Extended Abstract

Motivation Large language models are increasingly used as instructional assistants, but procedural knowledge is often difficult to convey through text alone. Diagrams such as flowcharts and sequence diagrams can clarify process structure, decisions, and participant interactions. Producing both is particularly challenging for small language models, which have limited capacity compared with larger ones. This project investigates whether Qwen 2.5 0.5B can be adapted into a diagram-aided teaching assistant that generates concise procedural explanations paired with valid Mermaid.js diagrams.

Method The model is trained to generate a concise explanation and Mermaid diagram from a procedural prompt. Training uses 450 synthetic examples organized into three increasingly complex curriculum levels. Level 1 uses linear flowcharts for step-by-step procedures, Level 2 adds branching flowcharts with decision points, and Level 3 uses sequence diagrams with multiple participants and ordered interactions.

Each example includes a prompt, a chosen response, and a rejected response. Rejected responses are realistic near-misses with one major flaw, such as invalid syntax, missing steps, incorrect ordering, missing branches, the wrong diagram type, or participant inconsistency. These negatives encourage the model to learn fine-grained structural and semantic distinctions.

Implementation Three training strategies are compared. The first is supervised fine-tuning (SFT), which trains the model on chosen responses using standard completion-based learning. The second is IPO Random, which applies Identity Preference Optimization to the SFT model using randomly ordered preference pairs. The third is IPO Curriculum, which applies Identity Preference Optimization using the same preference data but presents examples in curriculum order, progressing from simpler tasks to more complex ones.

Results Evaluation is performed on a test set of 150 examples balanced across all three curriculum levels. Both preference-based and generation-based metrics are considered. The results show that supervised fine-tuning is highly effective at teaching Mermaid syntax and output formatting. All evaluated models achieve 100% basic Mermaid validity, indicating that the model reliably learns the required diagram structure from the training data. However, strict validity remains near 80%, suggesting that most remaining failures arise from semantic issues rather than syntax errors.

Preference optimization improves the model's ability to distinguish preferred responses from flawed alternatives. IPO Random achieves the largest average margin improvement, while IPO Curriculum obtains the highest preference accuracy and performs best on examples that are difficult for the SFT model. Curriculum IPO also achieves the strongest results on branching-flowchart tasks, indicating that structured training order may be particularly beneficial when learning more complex procedural relationships. Overall, the findings suggest that the primary challenge is no longer generating valid Mermaid syntax but generating diagrams that are semantically faithful to the intended procedure.

Conclusion This work demonstrates that a 0.5B language model can be adapted into a diagram-aided pedagogical assistant using synthetic data, supervised fine-tuning, and preference optimization. The results show that SFT alone is sufficient to teach reliable Mermaid formatting and diagram syntax, while IPO improves alignment with preferred responses and strengthens the model's ability to distinguish correct diagrams from subtly flawed alternatives.

Curriculum-based IPO provides additional benefits on challenging examples and branching diagrams, suggesting that training order can play a useful role in structured generation tasks. Although syntax-related errors become rare after fine-tuning, semantic correctness remains a significant challenge. Future work should therefore focus on stronger semantic supervision, harder negative examples, and evaluation methods that more directly measure diagram correctness.

Extension The Countdown extension tests whether structured prompting with Mermaid diagrams also improves arithmetic reasoning. Structured text prompting improves pass@1, but Mermaid scaffolding does not outperform simpler step-by-step prompting and reduces pass@16 performance, suggesting that Mermaid diagrams help as instructional artifacts but are not necessarily effective intermediate representations for arithmetic reasoning.

Teaching Small Models to Teach

Rosemary Jiang
Department of Computer Science
Stanford University
rmjiang@stanford.edu

Abstract

Large language models are increasingly used as instructional assistants, but small models often struggle to combine natural-language explanations with structured visual representations such as flowcharts and sequence diagrams. This project fine-tunes Qwen 2.5 0.5B to generate concise procedural explanations paired with syntactically valid and semantically aligned Mermaid.js diagrams. A synthetic dataset of preference triplets was constructed across three curriculum levels: linear flowcharts, branching flowcharts, and multi-participant sequence diagrams. Supervised fine-tuning (SFT), Identity Preference Optimization (IPO) with randomly ordered preference pairs, and IPO with curriculum-ordered preference pairs are compared. Results show that SFT alone is sufficient to teach reliable Mermaid formatting, with all evaluated models reaching 100% basic Mermaid validity. IPO improves chosen-over-rejected preference margins, while curriculum IPO achieves the strongest preference accuracy and best performance on SFT-hard examples. However, strict generation validity remains mixed, indicating that the main challenge shifts from syntax to semantic specificity. Overall, the findings demonstrate that small models can be adapted into useful diagram-aided pedagogical assistants, but future gains will require stronger semantic supervision and harder negative examples.

1 Introduction

Large language models are increasingly used as instructional assistants, but most responses remain primarily text-based. For procedural teaching tasks, text alone can be insufficient: learners often benefit from visual structure, such as flowcharts for step-by-step processes or sequence diagrams for interactions among multiple agents. While larger models can often generate these diagrams reliably, smaller models struggle to maintain both syntactic validity and semantic consistency when producing structured outputs such as Mermaid.js diagrams.

This project studies whether a lightweight model, Qwen 2.5 0.5B, can be fine-tuned to generate concise procedural explanations paired with valid Mermaid diagrams. The target behavior is not only to produce parseable diagram code, but also to ensure that the diagram matches the explanation and the user's requested procedure.

Three training strategies were compared: supervised fine-tuning (SFT) on preferred diagram-aided responses, Identity Preference Optimization (IPO) with randomly ordered preference pairs, and IPO with curriculum-ordered preference pairs. The curriculum progresses from simple linear flowcharts, to branching decision diagrams, to multi-participant sequence diagrams. This design tests whether ordering examples from simple to complex improves a small model's ability to learn structured instructional behavior.

The central research question is: *Can curriculum-based preference optimization improve a small language model's ability to generate valid and pedagogically useful diagram-aided explanations?*

The results show that SFT is sufficient to teach reliable Mermaid formatting, while IPO improves the model's preference for correct responses over plausible flawed alternatives. Curriculum IPO provides targeted benefits on hard preference cases and branching diagrams, suggesting that curriculum-based alignment can improve reliability.

2 Related Work

This project builds on three lines of work: preference optimization for language-model alignment, curriculum learning for staged training, and structured tool-based generation. Direct Preference Optimization (DPO) showed that language models can be aligned directly from preference pairs without training an explicit reward model or running a full reinforcement-learning loop Rafailov et al. (2023). This is important for small-model fine-tuning because it reduces the complexity of preference-based alignment. Identity Preference Optimization (IPO) proposes a finite-margin objective that can reduce the tendency of preference methods to over-optimize separable preference pairs Azar et al. (2024). This project adopts this preference-optimization framing, but applies it specifically to a structured diagram-generation task.

Curriculum learning provides the second motivation for this work. Bengio et al. argue that training examples can be ordered from easier to harder cases to improve optimization and generalization Bengio et al. (2009). This idea is especially relevant for small models, which may struggle when asked to learn syntax, procedural reasoning, and diagram semantics simultaneously. This project instantiates the curriculum over diagram complexity: Level 1 examples require linear flowcharts, Level 2 examples introduce branching decision structure, and Level 3 examples require multi-participant sequence diagrams.

Recent work on tool-integrated reasoning also motivates treating Mermaid diagrams as more than surface formatting. ToRA demonstrates that combining natural-language reasoning with external tool use can improve mathematical problem solving, especially when models are trained on structured tool-use trajectories Gou et al. (2023). Although Mermaid is not an executable solver in the same way as Python or symbolic tools, it functions as a structured representation language: the model must translate a natural-language procedure into a constrained formal syntax. This project therefore treats Mermaid generation as a lightweight form of tool-integrated reasoning for pedagogy, where the tool output is a human-readable diagram.

The most directly related benchmark is MermaidSeqBench, which evaluates LLMs on generating Mermaid sequence diagrams from natural-language prompts Shbita et al. (2025). MermaidSeqBench highlights that structured diagram generation remains underdeveloped and that models can fail along fine-grained dimensions such as syntax correctness, activation handling, error handling, and practical usability.

Overall, prior work establishes that preference optimization can align language models, curricula can shape learning dynamics, and structured tool outputs are useful for reasoning. However, there is limited work on combining these ideas to train very small models for diagram-aided pedagogy. This project addresses that gap by using curriculum-ordered IPO to align Qwen 2.5 0.5B toward responses that are syntactically valid, structurally appropriate, and semantically aligned with procedural teaching prompts.

3 Methods

3.1 Task Definition

The goal of this project is to fine-tune a small language model to act as a diagram-aided procedural teaching assistant. Given a natural language prompt x, the model must produce a short explanatory response together with a syntactically valid Mermaid.js diagram. Unlike ordinary instruction-following, this task requires the model to satisfy both textual and structural constraints: the written explanation should describe the requested procedure, while the Mermaid diagram should encode the same procedure in a valid visual representation.

Each training example was defined as either a supervised pair (x, y) or a preference triplet

$$D_{\text{pref}} = \left\{ \left(x^{(i)}, y_w^{(i)}, y_l^{(i)} \right) \right\}_{i=1}^{N},$$

where $x^{(i)}$ is the user prompt, $y_w^{(i)}$ is the preferred response, and $y_l^{(i)}$ is a plausible but flawed rejected response. The preferred response contains a concise explanation and a valid Mermaid diagram. The rejected response is intentionally close to the preferred response but contains one main error, such as a syntax error, missing step, wrong order, missing branch, wrong diagram type, or missing participant. This design makes the task a fine-grained alignment problem rather than a simple detection problem between high-quality and random outputs.

3.2 Synthetic Data Construction

A synthetic dataset of instructional prompts and diagram responses was constructed across three curriculum levels. The levels were designed to increase in structural complexity:

- **Level 1: Linear flowcharts.** Prompts ask for simple step-by-step procedures. The target response must include a Mermaid flowchart using graph top-down or graph left-to-right, with a strictly linear sequence and no branches, loops, or multiple actors.
- **Level 2: Branching flowcharts.** Prompts involve at least one decision point. The target response must include a valid Mermaid flowchart with at least one decision node using curly braces and meaningful branch labels such as `|Yes|`, `|No|`, `|Valid|`, or `|Invalid|`.
- **Level 3: Sequence diagrams.** Prompts describe interactions among multiple participants, systems, or agents. The target response must include a valid Mermaid `sequenceDiagram` with at least three participants and logically ordered messages.

The examples were generated using a structured data-generation prompt that required a fixed JSON schema:

$$\{\texttt{id}, \texttt{level}, \texttt{category}, \texttt{prompt}, \texttt{chosen}, \texttt{rejected}, \texttt{error\_type}\}.$$

The data-generation instructions also enforced several formatting constraints: Mermaid diagrams had to be placed inside markdown code fences labeled `mermaid`, node labels could not contain parentheses, Mermaid lines could not use semicolons, and the explanation had to match the diagram semantically. The prompts span across education, coding, daily tasks, research workflows, customer support, finance operations, healthcare administration, travel planning, and software systems.

For supervised fine-tuning, chosen responses were converted into query-completion pairs. For preference optimization, the full triplet structure was kept. The same underlying preference examples were used for both random-order and curriculum-order IPO; the only difference was the ordering of examples during training. The training set contained 450 examples (150 per level). The held-out evaluation set contained 150 examples (50 per level).

3.3 Supervised Fine-Tuning

The first training stage was supervised fine-tuning (SFT) on the preferred completions. Starting from Qwen 2.5 0.5B, the model was trained to maximize the likelihood of the chosen response conditioned on the prompt. Let $y = (y_1, \ldots, y_T)$ denote the assistant completion. The SFT objective minimizes next-token cross-entropy over completion tokens:

$$\mathcal{L}_{\text{SFT}}(\theta) = -\sum_{t=1}^{T} \log \pi_\theta(y_t \mid x, y_{<t}).$$

SFT serves two purposes in the project. First, it teaches the model the basic output format: a short explanation followed by a Mermaid code block. Second, it provides the initialization for preference optimization. In this setup, IPO is applied after the model has already learned the basic task distribution, so preference training can focus on distinguishing higher-quality diagrams from plausible flawed diagrams.

3.4 Identity Preference Optimization

After SFT, Identity Preference Optimization (IPO) was applied to align the model with the preference triplets. IPO compares the model's likelihood of the preferred and rejected responses relative to a

fixed reference policy. In these experiments, the reference policy π_{ref} is the SFT checkpoint used to initialize IPO.

For each preference triplet (x, y_w, y_l), the log-ratio difference was computed

$$h_\theta(x, y_w, y_l) = \log \frac{\pi_\theta(y_w \mid x)}{\pi_{\text{ref}}(y_w \mid x)} - \log \frac{\pi_\theta(y_l \mid x)}{\pi_{\text{ref}}(y_l \mid x)}.$$

Then the IPO loss was optimized

$$\mathcal{L}_{\text{IPO}}(\theta) = \left(h_\theta(x, y_w, y_l) - \frac{1}{2\beta} \right)^2.$$

Two IPO variants were trained: IPO Random, where preference examples were shuffled randomly during training, and IPO Curriculum, where examples were ordered from Level 1 to Level 2 to Level 3, so the model encountered linear syntax before branching logic and then multi-participant sequence diagrams. Since both IPO variants use the same examples and objective, differences in performance can be attributed primarily to training order rather than data content.

4 Experimental Setup

4.1 Models and Training Conditions

All experiments used Qwen 2.5 0.5B as the base language model. Three variants were compared:

- **SFT:** the base model fine-tuned on chosen completions only.
- **IPO Random:** the SFT model further optimized with IPO on randomly ordered preference pairs.
- **IPO Curriculum:** the SFT model further optimized with IPO on curriculum-ordered preference pairs.

Table 1 shows the training hyperparameters used in each variant.

Setting	SFT	IPO Random	IPO Curriculum
Base model	Qwen 2.5 0.5B	SFT checkpoint	SFT checkpoint
Epochs	6	3	3
Learning rate	5e-5	5e-6	5e-6
Batch size	16	8	8
Max prompt length	512	512	512
Max response length	1024	1024	1024
IPO beta	–	0.1	0.1
Gradient checkpointing	Yes	Yes	Yes

Table 1: Training hyperparameters used for the final model variants.

4.2 Evaluation Dataset

Evaluation was performed on a held-out set of 150 prompts, with 50 examples from each curriculum level. The same held-out set was used to evaluate SFT, IPO Random, and IPO Curriculum. For generation-based evaluation, models were prompted with an evaluation instruction and decoded with a maximum of 160 new tokens. This limit was chosen to be long enough for a short explanation and diagram, while discouraging overly verbose or repeated outputs.

4.3 Preference-Based Metrics

The first group of metrics evaluates whether the model assigns higher likelihood to the chosen response than to the rejected response. For each triplet, the average log-probability of the chosen and

rejected completions under the evaluated model was computed. Preference accuracy is defined as

$$\text{Preference Accuracy} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{1}\left[\log p_\theta(y_w^{(i)} \mid x^{(i)}) > \log p_\theta(y_l^{(i)} \mid x^{(i)})\right].$$

The average log-probability margin was also reported:

$$\text{Average Margin} = \frac{1}{N}\sum_{i=1}^{N} \left[\log p_\theta(y_w^{(i)} \mid x^{(i)}) - \log p_\theta(y_l^{(i)} \mid x^{(i)})\right].$$

This metric captures not only whether the model ranks the chosen response above the rejected response, but also how strongly it separates them.

To test whether IPO improves examples that are genuinely difficult for the SFT model, an SFT-hard subset was also evaluated. This subset consists of examples where the SFT baseline has a low or incorrect chosen-over-rejected margin (with the hard margin threshold set to 1, there were 28 Level 1 examples in the set). Improvements on this subset are especially important because they indicate that IPO is not merely increasing confidence on already-easy preference pairs.

4.4 Generation-Based Metrics

The second group of metrics evaluates the quality of model-generated responses. Since the task requires valid Mermaid diagrams, the evaluator checks both surface formatting and structural constraints. The basic validity checks include:

- whether a Mermaid code block is present,
- whether exactly one Mermaid/code block is generated,
- whether the diagram type matches the requested level,
- whether the diagram satisfies level-specific structure constraints,
- whether the output avoids risky Mermaid syntax,
- whether the explanation is concise and cleanly formatted.

Basic Mermaid validity is defined as satisfying the core syntax and structure checks: a Mermaid block is present, the diagram type is correct, the level-specific structure is valid, and the output avoids risky syntax. **Strict validity** is defined as a stronger metric that also requires clean formatting, exactly one code block, no extra text after the final code fence, reasonable response length, and semantic alignment with the prompt or reference.

4.5 Semantic Alignment Metrics

Semantic proxy metrics include prompt overlap, reference overlap, label F_1, and structure-signature similarity. Label F_1 compares the content labels in the generated diagram against the reference diagram. Structure-signature similarity compares the high-level diagram structure, such as linear chains, branches, or sequence interactions.

These metrics are imperfect as they can penalize valid paraphrases or miss deeper semantic mismatches. However, they provide a useful automated signal for whether the model is generating diagrams that are not only syntactically valid but also aligned with the requested procedure.

5 Results

5.1 Quantitative Evaluation

Table 2 shows that IPO primarily improves the preference-ranking behavior of the model. The SFT model already achieves high preference accuracy, indicating that many rejected responses are distinguishable even before preference optimization. However, both IPO variants increase the average log-probability margin between chosen and rejected responses. IPO Random achieves the largest

Metric	SFT	IPO Random	IPO Curriculum
Preference accuracy ↑	98.7%	98.7%	99.3%
Average log-probability margin ↑	1.858	2.171	2.070
Margin improvement over SFT ↑	0.000	+0.313	+0.211
SFT-hard subset accuracy ↑	92.9%	92.9%	96.4%
SFT-hard margin improvement ↑	0.000	+0.029	+0.034
Structure signature similarity ↑	0.989	0.984	0.988

Table 2: Preference-based evaluation on the 150-example test set. IPO improves the chosen-over-rejected margin, while curriculum IPO gives the strongest performance on examples that were difficult for the SFT model.

overall margin improvement, while IPO Curriculum achieves the best preference accuracy and the strongest performance on the SFT-hard subset. This suggests that curriculum ordering is most useful on difficult examples, whereas random ordering can produce larger average separation across the full test set.

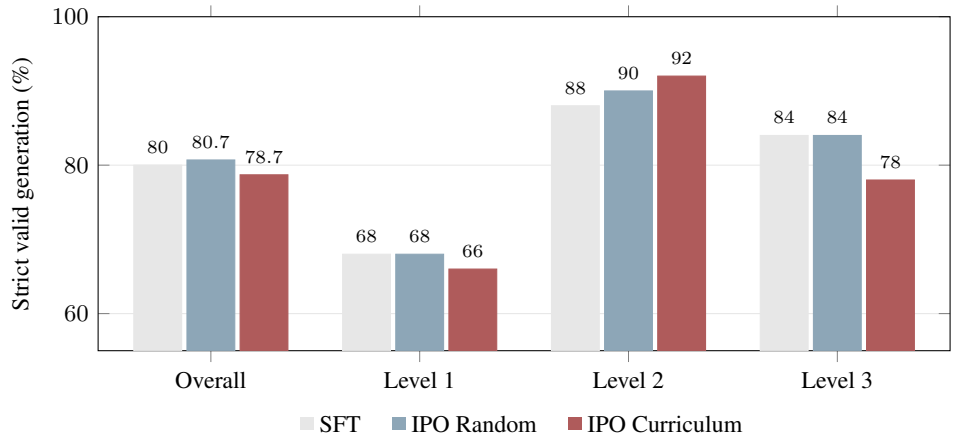

Figure 1: Strict generation validity by model and curriculum level. All models achieved 100% basic Mermaid validity, but strict validity varies once semantic alignment and formatting constraints are included.

Figure 1 shows that all three models perform well on basic Mermaid syntax, but strict generation validity remains more challenging. SFT reaches 80.0% strict validity overall, IPO Random reaches 80.7%, and IPO Curriculum reaches 78.7%. The strongest curriculum gain appears on Level 2 branching flowcharts, where IPO Curriculum reaches 92.0% strict validity compared with 88.0% for SFT and 90.0% for IPO Random. However, curriculum IPO is weaker on Level 3 sequence diagrams, suggesting that the curriculum ordering helped decision-structure tasks more than multi-participant interaction tasks.

Figure 2 shows that preference margins increase with task complexity. Level 1 examples have the smallest margins, while Level 3 sequence-diagram examples have the largest margins. This likely occurs because Level 3 rejected responses often contain more visible structural errors, such as missing participants or incorrect message order. Both IPO variants improve margins at every level. IPO Random produces the largest margins for Level 2 and Level 3, while IPO Curriculum is slightly stronger on Level 1 and better on the SFT-hard subset. Overall, IPO improves the model's preference separation more clearly than it improves free-form generation quality.

5.2 Qualitative Analysis

The quantitative results suggest that the main bottleneck shifts over the course of training. After SFT, the model has largely learned the surface format of the task: it usually emits one Mermaid block, uses

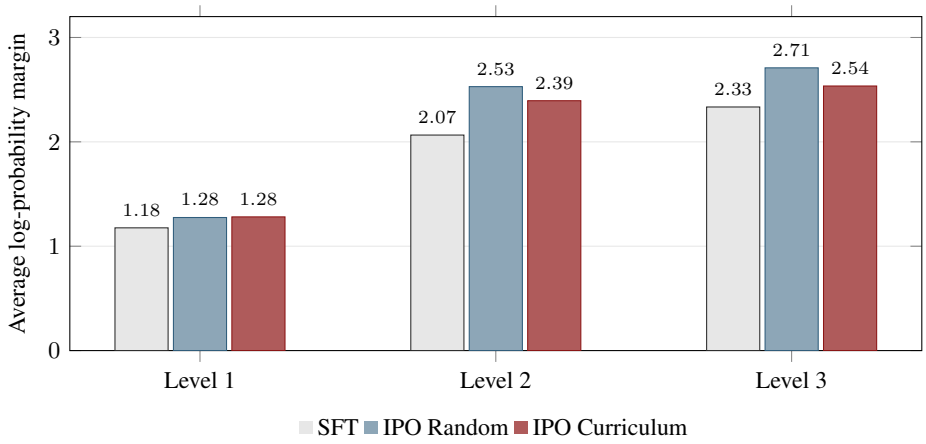

Figure 2: Average chosen-over-rejected log-probability margin by curriculum level. Both IPO variants increase the preference margin at every level.

the correct diagram type, and avoids risky Mermaid syntax. The remaining errors are more semantic than syntactic.

A representative example is `level1_162`, a Level 1 prompt asking the model to explain the procedure for reserving a rental locker and include a simple linear Mermaid flowchart. This example is useful because all models produce clean-looking linear diagrams, but the preference margins reveal differences in whether the model actually favors the better response.

Model	Generated semantic core	Strict	Label F1	Log-prob margin
Reference	Find locker location → choose locker size → enter rental time → pay fee → save access code	–	–	–
SFT	Check availability → choose locker → provide ID → pay fee → receive locker	Yes	0.40	-0.0039
IPO Random	Same semantic core as SFT, but with the requested left-to-right flowchart orientation	Yes	0.40	-0.0039
IPO Curriculum	Same semantic core as IPO Random, preserving linear structure and clean Mermaid formatting	Yes	0.40	$+0.0117$

Table 3: Qualitative comparison for `level1_162`. All models generate valid-looking linear diagrams, but the example still exposes differences in preference ranking.

In this example, the rejected response contains a wrong-order error. SFT and IPO Random both assign a slightly higher average log-probability to the rejected response, giving an average log-probability margin of -0.0039. IPO Curriculum corrects this preference ranking and assigns higher likelihood to the chosen response, with a margin of $+0.0117$. Although the absolute margin change is small, the direction matters because this example belongs to the type of SFT-hard case where the baseline model struggles to separate the preferred and rejected responses.

The qualitative pattern matches the aggregate metrics. SFT is strong at learning the required answer template and Mermaid syntax, which explains why all models reach 100% basic Mermaid validity. IPO improves preference margins, but its effect on direct generation is more mixed. Curriculum IPO is especially useful when the error corresponds to a structural concept introduced earlier in the curriculum, such as branching logic or step ordering. However, it does not uniformly improve semantic specificity across all levels. In particular, models sometimes reuse generic explanatory phrasing or produce plausible but underspecified diagram labels. This indicates that the next improvement

should come from harder negative examples and stronger semantic evaluation, rather than additional optimization for syntax alone.

6 Discussion

The results show that task-specific supervised fine-tuning is sufficient for teaching the small Qwen 2.5 0.5B model the basic output format required for diagram-aided procedural explanations. Across the test set, all evaluated models achieved perfect basic Mermaid validity, meaning they reliably produced a Mermaid block, used the correct diagram type, satisfied level-specific structure constraints, and avoided risky syntax. This suggests that, for this task, syntax and formatting are learnable even for a 0.5B model when the training data is consistent and highly structured.

However, the stricter evaluation reveals that syntactic validity is not the same as pedagogical quality. Strict validity remained around 80%, with failures often caused by weak semantic alignment, generic step labels, or partially mismatched procedures. This distinction is important because a diagram can be parseable and structurally valid while still being a poor teaching aid if it omits, reorders, or abstracts away key procedural details.

Both IPO variants increased the chosen-over-rejected log-probability margin relative to SFT, showing that preference optimization made the model better at distinguishing high-quality responses from plausible flawed ones. Curriculum IPO achieved the best preference accuracy and strongest performance on SFT-hard examples, suggesting that easy-to-hard ordering can help when the baseline model is uncertain. At the same time, random IPO slightly outperformed curriculum IPO on overall strict generation validity, indicating that curriculum ordering is not universally helpful.

These findings support a nuanced interpretation of curriculum-based preference optimization. Curriculum ordering appears most useful for targeted reliability, especially on difficult preference pairs and branching-logic tasks. However, the gains do not automatically transfer to all forms of generation quality. Future work should therefore focus less on basic syntax learning and more on semantic robustness: stronger hard negatives, execution-grounded Mermaid validation, larger training and test sets, and metrics that better capture diagram meaning.

7 Conclusion

This project investigated whether a small language model can be fine-tuned to generate clear procedural explanations paired with valid Mermaid.js diagrams, by comparing supervised fine-tuning, IPO with randomly ordered preference data, and IPO with curriculum-ordered preference data across linear flowcharts, branching flowcharts, and sequence diagrams. The results show that SFT alone can teach the model reliable Mermaid syntax and formatting, while IPO further improves its ability to prefer correct responses over flawed alternatives.

The strongest overall finding is that preference optimization improves chosen-over-rejected margins while largely preserving clean diagram generation. Curriculum IPO provides targeted gains on hard preference cases and Level 2 branching tasks, but it does not dominate random IPO across every generation metric. This suggests that curriculum-based IPO is a promising alignment strategy for small diagram-generating models, but that future improvements will depend on better semantic supervision rather than syntax-focused training alone.

Overall, this work demonstrates that lightweight models can be adapted into useful diagram-aided pedagogical assistants. While the model does not fully close the gap with larger systems, it shows that structured synthetic data, supervised fine-tuning, and preference optimization can substantially improve a 0.5B model's ability to produce valid and instructional diagram-based responses.

Contributions

Rosemary Jiang completed all components of the project.

Changes from Proposal Extension on the countdown task was added to the Appendix. There are no other significant changes from the project proposal.

References

Mohammad Gheshlaghi Azar, Mark Rowland, Bilal Piot, Daniel Guo, Daniele Calandriello, Michal Valko, and Rémi Munos. 2024. A General Theoretical Paradigm to Understand Learning from Human Preferences. In *International Conference on Artificial Intelligence and Statistics*.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum Learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*. 41–48.

Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Minlie Huang, Nan Duan, and Weizhu Chen. 2023. ToRA: A Tool-Integrated Reasoning Agent for Mathematical Problem Solving. *arXiv preprint arXiv:2309.17452* (2023).

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. 2023. Direct Preference Optimization: Your Language Model is Secretly a Reward Model. *arXiv preprint arXiv:2305.18290* (2023).

Basel Shbita, Farhan Ahmed, and Chad DeLuca. 2025. MermaidSeqBench: An Evaluation Benchmark for LLM-to-Mermaid Sequence Diagram Generation. In *NeurIPS 2025 Workshop on Evaluating the Evolving LLM Lifecycle: Benchmarks, Emergent Abilities, and Scaling*.

Appendix

A Additional Experiment: Does Diagram-Style Reasoning Improve Arithmetic Search?

A late-stage extension was whether the procedural structure learned through Mermaid diagram generation could transfer to the default Countdown task. Countdown provides a useful stress test because it requires a model to search over multi-step arithmetic expressions while respecting a strict output format: the final expression must appear inside an `<answer>...</answer>` span and must use the given numbers exactly once. Rather than retraining on a new Mermaid-Countdown dataset, which would require constructing and verifying a new source of paired diagram-and-arithmetic supervision, this hypothesis was evaluated through a controlled comparison of test-time prompting strategies.

The model checkpoint, test set, verifier, sampling budget, and decoding parameters were fixed, while the prompt scaffold varied in three ways: (i) the original Countdown prompt, denoted *baseline*; (ii) a *structured text* prompt that asked the model to briefly plan the arithmetic steps before producing the final answer; and (iii) a *Mermaid scaffold* prompt that asked the model to first express the arithmetic plan as a small Mermaid flowchart before giving the final answer. Each condition was evaluated with 16 sampled responses per test example, and performance was measured using pass@K.

Figure 3 shows the structured-text condition improves pass@1 from 28.6% to 32.3%, while the Mermaid-scaffold condition improves pass@1 from 28.6% to 30.9%. This suggests that adding an explicit planning instruction can help the model's first sampled solution. However, the benefits do not hold uniformly as the sampling budget increases. At pass@16, the baseline and structured-text prompts both reach 78.0%, while the Mermaid scaffold reaches only 72.0%. Thus, Mermaid-style externalization does not outperform simpler structured reasoning on Countdown, and in fact reduces high-K performance.

This result clarifies the scope of the main project. Mermaid supervision can improve a small model's ability to produce structured pedagogical artifacts, but this does not automatically imply that Mermaid diagrams are an effective intermediate representation for arithmetic search. A likely explanation is that the diagram requirement introduces additional formatting burden: the model must spend part of its limited generation capacity maintaining Mermaid syntax before producing the verifier-sensitive `<answer>` block. In contrast, the structured-text prompt provides planning guidance without imposing a second formal language. Overall, the preliminary findings are structured reasoning helps Countdown at low K, but Mermaid-specific scaffolding does not provide an additional benefit over generic step-by-step prompting.

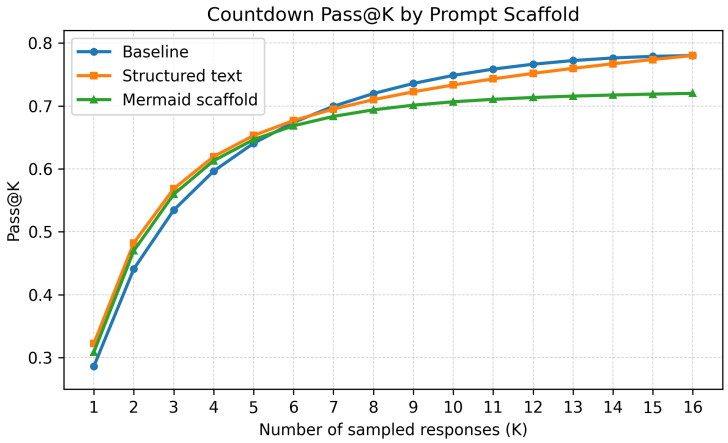

Figure 3: **Countdown pass@K under three prompt scaffolds.** The model, test set, verifier, sampling budget, and decoding parameters are held fixed while only the prompt scaffold changes. Structured text improves pass@1 relative to the baseline, while the Mermaid scaffold improves low-K performance less and falls behind at high K. This suggests that explicit planning helps, but forcing a diagrammatic Mermaid representation adds output-format overhead for Countdown arithmetic search.

A.1 Qualitative Example: Mermaid Scaffolding Can Add Formatting Burden

A representative example helps explain why the Mermaid scaffold underperformed the simpler structured-text prompt at higher sampling budgets. Consider the Countdown problem with numbers $[83, 78, 1, 39]$ and target 82. This example has a compact valid solution:

$$(83 - (78/39)) + 1 = (83 - 2) + 1 = 82.$$

Table 4 compares the three prompt scaffolds on this same problem. The baseline prompt produced no correct samples, although most generations were parseable enough to receive the partial format score. The structured-text scaffold found the correct quotient-based solution in one of its 16 samples. In contrast, the Mermaid-scaffold condition produced parseable but incorrect expressions in all 16 samples. This example illustrates the broader trend: generic structured reasoning can help the model search over arithmetic subgoals, while forcing the model to produce a diagrammatic representation may add formatting burden without improving the final verifier-sensitive answer.

]

Prompt scaffold	**Correct / 16**	**Verifier pattern**	**Representative final expression**
Baseline	$0/16$	Mostly format-only	$(83 - 39) + (78 + 1) = 123$
Structured text	$1/16$	One correct sample	$(83 - (78/39)) + 1 = 82$
Mermaid scaffold	$0/16$	Format-only for all samples	$83 - (78 - 39) = 44$

Table 4: **Qualitative Countdown example comparing prompt scaffolds.** For the problem $[83, 78, 1, 39] \rightarrow 82$, the structured-text scaffold discovers the key intermediate computation $78/39 = 2$, while the Mermaid scaffold produces parseable but incorrect arithmetic expressions. This supports the aggregate finding that Mermaid-style externalization adds output complexity without reliably improving Countdown arithmetic search.

This example is especially informative because the correct solution requires a non-obvious intermediate operation, $78/39 = 2$. The structured-text prompt explicitly encourages arithmetic planning and eventually finds this quotient. The Mermaid prompt, however, tends to produce arithmetic-looking steps that remain parseable but do not evaluate to the target. Thus, the diagram scaffold appears to distract from the underlying arithmetic search. This supports the interpretation that Mermaid diagrams are useful as pedagogical artifacts, but they do not automatically serve as an effective planning language for Countdown-style mathematical reasoning in small language models.