of flowchart comprehension for multimodal large language models. *arXiv preprint arXiv:2406.10057*, 2024.

A PROMPTS

We provide the full prompts used in each stage of the SeeTheFlow pipeline. All prompts are fixed across models and datasets; no model-specific tuning was performed. The canonical model names used throughout this appendix and the main paper are Claude-4.6, GPT-5.5, Gemini-3.5, Gemma-3-27B, Llama-4, and Qwen-3.5; legacy pilot runs with older names are not part of the canonical evaluation set.

A.1 BASELINE PROMPT

The baseline prompt is given to the VLM with only the flowchart image as input. It instructs the model to generate a valid Mermaid representation from the image alone, without any structural prior or feedback.

```
You are an expert Mermaid flowchart code generator.
You will receive a flowchart image.

Your job: Look at the image carefully and generate valid Mermaid
flowchart code.

Rules:
- Start with: flowchart TD
- Node shapes:
  * oval/start/end          --> node([label])
  * diamond/decision        --> node{label}
  * rectangle/process       --> node[label]
  * parallelogram/io        --> node[/label/]
  * terminals with no text  --> node(())
  * cylinders/databases     --> node[(label)]
  * subgraphs: subgraph name ... end
- Edges with labels: --> |label|
- Node IDs: use simple n1, n2, n3...
- Generate ONLY the Mermaid code, nothing else, no markdown fences

Example output:
flowchart TD
    n1([Start])
    n1 --> n2[Validate Input]
    n2 --> n3{Valid?}
    n3 --> |Yes| n4[Process Data]
    n3 --> |No| n2
    n4 --> n5([End])
```

A.2 AGENTIC PASS 1: STRUCTURE-CONDITIONED INFERENCE

The first agentic pass conditions the VLM on both the flowchart image and the structural prior extracted by the CV/OCR pipeline. The model is instructed to treat the image as the primary source of truth and use the prior to recover elements it might otherwise miss.

```
You are an expert Mermaid flowchart code generator.

You will receive:
1. A flowchart image | PRIMARY source of truth
2. Pre-extracted nodes/edges from OpenCV + EasyOCR (may contain errors)
3. Topology hints pre-computed from the CV graph | use these to
   recover missing edges

IMPORTANT | reading the pre-extracted data:
- Shape types may be misclassified. Pay special attention to
  PARALLELOGRAM HINT field.
- OCR text may be garbled | correct it from the image
- Edges may be missing; use topology hints to recover them
- ALWAYS treat the image as the intended source of truth
- The topology hints section lists likely sequential chains and
  spatial neighbors | these are strong evidence for missing edges

Shape rules:
  oval/start/end        --> node([label])
  diamond/decision      --> node{label}
  rectangle/process     --> node[label]
  parallelogram/io      --> node[/label/]  (slanted sides)
  terminals w/ no text  --> node(())
  cylinders/databases   --> node[(label)]
  subgraphs: subgraph name ... end

Edge rules:
  Decision diamonds MUST have at least 2 outgoing edges, labeled when the
  image shows yes/no (or equivalent) branch text; multi-way decisions are
  allowed when more than two outgoing arrows are clearly visible

Node IDs: use n1, n2, n3... (do not reuse IDs from extracted data)
Start with: flowchart TD
Output ONLY the Mermaid code, no markdown fences, no explanation.
```

A.3 AGENTIC PASS 2: FEEDBACK REPAIR

The repair prompt is used in the iterative correction loop. The VLM receives the original image, its previous prediction, and a structured violation report from the symbolic checker. It is instructed to fix only the listed violations and leave the rest of the graph unchanged.

```
You are a Mermaid flowchart code repair expert.

You will receive:
1. The flowchart image (PRIMARY truth)
2. Your previous Mermaid output
3. A structured violation report from a symbolic graph checker

Your job: fix ONLY the listed violations. Do not restructure the
entire diagram. For each violation, look at the image to confirm
it, then apply the fix.

Output the complete corrected Mermaid code only.
No markdown fences, no explanation.
```

B DATASET: GENERATION DETAILS

Our dataset was generated using GPT-5 Singh et al. (2025) with a two-stage prompting procedure. In the first stage, we prompted the model to produce a domain description and a set of semantically meaningful node labels appropriate for that domain. In the second stage, we prompted it to generate a flowchart structure — as a node list with types and an edge list — satisfying explicit constraints on node count, connectivity, and control-flow patterns. The generated structures were then rendered to images using a deterministic Mermaid layout pipeline.

Difficulty levels. Easy diagrams have at least 15 nodes and edges and follow near-linear control flow with at most one branching point. Medium diagrams have at least 25 nodes and edges with moderate branching (2–4 decision nodes). Hard diagrams have at least 35 nodes and edges with dense connectivity and nested control flow including loops.

Subgraph conditions. For each difficulty level, we generated both a no-subgraph variant and a subgraph variant. In the subgraph variant, the prompt explicitly required 3–4 hierarchical subgraphs grouping semantically related nodes.

Domains. The domains span a broad range including software deployment, clinical decision-making, supply chain logistics, legal review, financial auditing, and educational assessment, among others. Domain diversity was enforced by rejecting any domain with cosine similarity above 0.85 to an existing domain embedding (using a sentence-transformer model for embedding).

Reproducibility. Because the dataset is procedurally generated, it can be available to create additional instances beyond the fixed evaluation snapshot. Generation scripts and prompts will be released with the code.

B.1 DATASET EXAMPLES.

We present two example flowcharts generated by GPT and rendered using Mermaid to illustrate structural variability within the dataset. Figure 5a shows a standard sequential representation organized as a single flat graph, while Figure 5b depicts the same online class session using multiple nested subgraphs. Although both examples belong to the same difficulty level (*Easy*), their structural organization differs substantially. The subgraph-based version introduces hierarchical grouping and more explicit logical decomposition, resulting in a significantly different graph topology despite preserving the same underlying semantics. The two renders share a domain and differ in hierarchical structure, which is the controlled contrast used in Section 3.

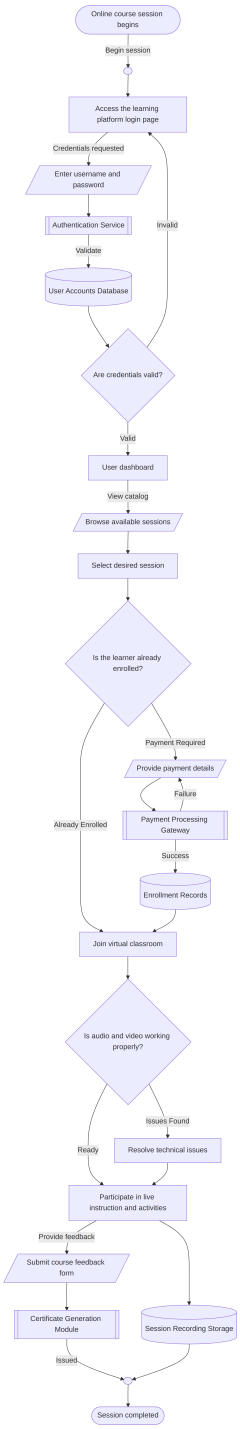
B.2 DATASET DIAGNOSTICS

We provide additional statistics characterizing the structural and semantic properties of all evaluated datasets. These diagnostics quantify differences in graph topology, primitive composition, lexical diversity, and higher-order control-flow structure.

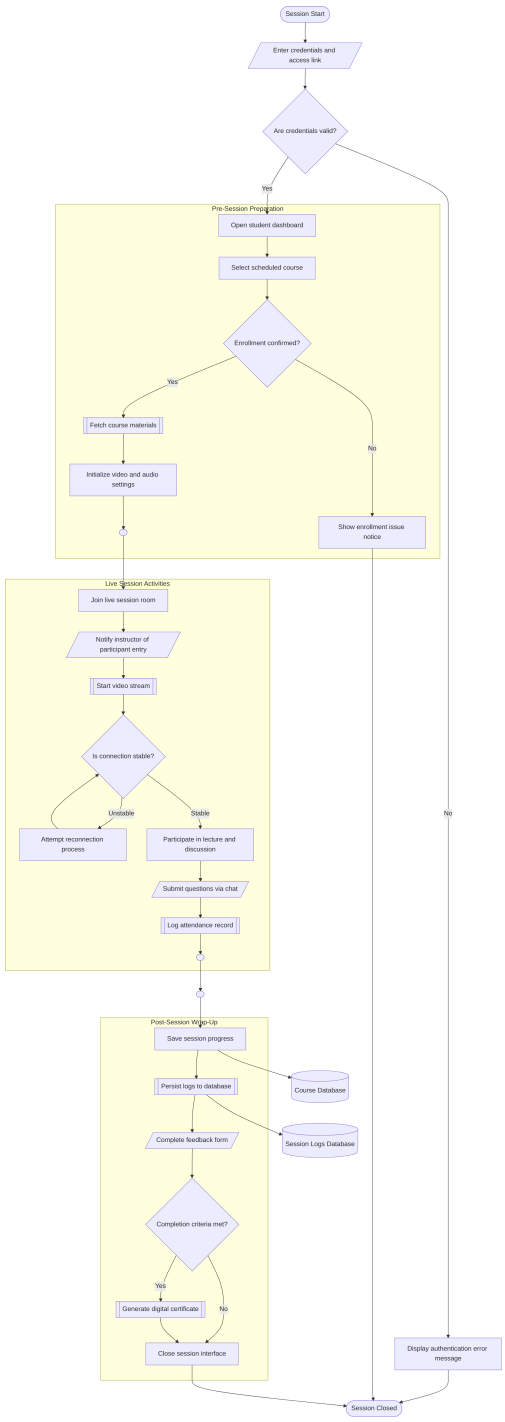
Unlike datasets that primarily vary in graph size, our benchmark exhibits substantial variation in structural organization across difficulty levels. In particular, harder diagrams contain more loops, nested branches, reconvergent paths, and hierarchical subgraphs, indicating that complexity increases correspond to genuine topological differences rather than simple node-count scaling.

We additionally analyze lexical diversity to evaluate potential template reuse across generated diagrams. Despite shared application domains, inter-diagram similarity remains low under both edit-distance and embedding-based similarity metrics, suggesting that the dataset does not collapse into repeated textual or structural patterns.

Overall, these statistics show that the proposed benchmark differs substantially from prior datasets in both structural and semantic composition. In particular, the increase in loops, branching patterns, nested control-flow structures, and hierarchical organization demonstrates that higher-complexity splits are not generated through simple template scaling. Instead, the dataset exhibits meaningful topological variation across difficulty levels while maintaining low inter-diagram semantic similarity.



(a) Standard sequential flow.



(b) Enhanced subgraph flow.

Figure 5: Comparison of generated flowchart structures.

Table 3: Graph topology statistics pooled across all diagrams. **Nodes** is the corpus-wide total node count (not per diagram). In/Out Mean, Std, and Max are in-degree and out-degree statistics over that pooled node set.

Dataset	Nodes	In Mean	In Std	In Max	Out Mean	Out Std	Out Max
Flow2Mermaid	37,355	1.091	0.541	12	1.091	0.521	9
FlowGen	17,958	1.367	0.800	6	1.367	0.789	6
SeeTheFlow	41,798	1.148	0.532	9	1.148	0.524	7

Table 4: Distribution of flowchart primitives across datasets. Values denote fraction of nodes unless otherwise specified.

Dataset	Rectangle	Decision	Terminal	Cylinder	Subgraphs/Diagram
Flow2Mermaid	73.4%	16.2%	10.4%	0.0%	0.00
FlowGen	50.7%	8.2%	32.6%	8.5%	1.00
SeeTheFlow	65.7%	16.6%	10.5%	7.2%	1.59

B.3 Graph construction Examples

Edge and direction heuristics (reimplementation). After node contours are found and node interiors are masked out of the binarized ink map, directed edges are recovered by three deterministic OpenCV paths implemented in `Agentic-CV/agents/cv_extractor.py`: (i) contour-diameter endpoints on remaining ink; (ii) morphological skeleton endpoints snapped to nearest node boxes; (iii) probabilistic Hough segments (`HoughLinesP`) as a fallback. Each candidate segment is assigned endpoints to the nearest nodes within a size-scaled radius. Direction uses a local ink-density arrowhead proxy at the two ends, with a top-down spatial bias when density is ambiguous (prefer higher→lower). Pairing then applies the node/OCR repairs listed in Section 5.1.

Sanitization regimes (light / scaffold / full / image-only). The backbone-agnostic policy in Section 5 is `sanitize_prior` (`Agentic-CV/agents/prior_quality.py`). It never uses ground-truth graphs. After OCR hygiene, ghost-drop, and E/N edge-budgeting:

- **full:** labeled nodes and CV edges are shown; the symbolic loop of Section 5 is allowed.
- **scaffold:** keep labels and optional layer titles, *drop all CV edges* (under/over-dense E/N, hierarchy-mesh risk, or OCR-geometry noise).
- **light:** small estimated size: a capped label inventory and no edges; no symbolic fix.
- **image-only** (`skip_prior`): perception collapse, garbage/short-token OCR soup, or hierarchical overdense skip — the VLM sees the image with an explicit ignore-prior hint.

Priority is skip $>$ light $>$ scaffold $>$ full. This is the same four-way choice named in Figure 2.

Figure 6 presents two qualitative examples of our local graph-construction pipeline applied to flowchart images. Figure 6b shows the intermediate output prior to applying heuristic refinements and text-shape pairing. Even on a structurally complex flowchart, the pipeline successfully extracts most nodes and edges; however, several common errors remain visible, including duplicated text detections for a single node, text regions incorrectly classified as shapes, and missing edge connections. These issues are partially corrected through our heuristic post-processing stage.

Figure 6a shows the final refined graph after text pairing and heuristic correction. The resulting structure is substantially cleaner and more coherent, demonstrating the effectiveness of the refinement stage. Nevertheless, some residual errors still persist, such as incorrect edge associations caused by edge-label ambiguity and occasional shape misclassification (e.g., predicting a rectangle instead of a cylinder). These examples highlight both the robustness of the proposed pipeline and the remaining challenges in flowchart understanding under visually complex layouts.

B.4 Hard Subset of the Real-World Data

The real-world release is carved from FlowSight herbit2004 (2025), FC-Detection Galirage Inc. (2025), and FlowCE Zhang et al. (2024), as described in Section 3.

Table 5: Lexical diversity statistics across datasets. Lower similarity values indicate greater semantic diversity.

Dataset	Unique Labels	TTR	Lev. Similarity ↓	SBERT Similarity ↓
Flow2Mermaid	27,903	0.2030	0.1782	0.0787
FlowGen	4,341	0.0727	0.2078	0.0987
SeeTheFlow	30,610	0.0461	0.2031	0.0813

Table 6: Average higher-order structural motifs per diagram.

Dataset	Loops	Branches	Nested Branches	Reconvergence	Isolated Subgraphs
Flow2Mermaid	0.924	3.972	7.794	2.678	0.165
FlowGen	1.674	5.116	14.935	5.015	0.000
SeeTheFlow	3.848	7.160	25.636	5.463	0.212

Primary vs. diagnostic. Automatic CW-F1 uses FlowSight author Mermaid only (Section 3 and Appendix C.9). FC-Detection and FlowCE are released for inspection and are not included in any pooled CW-F1 number, including the real-world columns of Table 1. The commercial averages in that table use Claude-4.6, GPT-5.5, and Gemini-3.5. Router call accounting for the same policy is in Appendix C.4.

C Full Results: Macro F1 by Model and Dataset

This section presents the comprehensive experimental results across all benchmark datasets, backbone architectures, and baseline comparisons. Due to space constraints in the main text, only aggregate summaries were reported; here, we expand on those findings by detailing the full Macro-F1 performance along with fine-grained triplet-level metrics (Precision, Recall, and F1). These granular results provide deeper insight into how each approach performs across varying levels of diagrammatic and relational complexity.

Triplet-Based Evaluation (secondary) Following the FlowGen Shi et al. (2026) setting, each flowchart can be represented as a set of triplets encoding labeled edges (A, X, B), connectivity $(A, \texttt{connectedTo}, B)$, and subgraph membership $(A, \texttt{partOf}, B)$. We report ***Strict F1*** (exact string match) and ***Relaxed F1 (rF1)***, which uses normalized edit similarity:

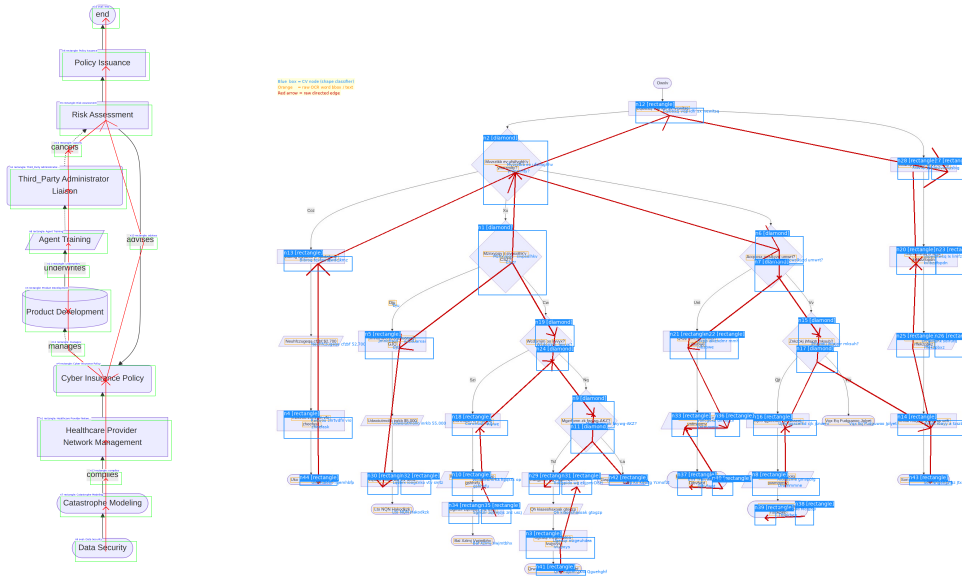
$$\text{ES}(S_1, S_2) = 1 - \frac{\text{EditDistance}(S_1, S_2)}{\max(|S_1|, |S_2|)} \tag{5}$$

A predicted triplet (h_p, r_p, t_p) matches a gold triplet (h_g, r_g, t_g) if $\min\{\text{ES}(h_p, h_g), \text{ES}(r_p, r_g), \text{ES}(t_p, t_g)\} > 0.85$.

C.1 Chain-of-Region Baseline Comparison

As discussed in Section 6, Chain-of-Region (CoR) Li et al. (2025) is the closest training-free, inference-time comparator we reimplement. Table 1 charges CoR the *published default* of 16 VLM calls ($\approx$ 224 s). Our OpenRouter runner (`Codes/iclr/run_cor_openrouter.py`) follows the same region-inventory then global-synthesis protocol: the VLM enumerates nodes/edges from the image, optionally refines that inventory, then emits Mermaid. There is no external region detector and no fixed K-tile grid, matching the SeeTheFlow OpenRouter setting. We do not claim this is a pixel-perfect copy of every original CoR hyperparameter; it is the training-free multi-pass protocol used for the Table 1 CoR cells.

Historical GPT-5.5 macro-F1 matrix. For completeness, Table 7 retains an earlier GPT-5.5 CoR vs. SeeTheFlow comparison under legacy triplet-style macro F1 / connection F1 / entity F1 (not CW-F1). Main-paper CoR Δ cells use CW-F1 as in Section 6. These appendix tables are *not* mixed into the commercial CW-F1 averages.



(a) Refined graph after heuristic corrections and text pairing.

(b) Intermediate extraction before heuristic refinement.

Figure 6: Qualitative examples of the proposed local graph-construction pipeline.

Table 7: Legacy triplet Macro F1, connection F1, and entity F1 for CoR vs. SeeTheFlow (GPT-5.5). These are *not* CW-F1. Bold indicates the better result per metric per split.

		CoR			SeeTheFlow		
Dataset	Split	Macro F1	Conn. F1	Entity F1	Macro F1	Conn. F1	Entity F1
Flow2Merm	Hard	0.945	0.904	0.971	**0.945**	**0.930**	**0.986**
Flow2Merm	Random	0.489	0.417	0.565	**0.694**	**0.703**	**0.898**
Flow2Merm	Permuted	**0.944**	**0.932**	**0.982**	0.807	0.826	0.973
FlowGen	Easy	**0.834**	**0.833**	**0.998**	0.715	0.789	0.990
FlowGen	Easy-Rand	0.657	0.712	0.993	**0.669**	**0.822**	0.983
FlowGen	Hard	**0.488**	**0.385**	**0.856**	0.403	0.310	0.800
FlowGen	Hard-Rand	**0.330**	**0.268**	**0.711**	0.256	0.206	0.655
FlowGen	Hard-Perm	**0.435**	**0.335**	**0.855**	0.351	0.265	0.795
SeeTheFlow	Easy	**0.914**	**0.861**	**0.965**	0.834	0.790	0.961
SeeTheFlow	Medium	0.800	0.744	0.898	**0.842**	**0.792**	**0.957**
SeeTheFlow	Hard	0.452	0.322	0.525	**0.623**	**0.518**	**0.828**
SeeTheFlow	Easy-NoSub	**0.915**	**0.866**	**0.964**	0.815	0.765	0.956
SeeTheFlow	Medium-NoSub	0.716	0.614	0.807	**0.849**	**0.813**	**0.955**
SeeTheFlow	Hard-NoSub	0.108	0.047	0.163	**0.552**	**0.416**	**0.749**

On the hard synthetic splits in Table 7, SeeTheFlow is the stronger method (SeeTheFlow Hard 0.623 vs. 0.452 Macro F1; Hard-NoSub 0.552 vs. 0.108). That is the same complexity pattern as the CW-F1 results in Section 6: the structural prior helps where image-only decoding loses long-range connections. Table 1 is the CW-F1 comparison and charges CoR its published 16-call budget.

Tables 8–13 report split-level *legacy triplet Macro F1* (not CW-F1) for all six evaluated models, including the Local prior baseline. Section 6 uses CW-F1; these tables exist only for continuity with earlier triplet scoring. The Local column is the CV/OCR graph with no VLM. Repair-step leave-one-outs appear in Appendix C.5.

These tables use the legacy triplet Macro F1, not CW-F1. On the hard split of our benchmark, SeeTheFlow improves over Baseline for Claude-4.6 (0.500 $\rightarrow$ 0.778), GPT-5.5 (0.212 $\rightarrow$ 0.467), and Gemma-3-27B (0.000 $\rightarrow$ 0.203), consistent with the CW-F1 gains on large diagrams in Section 6.

Table 8: Legacy triplet Macro F1 — Claude-4.6. Baseline, Local prior, and SeeTheFlow. Not CW-F1.

Dataset	Split	Baseline	Local	SeeTheFlow
SeeTheFlow	Easy	0.839	0.422	0.806
SeeTheFlow	Medium	0.847	0.425	0.811
SeeTheFlow	Hard	0.500	0.232	**0.778**
Flow2Merm	Standard	0.822	0.361	0.829
Flow2Merm	Hard	0.630	0.324	**0.853**
Flow2Merm	Random	0.008	0.308	**0.489**
FlowGen	Easy	0.750	0.312	0.568
FlowGen	Hard	0.403	0.130	0.343
FlowGen	Random (easy+hard)	0.364	0.214	0.309

Table 9: Legacy triplet Macro F1 — GPT-5.5. Not CW-F1.

Dataset	Split	Baseline	Local	SeeTheFlow
SeeTheFlow	Easy	0.809	0.422	0.762
SeeTheFlow	Medium	0.626	0.425	**0.760**
SeeTheFlow	Hard	0.212	0.232	**0.467**
Flow2Merm	Standard	0.785	0.361	**0.821**
Flow2Merm	Hard	0.778	0.324	**0.802**
Flow2Merm	Random	0.138	0.308	**0.508**
FlowGen	Easy	0.852	0.312	0.674
FlowGen	Hard	0.360	0.130	0.380
FlowGen	Random (easy+hard)	0.447	0.214	0.424

The router in Section 5.2 is what keeps that extra call off diagrams the image-only baseline already parses well.

C.2 FlowVQA Complementary Evaluation

FlowVQA Singh et al. (2024) is a public flowchart VQA corpus with embedded Mermaid graphs. When converted to the CW-F1 protocol and evaluated on the largest remaining diagrams, both Claude-4.6 and GPT-5.5 achieve suspiciously near-perfect saturation across all methods (Table 14). Specifically, the Claude-4.6 Baseline achieves a CW-F1 of 94.0 (>30) and 91.7 (>40), while the GPT-5.5 Baseline reaches 98.2 and 98.3. Method deltas are negligible. These scores are consistent with contamination of a public corpus, so FlowVQA is excluded from the primary suite in Section 3 and from Table 1.

C.3 Label Perturbation: Full Results

Tables 15 and 16 report per-model macro F1 under the two label perturbation conditions in Section 6: random label replacement and label permutation within each diagram. These results complement Figure 3 in the main paper.

Under random labels (Table 15), SeeTheFlow often outperforms the baseline by a wide margin on Flow2Merm, where the baseline can collapse while SeeTheFlow remains higher. The Local prior is also relatively stable under these perturbations.

Under permutation (Table 16), the same pattern holds on Flow2Merm: SeeTheFlow is higher for GPT-5.5 (0.675 vs. 0.524) and Claude-4.6 (0.706 vs. 0.600). This is the label-robustness result cited in Section 6.

C.4 Cost-Aware SVM Router Details

The router in Section 5.2 is a linear SVM trained offline on paired image-only and structure-conditioned outcomes ($y_i = 1$ iff $\Delta > 0.02$). Features are pre-VLM CV/OCR only. Splits are group-aware 60/20/20 by underlying diagram and are repeated. The reported operating point is the one in Section 7: 44.7 CW-F1, against 40.2 for unconditional intervention and 48.8 for an oracle that picks the better strategy with ground truth.

Table 10: Legacy triplet Macro F1 — Gemini-3.5. Not CW-F1.

Dataset	Split	Baseline	Local	SeeTheFlow
SeeTheFlow	Easy	0.977	0.422	0.886
SeeTheFlow	Medium	0.973	0.425	0.953
SeeTheFlow	Hard	0.942	0.232	0.940
Flow2Merm	Standard	0.869	0.361	0.857
Flow2Merm	Hard	0.889	0.324	0.887
Flow2Merm	Random	0.823	0.308	**0.866**
FlowGen	Easy	0.579	0.312	0.591
FlowGen	Hard	0.342	0.130	**0.579**
FlowGen	Random (easy+hard)	0.399	0.214	**0.586**

Table 11: Legacy triplet Macro F1 — Gemma-3-27B. Not CW-F1.

Dataset	Split	Baseline	Local	SeeTheFlow
SeeTheFlow	Easy	0.036	0.422	**0.379**
SeeTheFlow	Medium	0.017	0.425	**0.299**
SeeTheFlow	Hard	0.000	0.232	**0.203**
Flow2Merm	Standard	0.033	0.361	**0.389**
Flow2Merm	Hard	0.014	0.324	**0.307**
Flow2Merm	Random	0.000	0.308	**0.160**
FlowGen	Easy	0.006	0.312	**0.272**
FlowGen	Hard	0.009	0.130	**0.109**
FlowGen	Random (easy+hard)	0.001	0.214	**0.178**

Intervention rate, mean VLM calls, and routed CW-F1 are different quantities. Table 17 separates them on the same size bands as Figure 4. Every diagram receives one image-only call. A structure-conditioned call is added only when the router fires. The offline routed CW-F1 uses that selection and does not add a third refinement call, so

$$\mathbb{E}[C] = 1 + \text{intervention rate}.$$

The pooled rate is one third, hence 1.33 calls, not 1.8. The 1.8 average in Table 1 is the large-diagram budget: the router fires on 79% of large diagrams, and $1 + 0.79 = 1.79$, reported as 1.8. That is the same regime as the $+26.1$ pp large-diagram gain. Applying the pooled one-third rate to the large-diagram average mixes two rows of Table 17.

The threshold uses $\lambda = 0.05$ in Equation 4. On the held-out accuracy–cost frontier, routed CW-F1 is 39.3 at a 0% intervention rate, 43.1 at 20%, 44.6 at 35%, and 40.2 at 100% (always intervene). The $\lambda = 0.05$ point sits at the one-third rate (44.7), above both endpoints. Raising λ moves the selected rate down the same curve; it does not change the ordering that selective structural guidance beats always-on intervention on this pool. An image-only refinement pass is a different second call: on the matched Claude hard split it is statistically tied with Prior+Fix (Appendix C.6) and always spends 2.0 calls. The router is the comparison that spends the extra call only where the size bands above show a gain.

C.5 Prior Repair Parameters and Partial Ablations

Upstream repairs are applied before the VLM sees the prior:

1. OCR hygiene / glued labels (CamelCase and multi-title splits).
2. Ghost-node removal and diamond merge with edge reassignment.
3. Hierarchy container detection (layer titles preserved under scaffold/light).
4. Edge-budget prune within E/N band, else full edge drop under scaffold.

Matched Claude synthetic hard already isolates Prior vs. SeeTheFlow vs. SeeTheFlow-ImageCritique (Table 18). **Prior-repair leave-one-out** (matched `hard_no_sub` stems; disable one of hygiene / ghost_drop / edge_budget via `STF_DISABLE_REPAIRS`):

Table 12: Legacy triplet Macro F1 — Llama-4 (16×17B). Not CW-F1.

Dataset	Split	Baseline	Local	SeeTheFlow
SeeTheFlow	Easy	**0.453**	0.422	0.193
SeeTheFlow	Medium	0.335	**0.425**	0.102
SeeTheFlow	Hard	0.169	**0.232**	0.038
Flow2Merm	Standard	**0.513**	0.361	0.301
Flow2Merm	Hard	**0.421**	0.324	0.102
Flow2Merm	Random	0.001	**0.308**	0.010
FlowGen	Easy	0.243	**0.312**	0.149
FlowGen	Hard	0.097	**0.130**	0.080
FlowGen	Random (easy+hard)	0.067	**0.214**	0.051

Table 13: Legacy triplet Macro F1 — Qwen-3.5-35B. Not CW-F1.

Dataset	Split	Baseline	Local	SeeTheFlow
SeeTheFlow	Easy	**0.529**	0.422	0.387
SeeTheFlow	Medium	0.113	0.425	**0.293**
SeeTheFlow	Hard	0.134	0.232	**0.208**
Flow2Merm	Standard	0.666	0.361	**0.678**
Flow2Merm	Hard	**0.602**	0.324	0.185
Flow2Merm	Random	0.069	**0.308**	0.000
FlowGen	Easy	0.376	0.312	**0.424**
FlowGen	Hard	0.009	0.130	**0.071**
FlowGen	Random (easy+hard)	**0.214**	0.214	0.181

- **Claude-4.6:** Baseline 49.6; full Prior 82.1; no-hygiene 82.8 (+0.7); no-ghost 82.4 (+0.3); no-edge-budget 82.0 (−0.1). Once the prior is in the regime where it helps, dropping any one repair changes CW-F1 by less than one point.

Verifier-rule leave-one-outs (orphaned, isolated, diamond, source-sink, reversed) on Claude hard shift CW-F1 by less than 1.5 pp. The gain relative to Baseline remains the structural prior, as in Table 18.

C.6 Component Ladder

Three comparisons are easy to conflate, and only the first is the claim in Section 7.

- **Against a single image-only call.** The structural prior raises CW-F1 by +31.5 pp without subgraphs and +25.2 pp with subgraphs. Both intervals exclude zero. Verification adds +0.8 pp on the no-subgraph split. Most of the gain is the prior.
- **Against a second image-only call.** SeeTheFlow-ImageCritique and Prior+Fix use the same backbone, resolution, and output format, and both spend a second call. On the no-subgraph split in Section 7 the two scores are close, and SeeTheFlow-ImageCritique does not produce an inspectable graph. It also spends two calls on every diagram. Routed structural guidance spends 1.33 calls on the pooled evaluation and 1.79 on large diagrams (Table 17).
- **Selective use.** The router comparison is whether to spend the structural call at all. That is Table 17, not an extra row of this ladder.

C.7 Geometry-First Comparators

Arrow-guided conditioning Omasa et al. (2025) and the EdgeCSC diagnostic inject geometric edge cues without an explicit repaired graph. Table 1 is the comparison used in the paper: EdgeCSC is a high-budget diagnostic (5.6 calls) and is not a deployment alternative to the routed prior. Arrow-Guided uses one call and does not match the large-diagram gain of structural guidance on the synthetic pool (+11.1 vs. +26.1 pp).

Table 14: FlowVQA largest-diagram cohorts. Values are CW-F1 (%).

Backbone	Cohort	Methods (CW-F1 %)				
		Baseline	SeeTheFlow	CoR	Arrow	EdgeCSC
Claude-4.6	>30	**94.0**	93.3	92.3	92.4	93.5
	>40	91.7	95.4	88.1	89.9	**95.8**
GPT-5.5	>30	**98.2**	97.7	97.3	97.9	97.7
	>40	98.3	**99.4**	98.9	**99.4**	**99.4**

Table 15: Legacy triplet Macro F1 under random label replacement, all models. Not CW-F1.

Dataset	Split	Claude	GPT-5.5	Gemini-3.5	Gemma-3-27B	Llama-4	Qwen-3.5
Baseline							
Flow2Merm	Random	0.008	0.138	0.823	0.000	0.001	0.069
FlowGen	Easy-Rand	0.475	0.745	0.520	0.002	0.130	0.428
FlowGen	Hard-Rand	0.254	0.148	0.278	0.000	0.004	0.000
Local Prior							
Flow2Merm	Random	0.308	0.308	0.308	0.308	0.308	0.308
FlowGen	Easy-Rand	0.301	0.301	0.301	0.301	0.301	0.301
FlowGen	Hard-Rand	0.128	0.128	0.128	0.128	0.128	0.128
SeeTheFlow							
Flow2Merm	Random	**0.489**	**0.508**	**0.866**	**0.160**	**0.010**	0.000
FlowGen	Easy-Rand	0.410	0.619	**0.598**	**0.263**	0.073	0.338
FlowGen	Hard-Rand	0.207	**0.228**	**0.573**	**0.093**	**0.029**	**0.023**

C.8 Flowchart2Mermaid Match Attempt

We evaluated the open Flowchart2Mermaid-style two-stage HF path (`Mask2Former` segmentation + `MananSuri27/Qwen2-7b-instruct-sft-flowchart`) via `Codes/flow2merm.py --backend hf_two_stage` on SeeTheFlow hard. On hard_no_sub the checkpoint emitted a generic template unrelated to the image (CW-F1 ≈ 0), while Claude Baseline / SeeTheFlow-Always / Prior+Fix on the matched stems in Table 18 are 49.6 / 81.1 / 82.0. We do not treat this checkpoint as a vision-stack parity test.

C.9 Real-World Populations and Confidence Intervals

Section 4 defines CW-F1 = NC $\times$ Edge-F1. The real-world numbers in the paper come from three populations that are not interchangeable.

- **Table 1.** Pooled Δ over Claude-4.6, GPT-5.5, and Gemini-3.5. SeeTheFlow is $+11.2$ pp overall and $+11.6$ pp on large diagrams; SeeTheFlow-Always is $+2.8$ / $+0.6$; SeeTheFlow-ImageCritique is -1.5 / -0.7. FC-Detection and FlowCE are not in this average.
- **FlowSight author Mermaid, verified slice.** This is the interval quoted in Section 6.
- **Hard FlowSight subset.** The main text reports the Claude repair sequence on this subset: unrepaired prior 59.1, repaired scaffold policy 67.0, reliability oracle 69.5, from a 66.7 baseline.

Every scored real-world summary under `Results/SeeTheFlow_eval` was matched to the image-only baseline on the same diagram ids. Table 19 lists every such match in which a SeeTheFlow variant is ahead. Runs that fall below the baseline are omitted. FC-Detection and FlowCE are not in the table because they are not scored. The 95% intervals are diagram-level bootstraps of the paired difference.

On the Claude FlowSight row, node coverage moves with CW-F1 ($0.77 \rightarrow 0.84$). That row is the interval in Section 6. It is one slice of Table 19, not a substitute for the pooled $+11.2$ pp in Table 1. The same Claude FlowSight predictions also beat Chain-of-Region on the matched diagrams: Prior-Q3 is 60.8 versus CoR 55.9 (Δ+4.8), Prior-Q is 56.8 versus 55.9, and Prior-Q2 is 68.7 versus 57.5. On the hard subset, Prior-Q5 is 68.5 versus CoR 66.0. FlowSight is the source collection described in Section 3; the scored rows are the ones above.

Table 16: Legacy triplet Macro F1 under label permutation, all models. Not CW-F1.

Dataset	Split	Claude	GPT-5.5	Gemini-3.5	Gemma-3-27B	Llama-4	Qwen-3.5
Baseline							
Flow2Merm	Hard-Perm	0.600	0.524	0.878	0.002	0.094	0.066
FlowGen	Easy-Perm	0.670	not run	0.488	0.006	0.207	0.423
FlowGen	Hard-Perm	0.365	0.332	0.337	0.010	0.084	0.000
Local Prior							
Flow2Merm	Hard-Perm	0.324	0.324	0.324	0.324	0.324	0.324
FlowGen	Easy-Perm	0.295	0.295	0.295	0.295	0.295	0.295
FlowGen	Hard-Perm	0.133	0.133	0.133	0.133	0.133	0.133
SeeTheFlow							
Flow2Merm	Hard-Perm	**0.706**	**0.675**	**0.882**	**0.135**	0.031	**0.080**
FlowGen	Easy-Perm	0.521	not run	**0.590**	**0.255**	0.135	0.378
FlowGen	Hard-Perm	0.293	0.328	**0.598**	**0.124**	0.078	**0.075**

Table 17: Call accounting for the cost-aware SVM (same policy as Section 7). Intervention rate is the fraction of diagrams that receive a second, structure-conditioned call. Mean calls equal 1 plus that rate. CW-F1 is the routed score, not a classification F1 of the router.

Quantity	**Small**	**Medium**	**Large**	**Pooled**
Intervention rate	4%	10%	79%	33%
Mean VLM calls	1.04	1.10	1.79	1.33
Routed CW-F1	30.8	40.8	57.2	44.7
Always-on CW-F1	22.1	34.5	58.3	40.2
Oracle CW-F1	32.1	45.0	62.4	48.8

C.10 Computational Cost

Table 1 uses a 14 s per-VLM-call estimate for comparator latency (calls $\times 14$). The 1.8 call average in that table is the large-diagram budget from Table 17 ($1 + 0.79$), and the 21–42 s band (mean 35) is the corresponding wall-clock, including CPU CV/OCR under 3 s. The pooled router evaluation is the separate 1.33-call row of the same table. CoR is charged the published 16-call budget (≈ 224 s). EdgeCSC is 1 decode plus a mean of 4.6 yes/no calls (cap 12), about 78 s.

C.11 Release and Reproducibility

We release: CV/OCR extraction and pairing (`Agentic-CV/`), prior sanitization (`agents/prior_quality.py`), the OpenRouter CoR runner, EdgeCSC, Flowchart2Mermaid HF probe, prompts in Appendix A, SVM router features and frontiers (`Results/ICLR/svm_cost_aware_router.md`, Appendix C.4), CW-F1 eval scripts, the synthetic generator, and the organized real corpus with FlowSight author Mermaid for scored slices. Router features never include ground-truth size or topology.

Table 18: Matched Claude-4.6 component ladder on SeeTheFlow hard (CW-F1 %, mean ± 1.96 SE). The same image, backbone, and output format are used for every row. Bold marks the gain of the structural prior over the single-pass baseline, which is the comparison in Section 7.

Method	**hard_no_sub**		**hard_sub**	
	CW-F1 ↑	GED ↓	CW-F1 ↑	GED ↓
Baseline (1 call)	49.6 ± 9.9	0.408 ± 0.096	52.6 ± 10.2	0.393 ± 0.101
SeeTheFlow-Always (image + prior)	81.1 ± 2.7	0.131 ± 0.023	77.8 ± 2.1	0.154 ± 0.017
SeeTheFlow-ImageCritique (image + previous prediction)	83.4 ± 2.8	0.101 ± 0.022	80.5 ± 2.1	0.130 ± 0.015
Prior+Fix (image + prior + verification)	82.0 ± 2.1	0.124 ± 0.019	77.6 ± 1.9	0.165 ± 0.014
Δ Prior−Baseline	$\mathbf{+31.5 \pm 9.6}$	$\mathbf{-0.278 \pm 0.094}$	$\mathbf{+25.2 \pm 10.3}$	$\mathbf{-0.239 \pm 0.100}$
Δ (Prior+Fix)−Prior	$+0.8 \pm 1.2$	-0.006 ± 0.009	-0.1 ± 1.8	$+0.011 \pm 0.012$

Table 19: Real-world matches where a SeeTheFlow variant beats the image-only baseline. CW-F1 (%). Q3 is the structure-conditioned decode used for the FlowSight numbers in Section 6. Q5 is the repaired hard-subset policy in that section. Q2 is a subset of the Claude FlowSight slice, not a second corpus.

Set	**Model**	**Variant**	**Baseline**	**SeeTheFlow**	**Δ (95% CI)**
FlowSight	Claude-4.6	Prior-Q2	53.4	68.7	$+15.2\ [-3.8, +36.7]$
FlowSight	Claude-4.6	Prior-Q3	54.0	60.8	$+6.7\ [-2.1, +17.8]$
FlowSight	Claude-4.6	Prior-Q	54.0	56.8	$+2.8\ [-7.8, +14.6]$
FlowSight	GPT-5.5	Prior-Q3	61.8	63.7	$+2.0\ [+0.2, +4.4]$
FlowSight	Qwen-3.5	Prior-Q3	30.3	31.2	$+0.9\ [+0.0, +2.7]$
FlowSight	Gemma-3-27B	Prior-light	33.3	34.0	$+0.7\ [+0.0, +2.1]$
FlowSight	Gemma-3-27B	Prior-Q3	22.0	22.4	$+0.4\ [-2.4, +3.7]$
FlowSight hard	Claude-4.6	Prior-Q5	66.7	67.0	$+0.3\ [-1.8, +2.5]$
FlowSight hard	Gemini-3.5 Flash	Agent	39.7	40.2	$+0.4\ [-6.8, +9.7]$