Twin-T & TwintVQA: A Reliable Structure–Detail Separating VLM and a Comprehensive Benchmark for Chart and Table Tasks

Supplementary Material

1. Overview

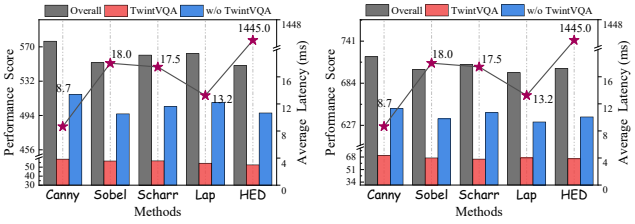
Due to space limits in the main paper, we move complementary experiments and detailed derivations to the appendix:

- Sec. 2: Additional experiments and analyses (Structure Image Extraction Experiments; Qualitative Experiments; More Ablation Experiments; More Stage 1&2 Experiments; Loss Analysis; Training Data and Settings Details).
- Sec. 3: Specification of our **TwintVQA** benchmark.
- Sec. 4: Motivation, derivations and details for the **Schurr-style** fusion.
- Sec. 5: Module-wise efficiency analysis of our models.
- Sec. 6: Discussion and analysis of failure cases and limitations.
- Sec. 7: Outline of our future work.
- Sec. 8: Summary and details of all benchmarks used in our main paper.

2. More Experiments and Analysis

2.1. Structure Image Extraction

In Stage 1 we introduce a dual-head image encoder that ingests the original image together with a structural view obtained by edge extraction method. Indeed, it is a way to isolate high-frequency content. To assess how the choice of extractor affects performance, we compare five mainstream methods in Fig.1.



(a) Twin-T-1B performance (b) Twin-T-7B performance

Figure 1. Sensitivity to structural-image extraction methods. We mimic human reading by separating high-frequency (structure) and low-frequency (detail) cues when parsing charts and tables. Bars report performance by datasets split, "w/o TwintVQA" shows overall public benchmarks scores excluding TwintVQA. The polyline shows average latency (ms/image) for each method.

Canny delivers the smallest performance drop and the best speed-accuracy trade-off, adding only ≈ 0.008s per image in inference, which is negligible relative to ≈ 0.02 s per output token. Across both our TwintVQA and the aggregate of nine public benchmarks, the score gaps among methods are modest, with Canny consistently strongest overall. Further analysis in the Appendix suggests that other methods are more sensitive to background clutter and spurious textures, they may over-extract irrelevant high-frequency patterns or miss true structural edges, weakening the subsequent separation between structure and detail and slightly degrading performance score, as shown in Fig.2.

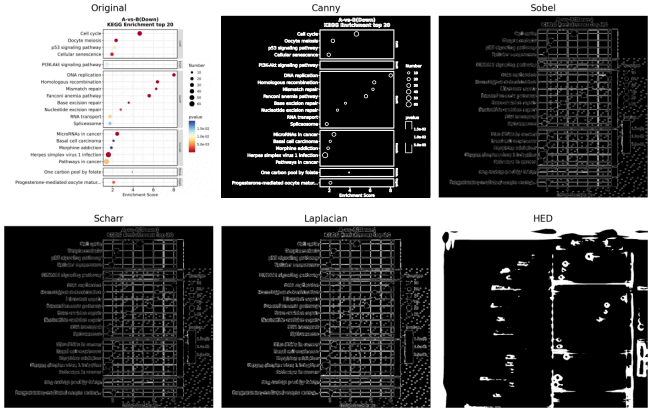


Figure 2. Visualisation of the structure image extraction methods. Zoom in for more details.

2.2. More Qualitative Experiments

In the main paper, we conduct extensive quantitative evaluations using a unified performance score to compare VLMs. Owing to space limits, this subsection presents additional qualitative case studies that demonstrate Twin-T's stronger chart-table VQA abilities. For a more complete qualitative assessment, we not only visualize the outputs of our Twin-T-1B and Twin-T-7B models, but also compare them against several strong baselines. Specifically, for each example we include other representative open-source VLMs whose scores on that example are above the overall average.

As shown in Fig.3. Our model shows clear advantages on LaTeX code generation tasks, excelling in both format fidelity and content fidelity, and consistently outperforming competing VLMs. Notably, the Twin-T-1B, despite its compact size delivers strong performance and surpasses other open-source small models in table to LaTeX conversion quality.

Twin-T also shows strong layout understanding on chart-

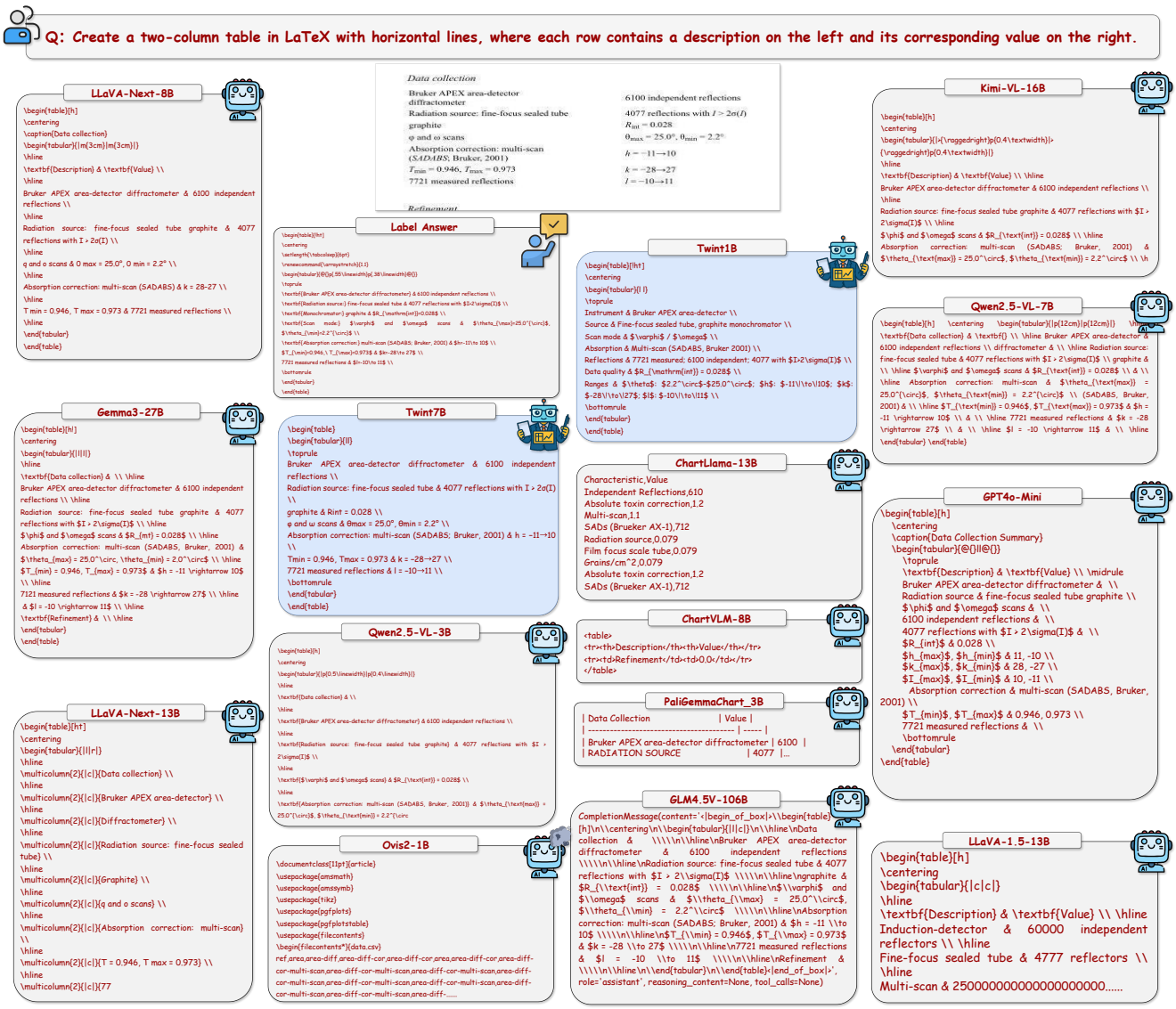


Figure 3. Qualitative visual example 1.

table VQA in Fig.4. For the question "How many words are in column 4?", both Twin-T-1B and Twin-T-7B answer "20", which matches the label answer. This question requires the model to (1) recognize that the figure is structured as five vertical columns, (2) isolate the fourth column, and (3) count how many distinct word entries appear in that column. Many competing VLMs fail on one of these steps. Some over-count (e.g., saying "30" or "25"), while others hallucinate impossible values (e.g., "100 words") or drift into long descriptions instead of giving a precise count.

In contrast, Twin-T gives a direct, correct numeric answer. More importantly, even the 1B variant behaves this way, which suggests that our dual-head encoder and MINT preference learning help the model treat a dense table as structured data rather than as unorganized text.

Our Twin-T also demonstrates strong numerical grounding and code-level reasoning in Fig.5. When asked for the correlation between `x17_TA` and `x29_TA`, both Twin-T-1B and Twin-T-7B return the exact value -0.67, which matches the correlation matrix explicitly defined in the provided Python code. In contrast, several competing VLMs either output the wrong sign (e.g., reporting 0.67 instead of -0.67), or give vague approximations rather than the precise value. This shows that Twin-T can (1) parse program-defined numeric tables, (2) bind the correct pair of variables, and (3) extract and report the correct scalar without drifting. Notably, this behavior holds even for the 1B variant, indicating that our approach captures reliable numeric reason-

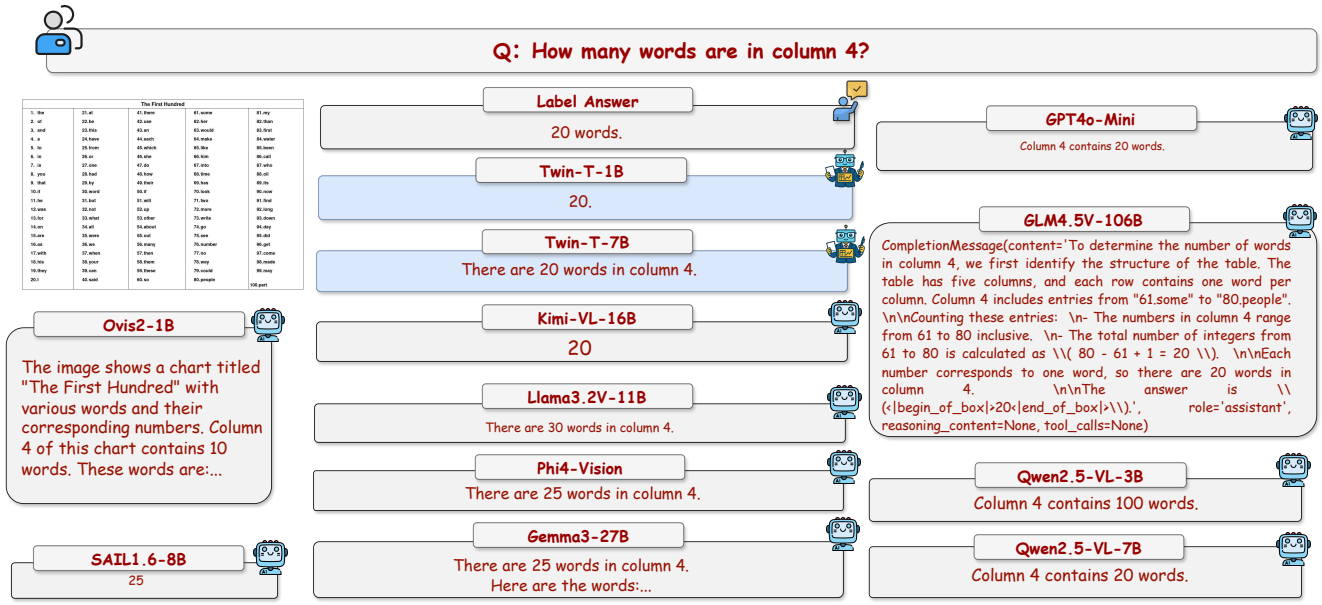


Figure 4. Qualitative visual example 2.

ing from code and is not limited to large parameter counts.

2.3. More Ablation Experiments Analysis

We conduct ablations to quantify the contribution of each module in Twin-T model. First, we ablate the two training stages, as summarized in Tab.1, where "✓" denotes inclusion and "✗" denotes exclusion. In general, removing either stage degrades performance.

Table 1. Stage ablation experiments.

Models	Stage1	Stage2	TwintVQA	w/o TwintVQA	Overall
Twin-T-1B	✓	✓	58.79	517.41	576.20
Twin-T-7B			70.20	649.74	719.94
Twin-T-1B	✓	✗	53.93 ▼4.86	475.33 ▼42.08	529.26 ▼46.94
Twin-T-7B			66.93 ▼3.27	608.12 ▼41.62	675.05 ▼44.89
Twin-T-1B	✗	✓	50.95 ▼7.84	461.87 ▼55.54	512.82 ▼63.38
Twin-T-7B			57.05 ▼13.15	615.47 ▼34.27	672.52 ▼47.42

To further pinpoint where the performance gains come from, we then ablate key components within each stage: in Stage 1, the dual-head encoder, the structure-aware gating, and the Schur-style fusion, as shown in Tab.2.

Notably, removing the dual-head encoder degenerates Stage 1 to traditional supervised fine-tuning, and causes the largest drop: Overall 529.26→502.81 (-26.45, -5.0%) for 1B and 675.05→642.28 (-32.77, -4.9%) for 7B. Removing structure-aware gating (omitting the soft gate derived from the structural embedding) yields -14.20 (-2.7%) on 1B and -18.07 (-2.7%) on 7B. Removing Schur-style fusion (replacing the Schur-style update with common summation $E_{\text{Img}}+E_{\text{Stru}}$) gives -11.74 (-2.2%) on 1B and -14.79 (-2.2%) on 7B. Trends are consistent on both TwintVQA and w/o TwintVQA subsets, confirming the dual-head design as the primary contributor, while structure-aware gating and Schur-style fusion provide stable $\sim$2–3% additional gains across scales.

Table 2. Ablation experiments of Stage 1 modules.

Models	TwintVQA	w/o TwintVQA	Overall
Twin-T-1B	53.93	475.33	529.26
↳w/o Dual-head encoder	49.21 ▼4.72	453.60 ▼21.73	502.81 ▼26.45
↳w/o Structure gating	51.12 ▼2.81	463.94 ▼11.39	515.06 ▼14.20
↳w/o Schur-style fusion	51.85 ▼2.08	465.67 ▼9.66	517.52 ▼11.74
Twin-T-7B	66.93	608.12	675.05
↳w/o Dual-head encoder	61.82 ▼5.11	580.46 ▼27.66	642.28 ▼32.77
↳w/o Structure gating	62.84 ▼4.09	594.14 ▼13.98	656.98 ▼18.07
↳w/o Schur-style fusion	61.71 ▼5.22	598.55 ▼9.57	660.26 ▼14.79

In Stage 2, we evaluated the numeric–keyword preference (Num-Key), low-entropy regularizer (Low-Entropy), and the text-vision evidence preference (Txt-Vis), as shown in Tab.3.

Table 3. Stage 2 ablations. "w/o TwintVQA": aggregate over public benchmarks excluding TwintVQA.

Models	TwintVQA	w/o TwintVQA	Overall	NK Acc (%)↑	Entropy (%)↓	Match (%) ↑
Twin-T-1B	58.79	517.41	576.20	84.20	16.80	90.60
w/o Num-Key	52.72 ▼6.07	490.44 ▼26.97	543.16 ▼33.04	81.50 ▼2.70	17.40 ▲0.60	88.30 ▼2.30
w/o Low-Entropy	55.91 ▼2.88	492.17 ▼25.24	548.08 ▼28.12	83.70 ▼0.50	24.90 ▲8.10	88.90 ▼1.70
w/o Txt-Vis	55.06 ▼3.73	503.11 ▼14.30	558.17 ▼18.03	84.00 ▼0.20	20.70 ▲3.90	82.40 ▼8.20
Twin-T-7B	70.20	649.74	719.94	90.60	13.40	95.80
w/o Num-Key	64.82 ▼5.38	627.68 ▼22.06	692.50 ▼27.44	84.70 ▼5.90	15.80 ▲2.40	94.10 ▼1.70
w/o Low-Entropy	66.79 ▼3.41	635.57 ▼14.17	702.36 ▼17.58	89.23 ▼1.37	17.20 ▲3.80	94.60 ▼1.20
w/o Txt-Vis	66.24 ▼3.96	629.41 ▼20.33	695.65 ▼24.29	88.75 ▼1.85	14.20 ▲0.80	89.80 ▼6.00

Across both 1B and 7B, the three objectives act in complementary ways and their intended effects are clearly re-

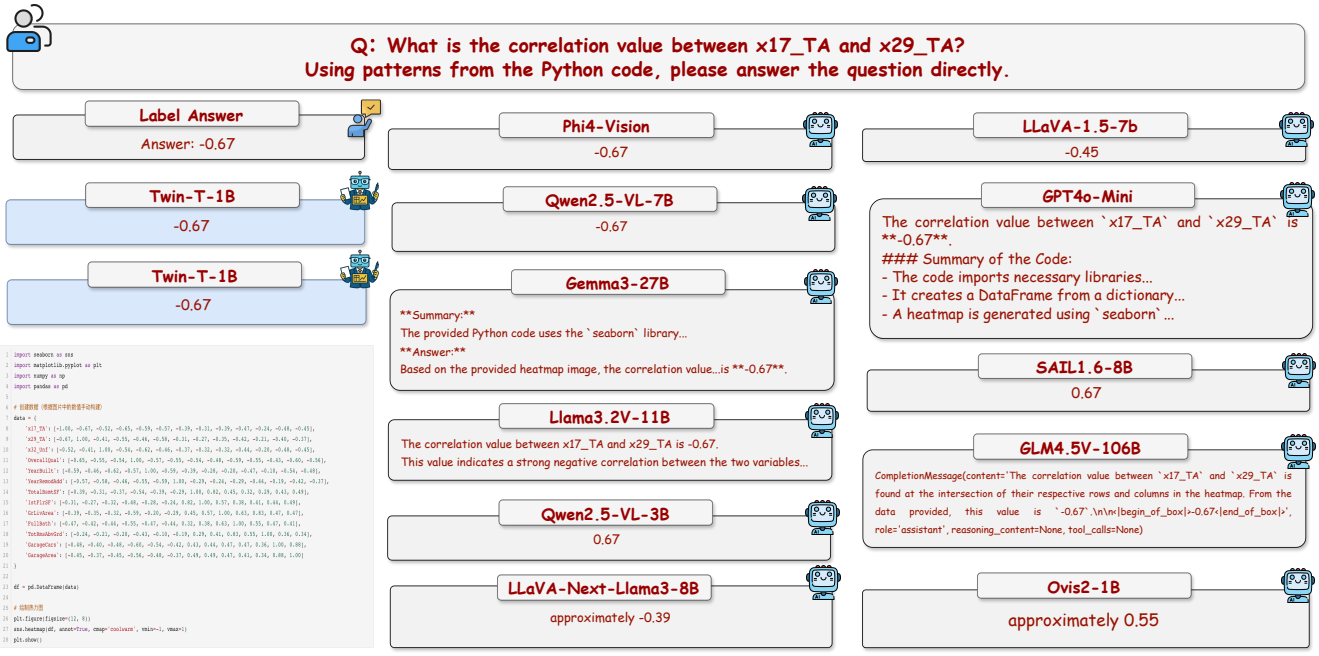


Figure 5. Qualitative visual example 3.

flected in the metrics. Num–Key preference drives the largest overall performance gains: removing it yields the biggest Overall drops (1B: -33.04; 7B: -27.44) and the sharpest decline in NK Acc (1B: -2.70%; 7B: -5.90%), confirming its role in numeric fidelity. Low-entropy regularization chiefly stabilizes number generation: ablating it spikes entropy markedly in Entropy metric (1B: $+8.10\%$; 7B: $+3.80\%$) with a moderate Overall decrease (1B: -28.12; 7B: -17.58). Text–vision evidence preference most strongly affects grounding: removing it causes the largest drop in Txt–Vis Match (1B: -8.20%; 7B: -6.00%) and a noticeable Overall decline (1B: -18.03; 7B: -24.29).

2.4. Stage 1&2 Method vs. SFT Method

We propose a two-stage training method. **Stage 1** constructs a *dual-head* visual encoder: besides the raw image, a Canny-derived structural view is fed through the same encoder. A parameter-free Schur-style module between the encoder and the connector softly gates structural directions to obtain a detail embedding, then fuses structure and detail into the final visual tokens. This stage adapts the VLM to the new dual-head pathway and fusion. **Stage 2** improves generation fidelity via preference learning on chosen vs. rejected responses.

In the main paper, Sec.4 ablates each stage and their inner modules. Here, we also include a direct supervised fine-tuning (SFT) baseline for Twin-T-1B/7B to replace Stage 1 and Stage 2. Concretely, we run two rounds of SFT: one on the Stage 1 dataset and one on the Stage 2 dataset. Note the distinction: in Tab.2, the "*w/o dual-head encoder*" employs SFT *only in Stage 1* (without a second SFT process in stage 2). As shown in Tab.4, the full *Stage 1&2* pipeline substantially outperforms SFT across performance scores.

Table 4. Stage 1&2 vs. plain supervised fine-tuning (SFT).

Model	Train	TwintVQA	w/o TwintVQA	Overall
Twin-T-1B	Stage 1&2	58.79	517.41	576.20
Twin-T-1B	SFT	51.19	472.07	523.26
Twin-T-7B	Stage 1&2	70.20	649.74	719.94
Twin-T-7B	SFT	55.95	602.52	658.47

Relative to SFT, Stage 1&2 brings consistent gains. For **1B**, it improves TwintVQA by +7.60 (+14.8%), w/o TwintVQA by +45.34 (+9.6%), and Overall by +52.94 (+10.1%). For **7B**, it improves TwintVQA by +14.25 (+25.5%), w/o TwintVQA by +47.22 (+7.8%), and Overall by +61.47 (+9.3%). These results support the necessity of *dual-head encoding* + *Schur-style fusion* (Stage 1) and *MINT* preference learning (Stage 2) beyond plain SFT.

2.5. Stage 2 Ablation Details

In the ablation section of the main paper, we present ablation experiments for Stage 2. Below, we detail the relevant experimental specifics.

NK Acc judge prompt

You are a strict numeric grader for chart/table QA. Given the question, the ground-truth answer, and a model answer, output a single scalar in [0,100] indicating how numerically correct the model answer is.
Scoring rules:
100: All numbers, units, and comparative claims match the ground truth. Wording differences are allowed.
Partial credit (0<score<100): Some numeric elements are correct but others are missing, off by the wrong unit, or slightly wrong. Minor numeric deviation is acceptable if $|\text{pred} - \text{gt}| \leq 1 \times 10^{-3}$ or relative error $\leq 1 \times 10^{-2}$.
0: The numeric facts, magnitudes, ordering, or comparisons are wrong or unsupported.
Focus ONLY on numeric correctness (values, units, comparisons like "higher/lower", percentages, ranks). Ignore politeness, style, or extra filler text.
Output format: Return *only* one number in [0,100] with no extra text.
Question: {QUESTION_TEXT}
Ground Truth Answer: {GROUND_TRUTH}
Model Answer: {PREDICTION}

Visual Match judge prompt
You are a factuality checker for chart/table QA. Given the *question*, the *chart/table image*, and the *model answer*, output a single scalar in $[0, 100]$ indicating how well the answer is grounded in the image.
Scoring rules:
100: Every claim in the answer (numbers, comparisons, trends, categories, summaries) is directly supported by the chart/table image. No hallucinated facts.
Partial credit (0<score<100): The answer is mostly supported but has minor exaggeration, missing qualifiers, or slightly off numbers.
0: The answer contradicts the image, fabricates values or trends, or addresses something unrelated.
Judge factual grounding only. Do *not* penalize wording, grammar, or style.
Output format: Return *only* one number in $[0, 100]$ with no extra text.
Question: {QUESTION_TEXT}
Image: {IMAGE}
Model Answer: {PREDICTION}

Entropy measures how confident the model is when generating numbers. For each numeric token position t in the generated answer (i.e., tokens that are digits, units, or numeric symbols), we take the model's next-token logit distribution, convert it to probabilities $p_t(v)$ over the vocabulary v, and compute the token-level entropy

$$H_t = -\sum_v p_t(v) \log p_t(v) \tag{1}$$

We then average this entropy over all numeric positions in the answer:

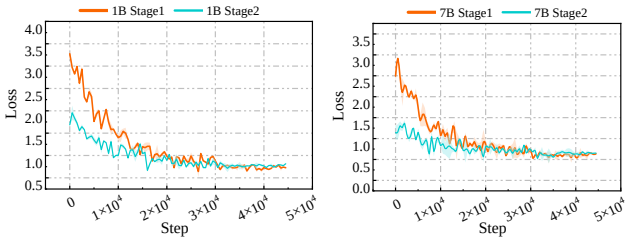
$$\text{Entropy} = \frac{1}{T_{\text{num}}} \sum_{t \in \mathcal{T}_{\text{num}}} H_t \tag{2}$$

where $\mathcal{T}_{\text{num}}$ is the set of all positions where the model is emitting a numeric token, and $T_{\text{num}} = |\mathcal{T}_{\text{num}}|$.

Lower Entropy means the model assigns sharper (lower-uncertainty) distributions to numeric tokens, which indicates more stable and more decisive numeric generation.

2.6. Loss Analysis

In order to show the training process, we plot the loss curves of **Twin-T-1B** and **Twin-T-7B**. Stage 1 employed a cross-entropy objective, while stage 2 employed our *MINT* preference-learning method. As shown in Fig.6, for both model sizes the loss decreases steadily with training steps and transitions to a flat plateau toward the end, indicating stable convergence in training process.



(a) Twin-T-1B Loss (b) Twin-T-7B Loss

Figure 6. Twin-T-1B and Twin-T-7B loss. We apply the Savitzky-Golay filter to smooth the original curves and then compute the difference between the original and smoothed curves to generate error bands.

2.7. More Training Data Details

We collected ∼40K chart–table images from public web sources. Using predefined question templates, we prompted GPT-4o to generate five QA pairs per image, yielding ∼200K QA pairs in total (details in the Appendix). Stage 1 is trained on the full ∼200K QA pairs for two passes. Stage 2 is then trained on a curated subset of ∼20K QA pairs: for each question we obtain GPT-4o chosen and rejected responses and apply preference learning. Stage 1 and Stage 2 both employed two training epochs. Below is my prompt examples:

Chosen/Rejected Response Generation Prompt
You are an expert data analyst. You are shown a chart or table image and a question about it. Your job is to produce two answers: (1) a **Chosen Answer** that is as correct and well-grounded as possible, and (2) a **Rejected Answer** that sounds fluent but contains subtle factual or numeric errors.
Question: {QUESTION_TEXT}
(Assume you can clearly read all numbers, labels, legends, and trends in the chart/table image.)
Guidelines for the Chosen Answer:
- Use only information that can be read from the chart/table.
- Read axes, headers, legends, row/column labels, and numeric values carefully.
- If comparison or calculation is needed, reason through the numbers (differences, ratios, trends).
- Use correct units (%, dollars, years, etc.).
- Be concise and directly answer the question.
- Do **not** invent values that are not visible in the chart/table.
- Do **not** say "I cannot see the image".

Guidelines for the Rejected Answer:
- The answer must look confident and natural. Do **not** say "I am not

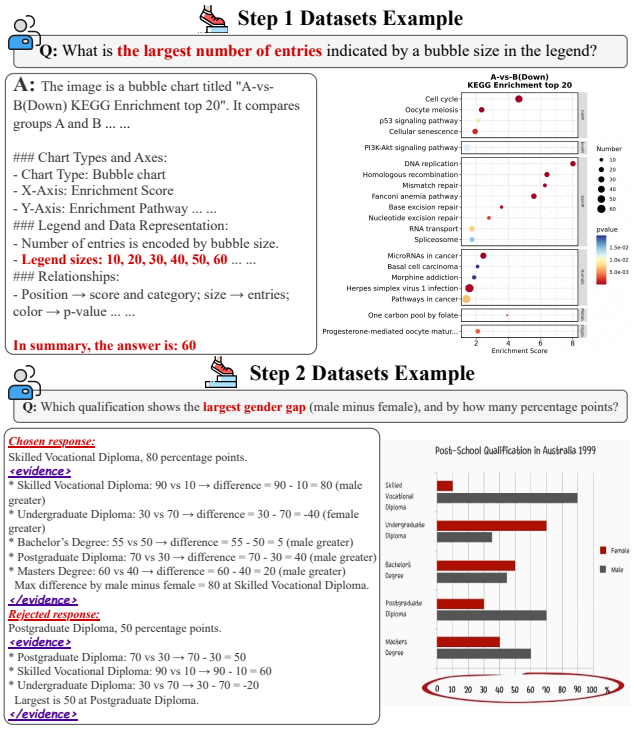


Figure 7. Our training data examples.

sure" or "I cannot see".
- Include at least one subtle factual error, for example:
  - a slightly wrong number (off by about 5 to 15%),
  - a wrong ordering or comparison (claim A > B when B is actually higher),
  - a mixed-up or omitted unit/category.
- Keep the style fluent and plausible to a casual reader. Do **not** add obvious nonsense.

Output format (very important):
Return *exactly* the following two fields in plain text, in this order, with no extra commentary:

```
Chosen: <your best correct answer>
Rejected: <your plausible but subtly incorrect answer>
```

Now generate both answers.

Our stage 1 and stage 2 training data examples are shown in Fig.7.

2.8. More Training Hyper-Parameters

We report the key optimization settings for both training stages of Twin-T.

In **Stage 1**, we train in `bfloat16` precision with a learning rate of 1×10^{-5}, for 2 epochs. We use a warmup ratio of 0.03 and AdamW as the optimizer.

In **Stage 2**, we continue tuning the same model under our preference-learning objective. The learning rate is reduced to 8×10^{-6}, we again train for 2 epochs, the warmup ratio is increased to 0.06, and we still use AdamW.

3. TwintVQA Benchmark Details

Our benchmark covers **17 chart/table types** (bar, line, pie, bubble, heatmap, radar, donut, sankey, scatter, rose, box, waterfall, stacked, candle, gantt, composite, table) and **11 tasks** across Image, LaTeX, and Python modalities. Included tasks are T2L (Table→LaTeX), C2P (Chart→Python), Analysis/Summary for each modality (IA / LA / PA; IS / LS / PS), Multiple Choice (MC), Numerical QA (NQA), and Open QA (OQA). The right panel visualizes the **question/answer length** in three bands: *Short* $(0, 256]$, *Medium* $(256, 1024]$, and *Long* $(1024, 4096]$ tokens. Our benchmark details are as follows.

3.1. TwintVQA Examples

To clarify the design of TwintVQA, we present some examples. Given the breadth of categories covered by TwintVQA, we present visualizations of representative examples, limiting the display to a few categories for readability. As shown in Fig.8, Fig.9 and Fig.10, we present representative examples from our TwintVQA benchmark across three facets:

- *Topic types* (e.g., MC, NQA, OQA, IA/IS, LA/LS, PA/PS)
- *Chart & table types* (e.g., bar, line, pie, scatter, heatmap, table, *etc.*)
- *Length types* (Short/Medium/Long, defined by total Question + Answer tokens).

3.2. Benchmark Evaluation Details

Our TwintVQA evaluation follows common practice: a deterministic GPT-4o-Mini judge maps each answer to a scalar score. In addition, we report six transparent, judge-model-agnostic metrics: EM, EM_norm, char_sim, token_f1, mc_acc, and num_acc, as defined above. Due to space limits in the main paper, we only display GPT-4o-Mini judge scores there. As shown in Tab.5 and Tab.6, we include complete results on all seven metrics in the Appendix for completeness. Chart–table QA places special emphasis on numeric fidelity, unit consistency, and evidence grounding. Several automatic string-level metrics are sensitive to formatting, tokenization, and minor wording, which may misalign with human judgment on these tasks.

Auxiliary Metrics. To avoid over-interpreting such effects, we present the full metric suite in the Appendix and use the deterministic GPT-4o-Mini judge as the main score in the paper body. Let $\hat{y}$ be the model answer and $y^\star$ the ground truth. We use a light normalizer $\mathrm{norm}(\cdot)$: lowercasing, trimming, punctuation removal, and collapsing spaces.

(1) Judge Model Prompts We score with GPT-4o-Mini and convert the $[0, 1]$ score to a percentage for reporting in

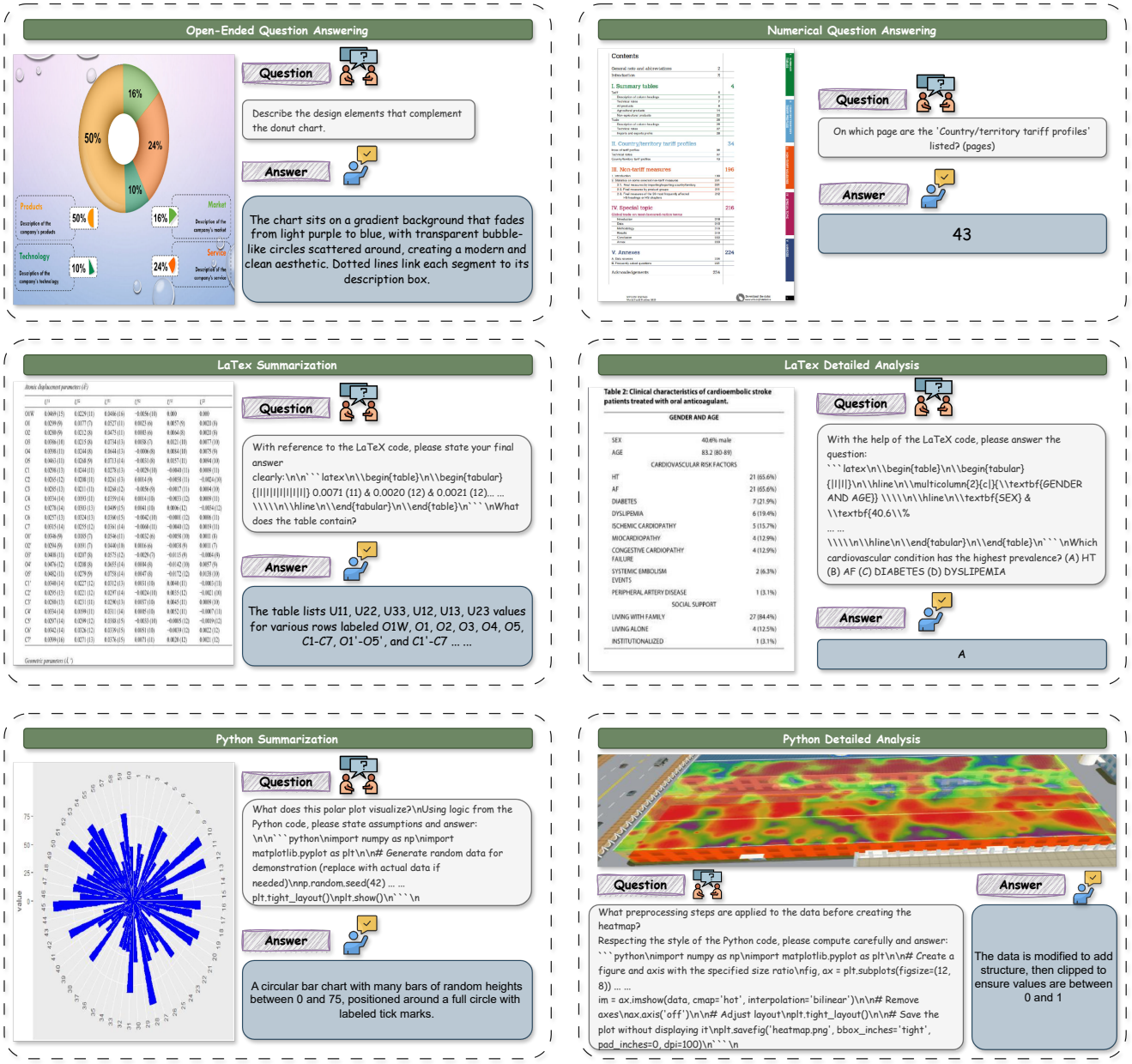


Figure 8. Topic types examples in our TwintVQA benchmarks. To prevent certain text or code from becoming excessively lengthy, we employ "... ..." to indicate ellipsis.

the main paper. Our GPT-4o-Mini score-only template is as follows:

Score-only judge prompt
You are a strict grader for chart/table VQA. Given the question, the ground-truth answer, and a model answer, return a single scalar in [0,1] indicating answer correctness and faithfulness.
Scoring guidance:

- 1.0 for exact match or semantically equivalent wording.
- Numeric tolerance: accept if $|\text{pred} - \text{gt}| \leq 1 \times 10^{-3}$ or relative error $\leq 1 \times 10^{-2}$; units must match.
- For multiple required facts, average sub-scores.
- Penalize unsupported claims or hallucinated values.
- If clearly incorrect or irrelevant, return 0.

Output format: return *only* one decimal number in [0,1] with no extra text.
Question: {QUESTION_TEXT}
Ground Truth: {GROUND_TRUTH}
Model Answer: {PREDICTION}

(2) Soft Exact Match.
We first preprocess with $\text{pre}(\cdot)$ which applies $\text{norm}(\cdot)$ and

Figure 9. Length types examples in our TwintVQA benchmarks. In our benchmark, length types are determined by the total token count of *Question*+*Answer*: **Short** $(0, 256]$, **Medium** $(256, 1024]$, and **Long** $(1024, 4096]$ tokens. To keep examples readable, overly long text or code is abbreviated with an ellipsis "... ...".

Unicode NFKC. Let $\mathrm{LCS}(a, b)$ be the length of the longest common substring of a and b. Let $\mathrm{edit\_norm}(a, b) = \frac{\mathrm{lev}(a,b)}{\max(1,|a|,|b|)}$, where lev is Levenshtein distance. We accept if any relaxed condition holds:

$$\mathrm{EM}_{\mathrm{soft}}(\hat{y}, y^\star) = \begin{cases} 1, & \text{if } \mathrm{pre}(\hat{y}) = \mathrm{pre}(y^\star), \\ 1, & \text{if } \dfrac{\mathrm{LCS}(\mathrm{pre}(\hat{y}), \mathrm{pre}(y^\star))}{\max(1, |\mathrm{pre}(y^\star)|)} \geq \tau_{\mathrm{sub}}, \\ 1, & \text{if } 1 - \mathrm{edit\_norm}(\mathrm{pre}(\hat{y}), \mathrm{pre}(y^\star)) \geq \tau_{\mathrm{char}}, \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

Default τ_{sub}=0.90, τ_{char}=0.85.

(3) Normalized Exact Match.
We compare after a stronger canonicalization that also normalizes numbers. Let $\mathrm{canon}_\kappa(\cdot)$ apply $\mathrm{pre}(\cdot)$ and then: (i) remove thousands separators, unify minus signs, (ii) convert percentages $x\%$ to decimals $x/100$, (iii) round every numeric literal to κ decimal places and strip trailing zeros. Then

$$\mathrm{EM}_{\mathrm{norm}}(\hat{y}, y^\star) = \begin{cases} 1, & \text{if } \mathrm{canon}_\kappa(\hat{y}) = \mathrm{canon}_\kappa(y^\star), \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

Default κ=3.

(4) Character Similarity.
We report a soft similarity by normalized edit distance:

$$\mathrm{char\_sim}(\hat{y}, y^\star) = 1 - \mathrm{edit\_norm}\big(\mathrm{pre}(\hat{y}),\ \mathrm{pre}(y^\star)\big). \tag{5}$$

For pass–fail use, accept if $\mathrm{char\_sim} \geq \tau_{\mathrm{char}}$ with the same default as above.

(5) Token F1.
Let $T(s)$ be whitespace tokens from $\mathrm{pre}(s)$. Optionally remove stopwords to get $C(s)$; otherwise set $C(s)$=$T(s)$.

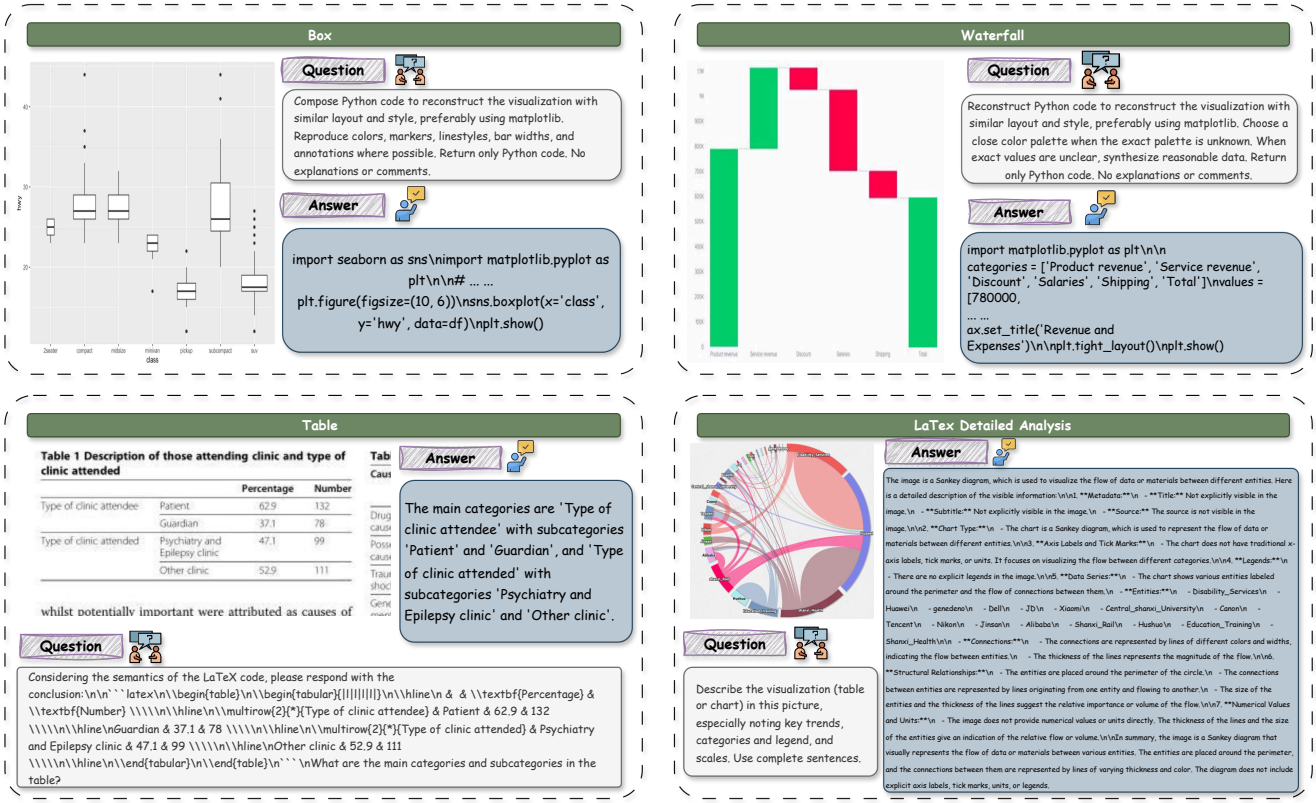


Figure 10. Chart types examples in our TwintVQA benchmarks. To keep examples readable, overly long text or code is abbreviated with an ellipsis "... ...".

Define multiset overlap by minimum term frequency.

$$\begin{aligned} \text{Prec} &= \frac{|C(\hat{y}) \cap C(y^\star)|}{\max(1, |C(\hat{y})|)}, \\ \text{Rec} &= \frac{|C(\hat{y}) \cap C(y^\star)|}{\max(1, |C(y^\star)|)}, \\ \text{token\_f1} &= \begin{cases} \dfrac{2\,\text{Prec}\,\text{Rec}}{\text{Prec}+\text{Rec}}, & \text{if Prec} + \text{Rec} > 0, \\ 0, & \text{otherwise.} \end{cases} \end{aligned} \tag{6}$$

For pass–fail use, accept if $\text{token\_f1} \geq \tau_{\text{tok}}$ with default τ_{tok}=0.80.

(6) Multiple-Choice Accuracy.

If both $\hat{y}$ and $y^\star$ contain a choice letter in $\{A, \ldots, E\}$:

$$\text{mc\_acc} = \mathbb{1}\{\text{letter}(\hat{y}) = \text{letter}(y^\star)\}, \tag{7}$$

otherwise we set $\text{mc\_acc} = 0$ (or NaN if the item is non-MC). "$\mathbb{1}$" denotes the indicator function: it returns 1 if the condition is true and 0 otherwise.

(7) Numeric-set F1.

Let $\mathcal{G} = \{g_i\}$ be all numbers parsed from $y^\star$ and $\mathcal{P} = \{p_j\}$ from $\hat{y}$ (percentages converted to $[0, 1]$). We form a one-to-one matching under dual tolerance:

$$|g_i - p_j| \leq \varepsilon_{\text{abs}} \quad \text{or} \quad \frac{|g_i - p_j|}{|g_i| + 10^{-12}} \leq \varepsilon_{\text{rel}},$$

with $\varepsilon_{\text{abs}} = 10^{-3}$ and $\varepsilon_{\text{rel}} = 10^{-2}$. Let m be the number of matched pairs, then:

$$\begin{aligned} \text{Prec} &= \frac{m}{\max(1, |\mathcal{P}|)}, \quad \text{Rec} = \frac{m}{\max(1, |\mathcal{G}|)}, \\ \text{num\_acc} &= \frac{2\,\text{Prec} \cdot \text{Rec}}{\text{Prec} + \text{Rec}}. \end{aligned} \tag{8}$$

4. Schur-Style Fusion

In Stage 1, we introduce a dual-head image encoder that splits the raw image embedding E_{Img} into a structural embedding E_{Stru} and a detail-biased embedding E_{Det}, mirroring how humans read charts and tables. This improves performance but raises a key question: how can we decouple the raw embedding in a principled way? Our idea is to estimate the structural direction in the embedding space and subtract that direction from the raw embedding; the residual is then the detail-biased component. More details are as follows:

Table 5. Twin-T-7B performance. Evaluation metrics: "JudgeScore" = GPT-4o-Mini score; "ExactMatch" = Soft Exact Match; "Norm-EM" = Normalized Exact Match; "CharSim" = Character Similarity; "Token-F1" = Token F1 value; "MC-Acc" = Multiple-Choice Accuracy; "Num-F1" = Numeric-set F1. These automatic string-level metrics are sensitive to formatting, tokenization, and minor wording, which may misalign with GPT-4o-Mini judgment on these tasks.

Key	**JudgeScore**	**ExactMatch**	**Norm-EM**	**CharSim**	**Token-F1**	**MC-Acc**	**Num-F1**
Task Types							
Chart ->Python	18.61	2.65	2.37	32.01	48.84	38.20	41.49
Multiple Choice	91.36	91.36	91.36	91.36	91.36	91.36	2.59
Image Analysis	73.70	1.26	1.58	11.19	44.69	98.85	43.23
Image Summary	54.38	1.37	1.37	38.37	39.31	73.61	41.46
Latex Analysis	91.69	72.78	74.05	87.27	86.07	85.94	79.97
Latex Summary	71.88	4.76	4.76	56.60	55.10	22.22	30.86
Numerical QA	76.04	56.10	56.91	71.19	59.45	2.12	57.99
Open QA	66.49	27.62	31.38	58.26	55.29	62.03	54.60
Python Analysis	90.58	57.75	58.45	82.50	81.17	92.86	73.74
Python Summary	82.07	6.90	6.90	63.78	63.36	81.82	65.93
Table ->Latex	83.57	1.60	3.20	80.41	90.39	62.58	92.96
Chart-Table Types							
Bar	78.90	36.10	36.59	60.81	61.71	80.85	59.77
Box	67.07	29.60	30.60	51.98	57.58	80.73	44.05
Bubble	71.85	30.60	31.15	54.47	59.81	85.00	50.40
Candle	68.39	28.75	32.50	50.97	62.98	82.35	37.24
Composite	70.10	22.58	25.81	50.79	52.88	81.82	43.76
Donut	76.79	36.21	37.07	56.64	64.18	89.00	58.27
Gantt	62.45	30.85	31.51	52.53	50.01	82.55	45.48
Heatmap	67.62	31.15	33.77	52.57	57.92	88.43	46.89
line	75.17	29.47	30.00	54.38	57.33	89.19	55.82
Pie	74.81	34.55	36.65	59.07	63.47	85.71	58.79
Radar	68.85	31.76	32.81	52.93	58.13	81.87	47.85
Rose	57.44	25.88	27.06	47.36	52.61	77.24	39.75
Sankey	61.62	28.10	28.47	47.21	52.43	75.54	45.76
Scatter	67.68	27.27	28.03	51.69	58.50	86.90	52.50
Stacked	66.80	29.63	29.63	53.10	57.80	87.10	47.33
Table	77.69	31.80	33.68	68.44	71.45	74.42	71.13
Waterfall	69.55	35.71	37.30	58.58	58.17	87.50	48.12
Length Types							
LongLenQA	61.16	17.73	17.73	45.97	56.95	71.54	52.40
MediumLenQA	69.70	10.90	11.37	39.21	59.87	88.73	54.71
ShortLenQA	70.91	39.20	40.87	62.90	60.41	78.87	53.40
All	70.20	31.07	32.36	56.14	60.13	82.06	53.80

4.1. Schur-Style Fusion Details

Token-wise setup and notations. We index tokens by $[b, t]$. For brevity, write

$$E = E_{\text{Img}}[b,t], \quad e = E_{\text{Stru}}[b,t],$$
$$u = \frac{e}{\|e\|_2 + \varepsilon}, \quad m = m_{\text{Stru}}[b,t],$$

with $\varepsilon > 0$ for numerical safety. Let $\lambda > 0$ be a prior strength and define

$$\gamma = \frac{\|e\|_2^2}{\lambda + \|e\|_2^2}, \qquad \alpha = \gamma\, m^2 \in [0, 1).$$

Coupled quadratic energy (where the Hessian and Schur come from). We couple a main variable x (image branch) and a structural variable y (structure branch) via the convex quadratic

$$\mathcal{E}(x,y) = \tfrac{1}{2}\|x - E\|_2^2 + \tfrac{1}{2}\, y^\top (\lambda I + ee^\top)\, y - \, m\,(u^\top x)\,(e^\top y). \tag{9}$$

The first-order optimality (normal equations) reads

$$\nabla_x \mathcal{E} = 0, \quad \nabla_y \mathcal{E} = 0 \quad \Longleftrightarrow \quad K \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} E \\ 0 \end{bmatrix},$$

whose *big Hessian matrix* is the PD block matrix

$$K \;=\; \begin{bmatrix} \frac{\partial^2 \mathcal{E}}{\partial x\, \partial x^\top} & \frac{\partial^2 \mathcal{E}}{\partial x\, \partial y^\top} \\ \frac{\partial^2 \mathcal{E}}{\partial y\, \partial x^\top} & \frac{\partial^2 \mathcal{E}}{\partial y\, \partial y^\top} \end{bmatrix} = \begin{bmatrix} I & -m\, ue^\top \\ -m\, eu^\top & \lambda I + ee^\top \end{bmatrix}. \tag{10}$$

Block elimination and the Schur complement (principle). Block Gaussian elimination eliminates y from

$$\begin{cases} Ax + By = E, \\ B^\top x + Cy = 0, \end{cases} \qquad A = I,\ B = -m\, ue^\top,\ C = \lambda I + ee^\top,$$

Table 6. Twin-T-1B performance. Evaluation metrics: "JudgeScore" = GPT-4o-Mini score; "ExactMatch" = Soft Exact Match; "Norm-EM" = Normalized Exact Match; "CharSim" = Character Similarity; "Token-F1" = Token F1 value; "MC-Acc" = Multiple-Choice Accuracy; "Num-F1" = Numeric-set F1. These automatic string-level metrics are sensitive to formatting, tokenization, and minor wording, which may misalign with GPT-4o-Mini judgment on these tasks.

Key	**Judge Score**	**ExactMatch**	**Norm-EM**	**CharSim**	**Token-F1**	**MC-Acc**	**Num-F1**
Task Types							
Chart ->Python	6.05	2.17	1.98	19.03	32.30	21.35	26.68
Multiple Choice	74.85	74.85	74.85	74.85	74.85	74.85	2.33
Image Analysis	60.88	1.76	1.44	9.40	39.91	98.71	39.39
Image Summary	43.16	0.91	1.07	35.50	36.16	77.84	36.61
Latex Analysis	67.53	44.30	46.84	69.00	64.28	65.62	46.27
Latex Summary	67.19	2.38	2.38	52.91	51.06	33.33	34.54
Numerical QA	68.73	44.07	45.85	61.89	49.17	1.86	46.45
Open QA	61.21	25.67	29.43	55.78	51.90	63.29	51.39
Python Analysis	70.15	35.21	37.32	62.00	56.91	75.71	45.20
Python Summary	72.02	6.90	6.90	49.65	48.08	84.85	38.56
Table ->Latex	57.94	0.80	1.20	65.59	80.95	52.15	81.49
Chart-Table Types							
Bar	62.62	28.29	29.27	51.59	49.34	74.47	46.94
Box	59.19	24.13	25.62	45.83	51.06	75.69	38.82
Bubble	60.87	25.14	26.23	45.97	50.88	80.00	42.69
Candle	59.12	21.25	25.00	44.35	52.51	82.35	32.92
Composite	58.91	17.74	17.74	45.89	46.86	81.82	38.74
Donut	67.21	29.02	31.90	50.35	56.56	84.50	52.81
Gantt	50.65	24.95	25.16	43.68	41.04	72.17	37.75
Heatmap	61.00	28.76	30.72	47.01	50.60	86.11	39.39
line	63.81	21.05	22.11	45.54	49.06	84.68	44.85
Pie	63.51	29.84	32.46	53.87	58.01	85.71	52.78
Radar	61.75	28.87	29.66	48.38	51.67	79.27	43.28
Rose	49.08	20.39	21.96	40.25	43.06	73.79	32.15
Sankey	50.14	23.36	24.45	42.45	45.61	69.78	39.15
Scatter	58.56	21.97	22.73	46.19	52.03	80.69	44.77
Stacked	57.04	28.70	29.63	48.72	55.66	85.48	43.62
Table	59.71	23.12	24.79	56.68	59.44	64.69	58.02
Waterfall	56.02	27.78	30.16	51.22	48.41	79.69	36.89
Length Types							
LongLenQA	41.40	9.85	9.85	31.83	40.97	62.60	37.02
MediumLenQA	54.30	7.03	7.50	30.74	49.45	84.35	45.41
ShortLenQA	61.45	32.69	34.59	56.12	53.00	72.85	46.17
All	58.79	25.18	26.63	48.62	51.60	76.48	45.40

by solving $y = C^{-1}(-B^\top x)$ and substituting into the first block:

$$\underbrace{\left(A - BC^{-1}B^\top\right)}_{S} x = E.$$

The matrix $S = A - B\,C^{-1}B^\top$ is the *Schur complement of C in K*, i.e., the *effective operator on the main block* after eliminating the structural block.

Sherman–Morrison inversion of the structural block. Since $C = \lambda I + ee^\top$ is a rank–1 update of λI, Sherman–Morrison gives

$$C^{-1} = (\lambda I + ee^\top)^{-1} = \frac{1}{\lambda}\left(I - \frac{ee^\top}{\lambda + \|e\|_2^2}\right). \quad (11)$$

$$e^\top C^{-1} e = \frac{1}{\lambda}\left(\|e\|_2^2 - \frac{\|e\|_2^4}{\lambda + \|e\|_2^2}\right) = \frac{\|e\|_2^2}{\lambda + \|e\|_2^2} = \gamma.$$

Step-by-step computation of $BC^{-1}B^\top$. With $B = -m\,ue^\top$ and $B^\top = -m\,eu^\top$,

$$BC^{-1}B^\top = m^2\,u\,\underbrace{(e^\top C^{-1} e)}_{\gamma}\,u^\top = \gamma\,m^2\,uu^\top.$$

Therefore the *effective operator* on the main block is

$$S \;=\; A - BC^{-1}B^\top \;=\; I - \gamma\,m^2\,uu^\top. \quad (12)$$

This is a *rank–1, token-wise soft projection subtraction* along the structural axis u.

De-biased (detail-biased) main component and fusion. Applied to the observed main embedding E,

$$E^{\text{shur}}_{\text{Img}}[b,t] \;=\; S\,E \;=\; E \;-\; \alpha\,(E^\top u)\,u, \quad \alpha = \gamma m^2. \quad (13)$$

We define the *detail embedding* as $E_{\text{Det}}[b,t] = E^{\text{shur}}_{\text{Img}}[b,t]$ and fuse it with the structural branch using the gates

$$E_{\text{fuse}}[b,t] = m_{\text{non}}[b,t]\; E_{\text{Det}}[b,t] + m_{\text{Stru}}[b,t]\; E_{\text{Stru}}[b,t]$$
$$m_{\text{non}}[b,t] = 1 - m_{\text{Stru}}[b,t].$$

Positive-definiteness (why PD is required and holds). A classical Schur/Sylvester criterion states that for $K = \begin{bmatrix} A & B \\ B^\top & C \end{bmatrix}$ with C invertible,

$$K \succ 0 \iff \big(C \succ 0 \text{ and } S = A - BC^{-1}B^\top \succ 0\big).$$

In our construction, $C = \lambda I + ee^\top \succ 0$ for any $\lambda > 0$. Moreover $S = I - \alpha uu^\top$ has eigenvalues 1 (multiplicity $\dim -1$) and $1-\alpha$ along u, so $S \succ 0$ when $\alpha \in [0,1)$. Therefore $K \succ 0$. PD ensures: (1) uniqueness and numerical stability; (2) a well-defined elimination of y; (3) clean backpropagation through both branches.

Block inverse (completeness). When C is invertible,

$$K^{-1} = \begin{bmatrix} S^{-1} & -S^{-1}BC^{-1} \\ -C^{-1}B^\top S^{-1} & C^{-1} + C^{-1}B^\top S^{-1}BC^{-1} \end{bmatrix},$$
$$S = A - BC^{-1}B^\top.$$

How we apply these formulas in Stage 1:

- Compute the structural norm $s[b,t] = \|E_{\text{Stru}}[b,t]\|_2$ and gate $m_{\text{Stru}}[b,t] = \sigma\big(\alpha_0(s[b,t]-\tau)\big)$, with temperature α_0 and soft threshold τ; set $m_{\text{non}} = 1 - m_{\text{Stru}}$.
- Form the unit direction $u[b,t] = E_{\text{Stru}}[b,t]/(\|E_{\text{Stru}}[b,t]\|_2 + \varepsilon)$ and the retention factor $\gamma[b,t] = \|E_{\text{Stru}}[b,t]\|_2^2/(\lambda + \|E_{\text{Stru}}[b,t]\|_2^2)$.
- Build the Schur operator $S[b,t] = I - \gamma[b,t]\, m_{\text{Stru}}[b,t]^2\, u[b,t]u[b,t]^\top$ and obtain $E^{\text{shur}}_{\text{Img}}[b,t] = S[b,t]\, E_{\text{Img}}[b,t]$ as in Equ.(13).
- Define $E_{\text{Det}}[b,t] = E^{\text{shur}}_{\text{Img}}[b,t]$ and fuse with $E_{\text{Stru}}[b,t]$: $E_{\text{fuse}}[b,t] = m_{\text{non}}[b,t]E_{\text{Det}}[b,t] + m_{\text{Stru}}[b,t]E_{\text{Stru}}[b,t]$.

Why this is intuitive: Humans typically read charts by *fixing the layout first* (axes, grids, headers) and then *focusing on details* (ticks, legends, colors, local variations). The direction u captures the layout axis inferred from the structural branch e. The Schur operator $S = I - \alpha uu^\top$ therefore performs a *controlled, rank–1 reduction only along the layout axis*, with strength $\alpha = \gamma m^2$ adapting to structure magnitude ($\|e\|$) and trust (m). All directions orthogonal to the layout are preserved, so the residual $E - \alpha(E^\top u)u$ concentrates detail cues. This yields a principled, differentiable, and numerically stable *detail-biased* embedding that we inject into the language model.

4.2. Embedding Fusion vs. Pixel Fusion in Stage 1

In Stage 1, we feed the raw image and its Canny-based structural image through the same image encoder to obtain an image embedding E_{Img} and a structural embedding E_{Stru}. We then apply a soft structure-aware gate to suppress noise and fuse the (de-biased) detail branch with the structure branch in the embedding space. **A natural question is whether performing gating/decoupling at the pixel level before the encoder could achieve comparable results.** We therefore conduct experiments, as shown in Tab.7.

Table 7. Embedding fusion vs. pixel fusion in Stage 1.

Model	Fusion Type	TwintVQA	w/o TwintVQA	Overall
Twin-T-1B	Embedding fusion	58.79	517.409	576.20
Twin-T-1B	Pixel fusion	51.32	453.96	505.28
Twin-T-7B	Embedding fusion	70.20	649.74	719.94
Twin-T-7B	Pixel fusion	62.18	650.74	651.67

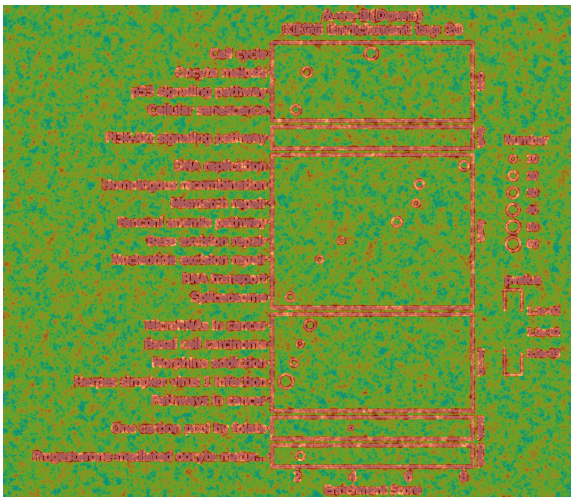
The results show that pixel-level fusion is inferior. Pixel-space interventions substantially alter the original image distribution, to which pretrained encoders are typically adapted, thereby degrading downstream performance. In contrast, the embedding space is high-dimensional and semantic, offering sparser, more robust representations that better tolerate separation and recombination of structure and detail. Under the same dual-head design, embedding-level fusion yields more stable and higher scores than pixel-level fusion.

4.3. Soft Gating Comparisons

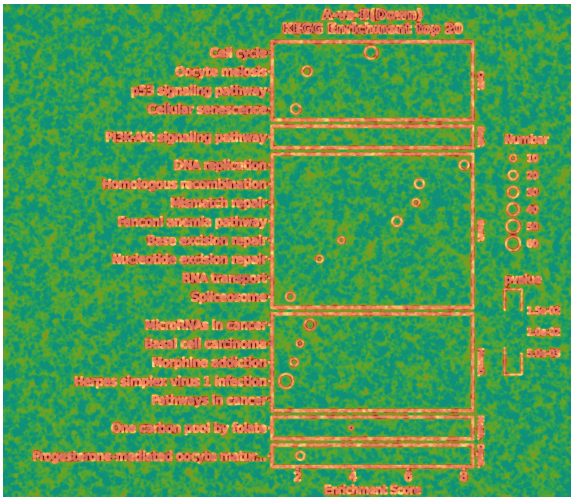
To assess where the model allocates capacity in the image encoder, we train the same backbone twice under identical settings, once *without* soft gating and once *with* soft gating, as shown in Fig.11. After training, we take the last-layer hidden states of the image encoder and compute a per-cell activation on a fine grid by aggregating hidden states within each cell using the mean l_2 norm, followed by min–max normalization within the image to obtain a relative activation map, where red indicates higher relative embedding magnitude and blue indicates lower.

In the no-gate case, high activation spreads beyond true chart structure and many red cells appear over large background and text-free regions, suggesting responses to local noise and weak edges rather than consistent structure. With the structural soft gate, red cells concentrate along borders, axes, tick marks, and other layout primitives while most background cells shift toward blue or green; activation becomes contiguous along true structure and sparser elsewhere. Comparing last-layer hidden states across the two independently trained models shows that soft gating improves spatial concentration of embedding magnitude on structural regions and reduces noise in non-structural areas.

We adopt Schur-style fusion because its element-wise interaction acts as a soft compatibility filter between the structure and detail branches. It strengthens dimensions supported by both streams while suppressing inconsistent signals, which is important for aligning local numbers with global axes or table layouts. To verify this choice, we ab-



(a) No soft gating. Relative embedding magnitude spreads into background, many red cells appear off structure.



(b) With soft gating. Red activations concentrate along borders, axes, and tick marks, while background is largely suppressed.

Figure 11. Visualization of the structural soft gate. We train two models with the same backbone and settings, one without soft gating and one with soft gating. After training, we take the last-layer hidden states of the image encoder, compute per-token embedding activation on a fine grid, and apply minmax normalization within each image. In the heatmap, red denotes higher relative embedding magnitude and blue denotes lower. Compared with the no-gate model, soft gating concentrates activations on structural elements and reduces background noise.

late several fusion operators between the structure and detail branches, including **Add**, **Concat+MLP**, and **Gated-Add**. As shown in Tab. 8, Schur-style fusion consistently performs best on both Twin-T-1B and Twin-T-7B.

Table 8. Fusion ablation on TwintVQA.

Fusion	Twin-T-1B ↑	Twin-T-7B ↑
Add	57.80	69.40
Concat + MLP	57.10	68.90
Gated-Add	58.05	69.70
Schur (ours)	**58.79**	**70.20**

5. Efficiency Analysis

We propose a two-stage training method. In this section, we analyze the computational overhead introduced by our newly added modules. All tests were conducted on NVIDIA A800 GPUs and Intel Xeon Platinum 8358P CPUs, with results averaged over 1,000 inference runs.

5.1. Canny Efficiency

In Stage 1, relative to the base model we add a Canny–based structural view. As shown in Fig.1, the additional cost is only $8.7\,\mathrm{ms}$ ($\approx 0.0087\,\mathrm{s}$) per image, with $+0.131$ GFLOPs and $+0.080$ GMACs. Compared with a decoding speed of $0.01\,\mathrm{s/output-token}$, this overhead is negligible. It is worth noting that here, Canny uses CPU computation, operating independently and unaffected by the size of Twin-T 1B/7B.

5.2. Schur-Style Fusion Efficiency

In **Stage 1**, to obtain the fused image embedding we insert a Schur-style fusion block between the image encoder outputs and the connector. This block is training-free (no learnable parameters). We measure its runtime on Twin-T-1B/7B, as shown in Tab.9.

Table 9. Stage 1 Schur-style fusion runtime.

Model	**Fusion (s)**	**Eqv. Tokens (speed)**
Twin-T-1B	0.07	∼4 (0.017 s/token)
Twin-T-7B	0.48	∼18 (0.026 s/token)

On **Twin-T-1B**, due to the smaller hidden dimension, the fusion takes only **0.07 s**. With a decoding speed of **0.017 s/token**, this is about **∼4 tokens**, which is practically negligible.

On **Twin-T-7B**, the fusion takes about **0.48 s** per VQA round. Importantly, it is executed *once* per round and does not affect the subsequent output-token latency. Given a decoding speed of **0.026 s/token**, this cost is roughly equivalent to generating **∼18 tokens**.

5.3. Stage 2 Training Efficiency

We compare our Stage 2 preference learning (MINT) with Direct Preference Optimization (DPO) under identical settings on 20K VQA pairs. MINT adds no extra modules. The average memory overhead is only $+418$ MB and train-

ing time increases by $+1.8\%$. As shown in Tab.10, MINT consistently outperforms DPO:

Table 10. Stage 2 (MINT) vs. DPO on 20k VQA pairs.

Model	Training	TwintVQA	w/o TwintVQA	Overall
Twin-T-1B	Stage 2 (MINT)	58.79	517.41	576.20
	DPO	52.03	490.69	542.72
Twin-T-7B	Stage 2 (MINT)	70.20	649.74	719.94
	DPO	65.78	610.74	676.52

For **Twin-T-1B**, $+6.76$ on TwintVQA ($+13.0\%$), $+26.72$ w/o TwintVQA ($+5.45\%$), and $+33.48$ Overall ($+6.17\%$); for **Twin-T-7B**, $+4.42$ on TwintVQA ($+6.72\%$), $+39.00$ w/o TwintVQA ($+6.39\%$), and $+43.42$ Overall ($+6.42\%$). Given the small resource overheads, these gains are well within a practical budget.

6. Failure Cases and Analysis

We provide a set of representative failure cases and analyze them in detail to more fully assess our model's performance.

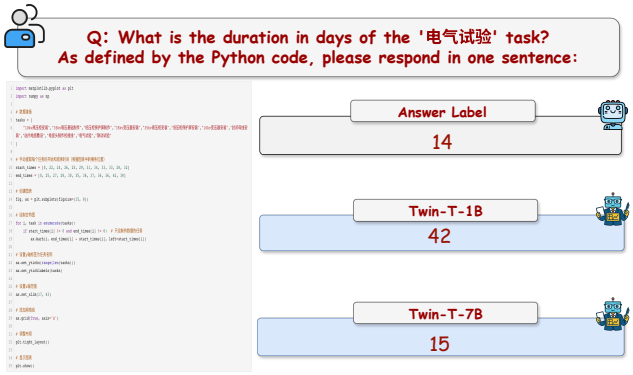


Figure 12. Twin-T VLM failure case 1

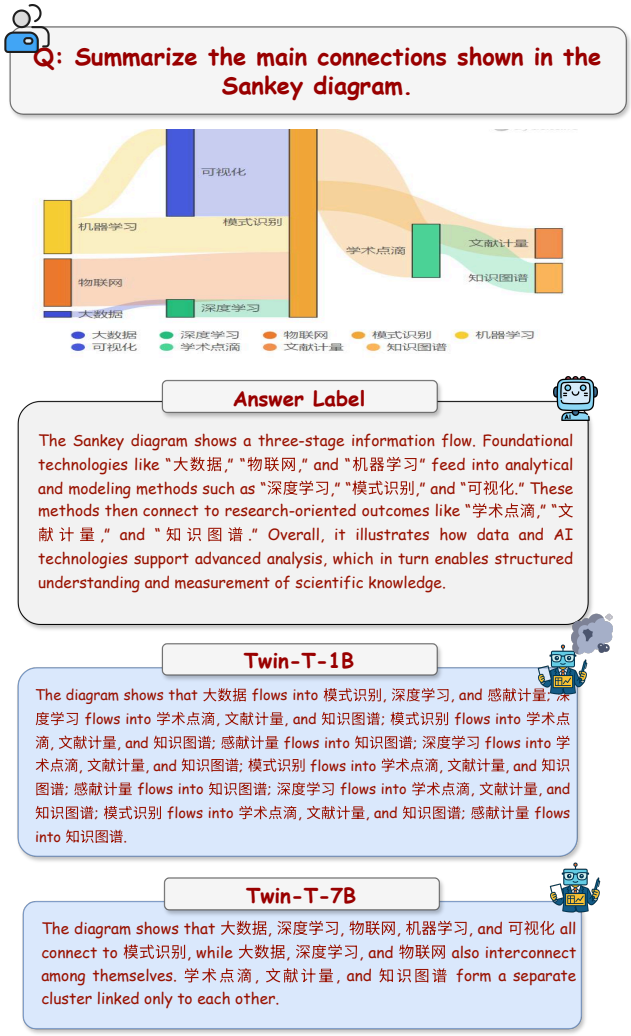


Figure 13. Twin-T VLM failure case 2.

Failure case 1. Dense Gantt image with overlapping timelines require reading two numeric spans (e.g., 42 vs. 28), aligning them to the correct task segment, and then performing a subtraction to get the duration. This means the model has to do three things at once: parse chart structure, bind each number to the right bar, and execute a small arithmetic step. In this case, Twin-T fails to recover the correct result. We believe this comes from two factors. First, this type of reasoning is often expressed in bilingual (Chinese/English) engineering charts with embedded code-like labels, which our data only weakly covers. Second, the model sees very few supervised examples where it must both interpret program-style numeric annotations and then compute a derived value, rather than just copying a number.

Failure case 2. The question asks for a high-level summary of the main connections in a Sankey diagram. The reference answer describes a three-stage flow: foundational technologies feed into analytical and modeling methods, which then connect to research-oriented outcomes. In short, many thin links aggregate into a pipeline from foundations to methods to outcomes, and the summary should state this coarse structure rather than enumerate every small edge. Notably, the diagram mixes English and Chinese labels; the task is bilingual and difficult, since correct summarization requires cross-lingual matching and aggregation before abstraction.

Twin-T-1B lists many local edges and mixed Chinese label fragments but fails to abstract them into the three-stage pipeline, often treating intermediate method labels as endpoints and pulling in stray terms. Twin-T-7B is more concise yet still misclusters nodes and understates cross-stage flows, missing the direction from foundations to methods to outcomes. The likely causes are overweighting local link

segments, underusing global stage structure, and sensitivity to bilingual long labels and small-width flows. This task requires hierarchical summarization of a graph-like layout: detect stages and hubs first, then verbalize the coarse flow. Without this abstraction step, the 1B model over-enumerates and the 7B model over-compresses, so both miss the intended foundations → methods → outcomes narrative.

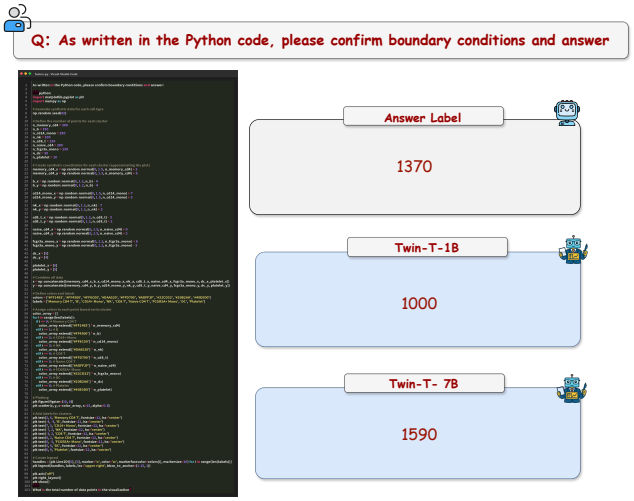


Figure 14. Twin-T VLM failure case 3.

Failure case 3. The question asks the model to read a block of Python code that builds a scatter plot using `matplotlib` and then answer: "What is the total number of data points in the visualization?" The code does not directly print this number. Instead, it first defines how many points belong to each of nine cell-type clusters, for example `n_memory_cd4 = 300, n_b = 150, n_cd14_mono = 250`, and so on. It then generates synthetic coordinates for each cluster and finally concatenates all clusters together with `np.concatenate(...)` before plotting them as a single scatter plot. Therefore, the correct way to answer the question is to identify all nine cluster sizes and sum them: 300 + 150 + 250 + 100 + 120 + 280 + 130 + 30 + 10 = 1370. The correct answer is 1370 total data points.

This failure case is challenging for two reasons. First, it is not a simple visual recognition or single-number lookup problem. The model must (1) understand the structure of the code, (2) recognize that variables such as `n_memory_cd4`, `n_b`, etc. represent the number of points in each cluster, (3) verify that all of these clusters are actually included in the final concatenation that produces the plot, and (4) add them without skipping or double-counting. This requires multi-step reasoning over long context, rather than surface-level pattern matching. Second, the model must maintain numerical consistency and distinguish true sample counts from other constants such as offsets or plot styling parameters.

In our experiment, Twin-T-1B predicted 1000 and Twin-T-7B predicted 1590, while the correct answer is 1370. Both models failed. This indicates a consistent weakness: when the question requires extracting multiple related quantities from code and performing arithmetic over them, the models tend to either drop some clusters or include unrelated values. In other words, even though the question appears to be "count the dots in the figure," it is in fact a code reasoning and aggregation problem, and both the 1B and 7B models are still unstable on this kind of structured numerical reasoning.

We further conduct corruptions that directly affect edge extraction, including Gaussian noise ($\sigma = 10, 20$), motion blur (kernel size 7), and perspective warp (random homography). Under these perturbations, Twin-T shows average drops of only **1.9** for 1B and **0.9** for 7B. In contrast, when the dual-head design is removed and replaced with single-head training, the average drops increase to **3.1** for 1B and **1.6** for 7B, showing weaker robustness. For example, in a bar chart that asks for the category with the highest value, Gaussian noise or motion blur can corrupt bar boundaries and axis cues. A single-head encoder may mix these perturbations with useful visual signals and misread the chart, while the dual-head encoder separates structural cues from local details, leading to more stable predictions under corruption.

7. Future Works

Twin-T achieves state-of-the-art performance among open-source VLMs and is already competitive with large API models on many chart-table tasks. We highlight two main directions for future work.

First, Twin-T can still fail on high-density, and mixed-language reasoning cases. These include questions that require (i) locating the exact relevant value from long code or densely annotated figures, and (ii) aligning that value with the correct semantic target (e.g., "current capacity" vs. historical values). Such cases require full-chain supervision (*read* → *align* → *compute*) rather than only supervising the final answer. We plan to further strengthen this style of stepwise supervision so the model not only outputs numbers, but can also point to their source and carry out the intermediate calculation reliably.

Second, Twin-T still trails some large closed models on OCR-heavy and structure-reconstruction tasks, such as reading crowded annotations, restoring precise table/chart layout, or inferring plotting logic. We attribute this gap to limited OCR-centric pretraining and to the fact that our structural branch currently depends on fixed edge maps. Going forward, we will incorporate stronger document/OCR pretraining and multilingual tabular data, and re-

place fixed edge extraction with learnable structural extraction and adaptive gating. This should improve robustness on noisy, real-world, bilingual charts and tables.

8. Benchmarks Overview

SEEDBench2 SEED-Bench is a large-scale benchmark designed to evaluate the generative comprehension abilities of Multimodal Large Language Models (MLLMs). With 19,242 multiple-choice questions, it is six times larger than MMBench and nine times larger than MME, covering 12 evaluation dimensions across spatial and temporal understanding. Spatial understanding includes scene comprehension, instance attributes, localization, counting, spatial relations, interactions, visual reasoning, and text recognition. Temporal understanding focuses on action recognition, prediction, and event sequencing, assessing model performance in both static images and dynamic video.

The dataset is generated through an automated pipeline that uses foundational models to extract visual features, followed by question generation through GPT-4, and filtering by language models to ensure question quality. Human annotators then verify answers and classify questions into specific dimensions. The evaluation approach uses log-likelihood scoring for each option, eliminating reliance on external tools like ChatGPT and ensuring consistent, comparable evaluation. SEED-Bench evaluates 18 models, including LLMs, ImageLLMs, and VideoLLMs, revealing significant performance gaps, particularly in temporal reasoning, where VideoLLMs struggle.

SEED-Bench is designed to be scalable, with potential extensions to new domains such as audio or 3D. It also includes a public leaderboard to encourage community-driven progress. By setting new standards for multimodal evaluation, SEED-Bench plays a critical role in advancing the research and development of generative multimodal reasoning systems in AI.

AI2D The AI2D benchmark is a multimodal dataset designed to evaluate models' abilities to understand and reason about diagrammatic information in educational contexts, specifically in STEM education. It contains 1,000 English diagrams from elementary science textbooks, covering topics like food webs, life cycles, moon phases, and human physiology. The dataset combines images, text, and diagrammatic elements (e.g., arrows, labels, flowcharts) to simulate real-world educational materials.

The primary task involves answering multiple-choice questions that require joint understanding of textual captions, visual layouts, and diagrammatic relationships. Key challenges include cross-modal reasoning, which links textual explanations with diagrammatic structures, and hierarchical understanding, such as parsing multi-step processes or interconnected systems. The benchmark is used to assess models like Aquila-VL-2B, particularly for tasks requiring OCR and diagram interpretation.

While AI2D focuses on educational diagrams, it is limited to elementary science topics, restricting its generalization to broader domains. Its smaller scale, with 1,000 questions compared to newer benchmarks like SEED-Bench, limits its scalability. However, it has inspired extensions, such as AI2D-RST, which adds rhetorical structure annotations for deeper discourse analysis. AI2D plays a crucial role in advancing the development of models that understand and interpret educational diagrams in STEM contexts.

OCRVQA OCR-VQA is the first large-scale benchmark specifically designed for text-centric visual question answering, focusing on understanding and reasoning over text in images. It contains 207,572 book cover images paired with over 1 million question-answer pairs, spanning 32 genres and incorporating structured metadata such as titles, authors, and publication years.

The benchmark includes diverse question types such as text recognition, genre classification, and numerical reasoning. It combines human-written and template-based questions to reduce bias and support natural language variation. The answer space is notably large, with over 320,000 unique answers, many of which are out-of-vocabulary terms, requiring models to reason across both textual content and visual design elements.

OCR-VQA presents significant challenges due to the varied fonts, orientations, and layouts of book cover text, which often lead to OCR errors. Approximately 20% of test answers are unseen during training, demanding strong generalization capabilities from models. Baseline models combining CNN features, text layout, and question encodings achieve modest performance, limited mainly by OCR inaccuracies.

Compared to other benchmarks like TextVQA, OCR-VQA emphasizes structured visual text (e.g., book titles) over unstructured scene text and offers a significantly larger and more diverse dataset. It serves as a bridge between document understanding and visual reasoning, laying the groundwork for OCR-integrated VQA systems in real-world applications such as digital libraries, cataloging, and document-based QA.

TableVQA TableVQA is a benchmark developed to assess models' abilities to understand and reason over tabular data rendered as images. It includes approximately 1,500 question–answer pairs derived from real-world tables across various domains, rendered into image form to simulate OCR and visual layout challenges.

Unlike text-based table QA datasets that use structured table formats, TableVQA requires models to visually parse table layouts—recognizing rows, columns, merged cells,

headers, and numeric entries—and perform reasoning such as numerical comparison, aggregation, and relation extraction.

Results indicate that MLLMs exhibit significant performance drops on visual table images compared to their textual representations, reflecting limited capability in visual structure parsing. TableVQA provides a focused benchmark for developing models that combine vision, OCR, and reasoning for document-level table understanding.

CharXiv CharXiv is a benchmark designed to evaluate Multimodal Large Language Models (MLLMs) on scientific chart understanding. It contains 2,323 real-world charts and tables collected from arXiv papers, divided into two complementary splits: Descriptive (CharXivD) and Reasoning (CharXivR). Together, they assess both perception-level comprehension and higher-level analytical reasoning over authentic scientific graphics.

CharXivD focuses on factual comprehension and element identification. It requires models to recognize visual components such as axes, labels, legends, and data encodings. The dataset uses open-ended short-answer questions emphasizing recognition and interpretation of visual elements rather than reasoning. This split measures perception-level understanding and evaluates a model's capacity to extract structured visual information from real scientific graphics, moving beyond synthetic or template-based datasets.

CharXivR targets analytical and inferential reasoning tasks based on the same set of real-world charts and tables. Its open-ended short-answer questions demand integration of textual and numerical information to infer trends, compare categories, and draw conclusions from data relationships. While CharXivD captures descriptive comprehension, CharXivR advances to reasoning that links visual patterns to underlying semantics. Empirical results show that MLLMs perform well in recognition tasks but struggle with reasoning, particularly in quantitative comparison and causal inference, making CharXivR a vital complement within the unified CharXiv framework.

ChartMimic ChartMimic is a benchmark designed to evaluate models' capabilities in chart-to-code generation—a multimodal task requiring vision, language, and structured reasoning. It consists of approximately 4,800 high-quality triplets of chart images, natural-language descriptions, and corresponding executable code snippets that reproduce the original visualization.

Covering 22 chart types and over 200 subcategories, ChartMimic challenges models to extract visual semantics, identify data mappings, and produce syntactically valid code. This task moves beyond recognition or question answering by demanding compositional generation that bridges perception and symbolic output.

Evaluation metrics include code correctness, rendering fidelity, and semantic consistency. Current MLLMs show significant limitations in aligning visual patterns with programmatic constructs. ChartMimic establishes a unique platform for advancing research in visual-to-code understanding and multimodal synthesis.

ChartQA ChartQA is a visual question answering benchmark centered on reasoning over chart images. It includes roughly 9,600 human-authored and 23,000 template-generated question–answer pairs across various chart types. The benchmark focuses on quantitative reasoning, arithmetic computation, and trend inference based on visualized data.

Unlike generic VQA datasets that prioritize object recognition, ChartQA emphasizes numerical understanding—requiring models to extract values, compare magnitudes, and reason about relationships in charts. It also includes textual chart components (titles, legends, axis labels), promoting cross-modal reasoning. Some methods leverage underlying data tables extracted from charts, but the benchmark itself evaluates on chart images.

Empirical evaluations indicate that current MLLMs face difficulty performing multi-step arithmetic and logical reasoning over chart data. ChartQA provides a standard framework for assessing quantitative reasoning across structured visual formats.

LogicVista LogicVista assesses the logical reasoning skills of Multimodal Large Language Models in visually grounded tasks involving diagrams, charts, and other structured visuals. It contains 448 carefully annotated multiple-choice questions spanning five reasoning categories—inductive, deductive, numerical, spatial, and mechanical—and eleven specific reasoning skills. Certain problems use tabular layouts akin to tables, which require structural parsing before reasoning.

Each question integrates visual content such as geometric figures, tabular layouts, or object arrangements, requiring models to infer rules, predict outcomes, or apply formal reasoning steps. In contrast to perception-focused benchmarks, LogicVista concentrates on structured reasoning and problem-solving.

Experimental results show that MLLMs achieve limited success in formal reasoning, particularly in tasks demanding multi-step inference and logical abstraction. LogicVista offers a challenging testbed for models that must integrate visual understanding with symbolic logic.