Improving Improvisational Reasoning in Vision-Language Models for Procedural Adaptation

Aditya Kuniyil Kattil
Purdue University
akuniyil@purdue.edu

Mahad Faruqi
Purdue University
mfaruqi@purdue.edu

Abstract

Instruction-following systems are most useful when they can recover from disruptions that make the original plan infeasible. This project studies procedural adaptation under visual missing-ingredient disruptions in cooking procedures derived from YouCook2. We construct a benchmark of disrupted procedural instances and evaluate multiple open vision-language models under text-only, vision-conditioned, retrieval-augmented, and reranked retrieval settings. The text-only settings deliberately remove image input while using the same VLM, allowing us to isolate whether improvements come from visual grounding or from structured textual context and retrieved analogies. On a 390-example held-out visual-disruption test set, the base Qwen2-VL-2B study shows that retrieval provides the largest gain at small scale: the text-only baseline achieves 137 successful adaptations, adding visual context improves performance to 163 successes, text retrieval reaches 256 successes, and vision retrieval reaches 248 successes. However, larger-model follow-up experiments with Qwen2.5-VL and Gemma show that this ordering is not stable across model families or scales. In the stronger models, vision-grounded retrieval becomes the most effective strategy among the completed settings. These results suggest that the relative value of retrieval, visual context, and reranking depends on model capacity and model family rather than being constant across settings.

1 Introduction

Instruction-following systems are typically evaluated under idealized conditions: required ingredients are available, tools are present, inputs are correct, and each step can be executed as written. Real environments rarely satisfy these assumptions. A cooking assistant may need to proceed when olive oil is missing, when a needed pan is unavailable, or when the visible workspace no longer matches the expected state. In such cases, simply continuing the original recipe is not enough; the system must revise the plan while preserving the user's goal.

This project studies *procedural adaptation under disruption*. We focus on a visually grounded cooking setting in which a model receives a procedural goal, the current state of the recipe, a disrupted step, and, in vision settings, an image of the current workspace. The reported experiments isolate one common and practically important disruption type: missing ingredients. The model must generate an adapted continuation that avoids the unavailable ingredient and gives a feasible substitute or workaround.

The central research question is whether a vision-language model benefits more from visual context, retrieval of analogous procedural cases, or the combination of both. We evaluate a common six-condition prompting matrix: a text baseline, a vision baseline, text retrieval, vision retrieval, text retrieval with reranking, and vision retrieval with reranking. In the base 2B study, all six conditions use the same generator, Qwen2-VL-2B-Instruct (Wang et al., 2024), so the text-only settings serve as clean ablations of visual conditioning. We then extend the same comparison to larger Qwen and Gemma models where hardware permits.

The contributions of this work are threefold. First, we build a YouCook2-derived benchmark for visual procedural adaptation under missing-ingredient disruptions. Second, we evaluate a common six-condition prompting matrix that cleanly separates text-only prompting, visual grounding, retrieval, and reranking. Third, we show that the relative value of these components changes with model scale and model family: retrieval is dominant in the 2B Qwen setting, while stronger Qwen and Gemma models benefit more consistently from vision-grounded retrieval.

To support reproducibility, our implementation, including benchmark construction, retrieval config-

urations, and evaluation code, is publicly available at the project repository.

The rest of the paper is organized as follows. Section 2 reviews related work on procedural understanding, retrieval-augmented generation, prompting, and vision-language models. Section 3 describes the YouCook2-derived benchmark and visual missing-ingredient subset. Sections 4 and 5 define the task, method, and experimental setup. Section 6 presents the main results, ablations, and qualitative analysis, followed by discussion, limitations, and references.

2 Related Work

Procedural understanding has been studied through instructional datasets that capture multi-step human activities. YouCook2 (Zhou et al., 2018) provides temporally grounded procedural content derived from web instructional videos and motivates the use of realistic cooking procedures in this project. Related embodied and instructional benchmarks, such as ALFRED (Shridhar et al., 2020), extend procedural understanding into household action execution. The present work differs by focusing on recovery behavior when a procedure becomes infeasible, rather than on understanding or executing an undisrupted procedure.

Retrieval-augmented generation is another relevant line of work. Dense retrieval methods (Karpukhin et al., 2020) and retrieval-augmented generation frameworks (Lewis et al., 2020) show that external context can improve generation without changing model weights. Reranking methods refine retrieved candidates using stronger relevance estimation after initial retrieval (Nogueira and Cho, 2019). In this project, retrieval is not used primarily for factual knowledge access. Instead, it supplies analogous examples of procedural recovery, making retrieval a mechanism for adaptation.

Prompting and reasoning methods also inform this work. In-context learning (Brown et al., 2020) showed that large models can perform new tasks from examples, while chain-of-thought prompting (Wei et al., 2022) and planning-oriented approaches such as ReAct (Yao et al., 2023b) and Tree-of-Thought (Yao et al., 2023a) demonstrate that structured reasoning context can improve model behavior. Retrieval-augmented prompting in this project serves as a lightweight way to provide procedural precedents.

Finally, grounded planning systems such as SayCan (Ahn et al., 2022) emphasize that useful plans must be both semantically appropriate and practically feasible. Vision-language models such as Qwen2-VL (Wang et al., 2024) make it possible to condition generation on both text and images. Our experiments use this capability to compare text-only adaptation with visually grounded adaptation under the same procedural disruptions.

3 Datasets

The benchmark is derived from YouCook2 (Zhou et al., 2018), a cooking-video dataset containing procedural activities. Source procedures are normalized into a structured representation containing a goal, ordered steps, and metadata. The benchmark builder injects disruptions into procedural instances and records the context needed to ask a model for an adapted continuation.

The broader constructed benchmark contains 6044 disrupted procedural examples. The split sizes are 4286 training examples, 893 development examples, and 865 test examples. The reported experiments use the visual missing-ingredient subset, filtered to YouCook2 examples with an associated current-state image. This subset contains 1950 training examples, 418 development examples, and 390 held-out test examples.

We split the benchmark at the source-procedure level rather than at the individual disruption level. Concretely, all disrupted instances derived from the same `source_item_id` are assigned to the same partition using a deterministic 70/15/15 train/development/test split with seed 42. This design prevents leakage between splits, since multiple disruptions from the same underlying recipe video are often highly similar in goal, state, and visual context.

Each evaluated instance contains a task goal, current procedural state, disrupted step, missing ingredient, known safe substitute, and current-state image. In the reported study setting, disruptions are synthetic missing-ingredient interventions applied to otherwise intact YouCook2 procedures, and the reference adaptation is generated from a rule-based substitute template associated with the missing ingredient. The image input is a single current-state frame from the source video corresponding to the disrupted procedural context. The retrieval library is built from the training split, while all reported test results are computed on the same 390 held-out examples. Table 1 summarizes the benchmark used

Quantity	Count
Full benchmark examples	6044
Train split	4286
Development split	893
Test split	865
Final source dataset	YouCook2
Final disruption type	Missing ingredient
Final disruption modality	Vision
Target adaptation provenance	Rule-based
Visual missing-ingredient train subset	1950
Visual missing-ingredient development subset	418
Visual missing-ingredient test subset	390

Table 1: Benchmark and evaluation subset sizes. The reported experiments use the 390-example visual missing-ingredient test subset.

Example of a disrupted procedural instance
Goal: Tomato soup **Current state:** The soup has already been cooked and blended, and the next step is to finish the seasoning. **Disrupted step:** Add pepper to the soup. **Disruption:** Missing ingredient: pepper. **Known substitute:** Paprika, chili flakes, or omit it. **Desired behavior:** Replace pepper with a suitable substitute or omit it and continue serving the soup.

Table 2: Illustrative visual missing-ingredient adaptation instance.

in the reported experiments, while Table 2 and Figure 1 show a representative disrupted instance and its corresponding visual state.

4 Methodology and Problem Formulation

Figure 2 summarizes the common evaluation pipeline. Each example combines structured procedural context with a visual current state, and the experimental conditions differ in whether they include the image, retrieved cases, and reranked cases.

Let a procedural instance be represented as

$$x = (g, c, s_d, m, u, I), \tag{1}$$

where g is the task goal, c is the current procedural state, s_d is the disrupted step, m is the missing ingredient, u is a known safe substitute or workaround, and I is the current-state image. The model must produce an adapted plan

$$\hat{y} = f_\theta(x) \tag{2}$$

that addresses the disruption, avoids relying on the unavailable ingredient, remains feasible, and preserves the intended dish.

Figure 1: Example current-state image corresponding to the disrupted tomato-soup instance in Table 2. The model receives this visual context in the vision-conditioned settings.

Text and Vision Conditions. The text baseline receives the procedural fields but not the image. The vision baseline uses the same textual information and additionally provides the current-state image to the chosen VLM generator. Within each model-specific experiment, this isolates the effect of visual grounding while holding the generator fixed. In this sense, the text settings are not separate non-VLM experiments; they use the same vision-language model as a language model over structured procedural context, making them controls for measuring the added value of image input.

Retrieval-Augmented Prompting. When retrieval is enabled, the model is conditioned on retrieved adaptation cases from the training split. Let

$$R_k(x) = \{r_1, r_2, \ldots, r_k\} \tag{3}$$

denote the top-k retrieved cases. The generated output is then conditioned on both the test instance and retrieved examples:

$$\hat{y} = \arg\max_y P_\theta(y \mid g, c, s_d, m, u, I, R_k(x)). \tag{4}$$

In text retrieval settings, I is omitted from the model input; in vision retrieval settings, the image is included.

All prompting conditions use the same structured procedural fields: goal, current state, disrupted step, disruption type, missing ingredient, and suggested substitute. Retrieval uses a hybrid query formed by concatenating these fields into a single textual query. Each retrieval document contains the analogous training example's goal, current state, disrupted step, missing ingredient, and target adaptation. Dense embeddings are computed with `BAAI/bge-small-en-v1.5` through a local Hugging Face backend (Xiao et al., 2023). We retrieve from a candidate pool of 20 examples and use $k = 3$ examples in the prompt.

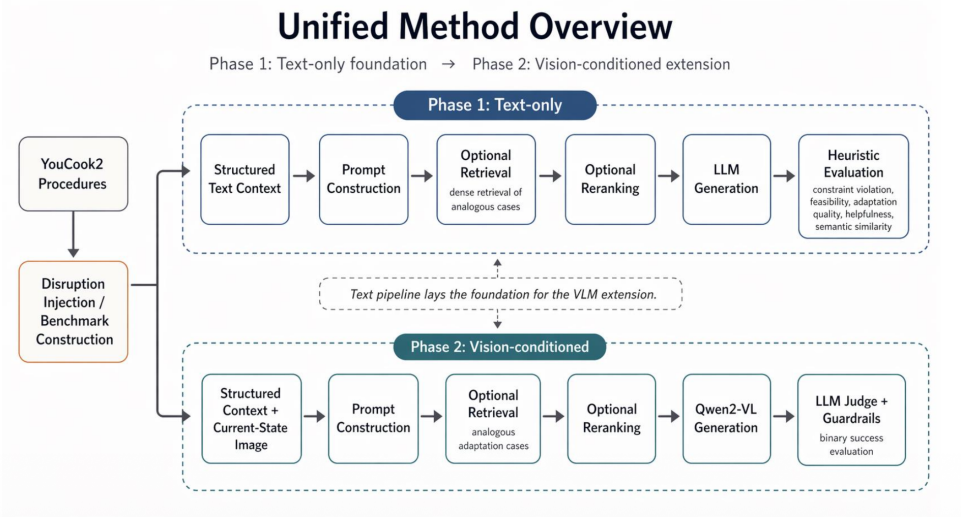

Figure 2: Common evaluation pipeline. Each example combines a YouCook2 procedure, a visual current state, and a missing-ingredient disruption. Depending on the condition, the prompt may include the image, retrieved adaptation cases, and reranked cases before a vision-language model such as Qwen2-VL, Qwen2.5-VL, or Gemma 3 generates an adapted plan. The output is then scored by an LLM judge with missing-ingredient guardrail checks.

Reranking. The reranker variants first retrieve candidates with the same dense retriever, then rerank the candidate pool with `BAAI/bge-reranker-base`. The top three reranked examples are inserted into the prompt. If reranking fails, the system falls back to the original retrieval order. Formally, if $C(x)$ is the 20-example candidate pool, the reranked prompt set is

$$\tilde{R}_k(x) = \mathrm{TopK}_{r \in C(x)} \, s_{\mathrm{rerank}}(x, r), \qquad (5)$$

where $s_{\mathrm{rerank}}(x, r)$ is the cross-encoder relevance score assigned by the reranker. We treat the reranker as an off-the-shelf scoring model and do not modify its internal objective or weights.

Prompt and Judge Structure. All six prompting conditions share the same core task fields: goal, procedure, current state, disrupted step, disruption description, missing ingredient, and a known safe substitute or workaround. The text-only conditions provide these fields as structured text, while the vision conditions add the current-state image to the same underlying prompt. Retrieval conditions append three retrieved adaptation cases, each containing the analogous training example's goal, disrupted context, and target adaptation; reranker conditions differ only in how those three cases are selected. The LLM judge receives the disrupted instance, the candidate adaptation, and a fixed rubric asking whether the response addresses the disruption, remains feasible, avoids relying on the unavailable ingredient, and preserves the intended dish. This shared prompt-and-judge structure is held constant within each model-specific matrix so that differences can be attributed to modality, retrieval, and reranking rather than to task wording.

Generation. Within each model-specific experiment, the generator is held fixed across all prompting conditions so that the comparison isolates prompting modality and retrieval strategy rather than model identity. The base experiment uses `Qwen/Qwen2-VL-2B-Instruct` (Wang et al., 2024); scaling and cross-family follow-ups use `Qwen/Qwen2.5-VL-3B-Instruct`, `Qwen/Qwen2.5-VL-7B-Instruct`, and `google/gemma-3-4b-it` where hardware permits. Inference is served through vLLM (Kwon et al., 2023). The main generation hyperparameters remain conservative across runs, with temperature 0.2 and model-specific output budgets chosen to fit the available A10 infrastructure.

Evaluation. Evaluation uses an LLM judge to assign a binary success score. In the reported runs, the judge is matched to the generator family used in that experiment: Qwen2-VL-2B for the 2B matrix, Qwen2.5-VL for the 3B and 7B matrices, and Gemma 3 4B for the Gemma comparison. The judge sees the disrupted instance, the model output, and the target evaluation rubric. A score of 1 means the candidate directly addresses the disruption, remains feasible, and preserves the goal. A score of 0 means the adaptation fails, ignores

the disruption, relies on the unavailable ingredient, or gives an infeasible plan. The evaluator also applies deterministic guardrail checks for missing-ingredient violations and records judge notes for each example. If the judge output cannot be parsed into the expected schema, the example is recorded as a null judgment rather than coerced into success or failure. We report the mean judge score, number of successes, number of failures, and number of unparsed or null judge outputs.

5 Experimental Details

All experiments are prompting-based, so there is no parameter training, optimizer, learning rate, batch size, epoch count, or early stopping criterion. The shared generation setting across all reported experiments is a conservative temperature of 0.2. The base 2B matrix used a 512-token generation budget, while the later 3B, 7B, and Gemma follow-up runs used smaller model-specific output budgets chosen to fit the available A10 infrastructure. Retrieval hyperparameters were selected from development experiments and then held fixed for the reported test runs: hybrid retrieval, $k = 3$ retrieved examples, and a candidate pool of 20 before optional reranking. The reranker configurations use `BAAI/bge-reranker-base`; non-reranker configurations use the same dense retrieval output directly.

Hyperparameter choices follow two sources. Retrieval settings were chosen from development experiments by comparing alternative retrieval strategies, retrieval depths, and reranker usage, after which the reported matrices fixed hybrid retrieval, a 20-example candidate pool, and $k = 3$ examples in the final prompt. Generation settings were chosen conservatively rather than tuned for maximal score: temperature was fixed at 0.2 across all experiments, and output-token budgets were set manually to values that completed reliably on the available A10 hardware. Thus, the reported comparisons are controlled prompting comparisons rather than fully tuned per-model best-case runs.

The common experiment matrix contains six configurations:

- **Text baseline**: structured text prompt, no image, no retrieval; this isolates procedural reasoning from visual grounding.
- **Vision baseline**: multimodal prompt with image, no retrieval; this measures the value of visual context alone.
- **Text retrieval**: text prompt with retrieved examples; this tests whether analogous prior adaptations help without image input.
- **Vision retrieval**: multimodal prompt with retrieved examples; this combines visual grounding with retrieved analogies.
- **Text retrieval + reranker**: text retrieval with BGE reranking; this tests whether higher retrieval precision improves text-only adaptation.
- **Vision retrieval + reranker**: vision retrieval with BGE reranking; this tests whether reranking helps once both vision and retrieval are available.

All runs filter to YouCook2 examples with disruption type `missing_ingredient`, disruption modality `vision`, and a required image. The same 390 test examples are used for all model-specific comparisons. For retrieval-augmented experiments, the retrieval library is constructed from the training split only, so development and test instances cannot be retrieved at evaluation time. We report binary LLM-judge success rate because the task admits many valid adaptations and exact-match or lexical-overlap metrics would undervalue correct substitutes. We also report paired fixed/broken counts because all configurations are evaluated on the same examples, making it possible to analyze whether a component repairs or damages individual cases. Means are computed over non-null judge outputs; one 2B run produced a single unparsed judgment, so its mean is reported over 389 examples.

Experiments were run on Purdue's Gilbreth cluster using A10 GPUs. Model inference was served through vLLM, and Hugging Face sentence-transformer models were used locally for dense retrieval and reranking. Individual completed runs processed 390 examples, with typical run times on the order of tens of minutes after vLLM startup for the smaller completed matrices. The software stack used Python, PyTorch, vLLM, Hugging Face Transformers, and sentence-transformers. All reported results are single-seed runs with seed 42, so we do not report multi-seed variance or confidence intervals.

The six prompting conditions function as the main internal baselines for this study. Text baseline and vision baseline measure whether image input helps without retrieval; text retrieval and vision retrieval test the value of analogous adaptation cases

with and without image grounding; and the two reranker conditions test whether a stronger second-stage relevance model improves over raw dense retrieval. The precursor text-only system, the 3B scaling study, and the Gemma cross-family runs then serve as higher-level comparisons that test whether the main 2B pattern survives changes in model size and model family.

6 Results and Analysis

6.1 Precursor Text-Only Results

Before the main VLM study, we ran a precursor text-only procedural adaptation pipeline on 334 YouCook2 missing-ingredient test examples using heuristic evaluation metrics. The precursor runs used the language-only generator `mistral-large:123b`, a `qwen3-embedding:8b` retriever, and the reranker `hf.co/jinaai/jina-reranker-v3-GGUF:BF16`. These earlier runs are not directly comparable to the later VLM results because they use a different generator family, a broader text-only setup, and heuristic evaluation rather than the later LLM judge. Even in this text-only setting, retrieval improved adaptation quality and helpfulness while sharply reducing constraint violations relative to the no-retrieval baseline. The main retrieval run improved adaptation quality from 0.452 to 0.464 and helpfulness from 0.537 to 0.601, while reducing constraint violation from 0.029 to 0.006. Reranking did not provide a consistent gain, and retrieval-strategy differences were modest, with hybrid retrieval slightly outperforming the disruption-only and full-context variants. These precursor runs motivated the later visually grounded study and suggested early on that retrieval was more important than prompt-only variation.

Run	Constr.	Feas.	Adapt.	Help.
No retrieval	0.0287	0.8114	0.4515	0.5365
Main retrieval	0.0057	0.7775	0.4643	0.6009
No reranker	0.0054	0.7629	0.4531	0.6009
Disruption-only retrieval	0.0063	0.7734	0.4608	0.6004
Full-context retrieval	0.0135	0.7719	0.4622	0.5997
Hybrid retrieval	0.0099	0.7665	0.4645	0.6001

Table 3: Representative results from the precursor text-only pipeline on 334 YouCook2 missing-ingredient test examples. Lower constraint violation is better; higher values are better for the other metrics. These heuristic scores are reported separately from the later VLM results because they use a different evaluation protocol.

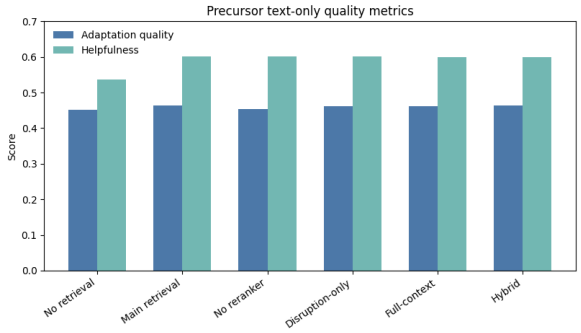

Figure 3: Precursor text-only results on 334 YouCook2 missing-ingredient test examples. Retrieval variants improve adaptation quality and helpfulness relative to the no-retrieval baseline.

6.2 Main Test Results

Table 4 reports the main 2B held-out test results. The strongest configuration is text retrieval, with 256 successes out of 390 judged examples and a mean judge score of 0.656. Vision retrieval is second, with 248 successes out of 389 parsed judgments and a mean score of 0.638. The six configurations also function as a compact ablation study. Text baseline versus vision baseline isolates the contribution of image input, text baseline versus text retrieval isolates the contribution of retrieval without vision, vision baseline versus vision retrieval isolates retrieval on top of visual grounding, and each retrieval setting versus its reranked counterpart isolates the effect of reranking.

Figure 5 visualizes the same comparison as success rates for the main 2B matrix. The plot highlights the main pattern: retrieval produces the largest improvement, vision helps relative to the text baseline, and reranking does not improve over raw retrieval.

The text baseline solves 137 of 390 examples. Adding visual context improves the no-retrieval baseline to 163 successes, a gain of 26 examples. In the 2B Qwen setting, this shows that the image helps the generator identify or reason about the current state, but the gain is modest relative to retrieval.

Retrieval produces the largest improvement. Text retrieval increases performance from 137 to 256 successes, a gain of 119 examples over the text baseline. Vision retrieval improves from 163 to 248 successes, a gain of 85 examples over the

Configuration	Success	Failure	Judge n	Nulls	Mean judge
Text baseline	137	253	390	0	0.351
Vision baseline	163	227	390	0	0.418
Text retrieval	256	134	390	0	0.656
Vision retrieval	248	141	389	1	0.638
Text retrieval + reranker	234	156	390	0	0.600
Vision retrieval + reranker	238	152	390	0	0.610

Table 4: Main Qwen2-VL-2B test results on the 390-example visual missing-ingredient test set. Mean judge is the average binary LLM-judge score over non-null judgments.

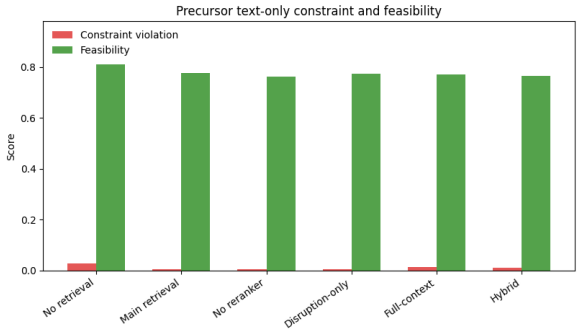

Figure 4: Precursor text-only constraint and feasibility results. Retrieval variants generally reduce constraint violations relative to the no-retrieval baseline while keeping feasibility high.

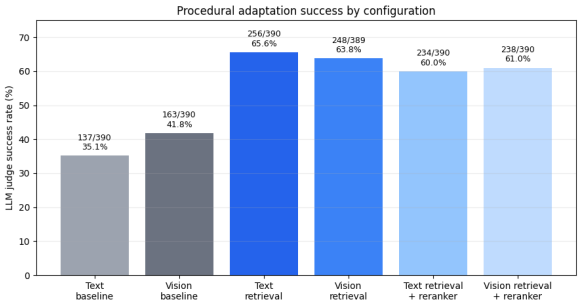

Figure 5: LLM-judge success rate across the six main Qwen2-VL-2B configurations. Retrieval provides the largest improvement at 2B, while reranking does not improve over raw retrieval.

vision baseline. These results suggest that retrieved adaptation cases provide directly useful procedural analogies for handling missing ingredients.

Reranking does not help in the main 2B setup. Text retrieval with reranking drops from 256 to 234 successes, and vision retrieval with reranking drops from 248 to 238 successes. This indicates that the reranker may select examples that are semantically similar to the query but less useful for adaptation.

6.3 Model Scaling: 2B vs. 3B

We additionally repeated the full six-condition matrix with `Qwen/Qwen2.5-VL-3B-Instruct` to test whether the 2B findings were a small-model artifact. Table 5 shows that scaling the generator changed the relative value of retrieval and vision. The 3B model substantially improved both no-retrieval baselines: text baseline rose from 137 to 217 successes and vision baseline rose from 163 to 243 successes. However, the strongest 3B configuration was no longer text retrieval. Instead, the best 3B system was vision retrieval with reranking, which achieved 273 successes and a mean judge score of 0.700.

These scaling results suggest that larger VLMs use visual context more effectively than the 2B model. At 2B, text retrieval was the strongest condition, implying that procedural analogies compensated for a weak base model. At 3B, visual grounding became much more valuable, and vision-conditioned retrieval overtook text-only retrieval. Interestingly, text retrieval and text retrieval with reranking both declined relative to their 2B counterparts, which suggests that retrieved text cases can over-constrain or distract a stronger generator when image input is absent. By contrast, reranking became beneficial in the vision setting, improving vision retrieval from 270 to 273 successes. This indicates that reranking is not uniformly harmful, but instead interacts with model capacity and modality.

6.4 Cross-Family Check: Gemma 3 4B

To test whether the 3B result was specific to the Qwen family, we also ran a cross-family comparison with `google/gemma-3-4b-it` from the Gemma 3 family (Gemma Team et al., 2025) on the same 390-example test set. On the available single-A10 setup, four of the six Gemma conditions completed successfully: text baseline, vision baseline, text retrieval, and vision retrieval. The two reranker conditions failed before generation because the `BAAI/bge-reranker-base` cross-encoder could not be colocated on the same GPU after the Gemma generator server had already loaded. Thus, the missing Gemma reranker points reflect a

Setting	2B Succ.	3B Succ.	Δ Succ.	2B Mean	3B Mean	Δ Mean
Text baseline	137	217	+80	0.351	0.562	+0.210
Vision baseline	163	243	+80	0.418	0.627	+0.209
Text retrieval	256	226	-30	0.656	0.582	-0.074
Vision retrieval	248	270	+22	0.638	0.692	+0.055
Text retrieval + reranker	234	197	-37	0.600	0.509	-0.091
Vision retrieval + reranker	238	273	+35	0.610	0.700	+0.090

Table 5: Scaling from `Qwen2-VL-2B-Instruct` to `Qwen2.5-VL-3B-Instruct`. The larger model substantially improves the no-retrieval baselines and shifts the strongest configuration from text retrieval to vision retrieval with reranking.

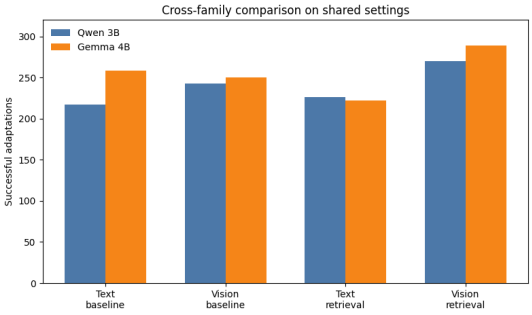

Figure 6: Cross-family comparison between Qwen 3B and Gemma 4B on four shared settings. Both model families show improved performance with vision-grounded retrieval compared to text-only retrieval.

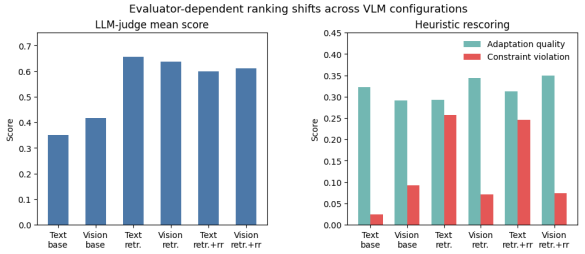

Figure 7: Comparison between the LLM-judge results and the secondary heuristic rescoring across the six VLM configurations. The LLM judge ranks text retrieval highest at 2B, whereas the heuristic evaluator penalizes the text-retrieval variants more heavily for constraint violations and favors the vision-retrieval variants.

hardware capacity limit rather than a generation or evaluation failure.

The completed Gemma runs show a clear and useful pattern. Vision retrieval is the strongest Gemma configuration, achieving 289 successes and a mean judge score of 0.699. This slightly exceeds the corresponding Qwen 3B vision-retrieval condition, which achieved 270 successes and a mean of 0.692. By contrast, Gemma text retrieval performs worse than either Gemma baseline, dropping to 222 successes and a mean judge score of 0.484. This makes the cross-family result stronger rather than weaker: both Qwen 3B and Gemma 4B favor vision-grounded retrieval, while text-only retrieval does not transfer reliably across model families.

6.5 Secondary Heuristic Rescoring

To test whether the main conclusions depend on the LLM judge, we also rescored the main 2B VLM outputs with the heuristic evaluator from the earlier text-only pipeline. This evaluator computes constraint-violation, feasibility, and adaptation-quality scores from the generated output text itself. The resulting ranking does not fully match the LLM-judge ordering. In particular, the heuristic evaluator favors the vision-retrieval variants more strongly and penalizes the text-retrieval variants for substantially higher estimated constraint violations. We interpret this discrepancy as evidence that the heuristic scorer is more sensitive to explicit mentions of unavailable ingredients and more favorable toward conservative outputs, whereas the LLM judge better captures whether an adaptation actually resolves the disruption while preserving the intended dish.

6.6 Paired Comparisons

Because all configurations use the same test examples, paired comparisons identify how often one configuration fixes or breaks examples relative to another. Table 8 summarizes these changes.

Figure 8 presents the same paired comparisons as net changes for the main 2B matrix. The paired results sharpen the aggregate findings. Vision helps, but it also breaks some cases that the text baseline solved. Retrieval has a much larger positive margin, especially in the text condition. Reranking has negative net effects in both text and vision settings, despite fixing some individual examples.

Setting	Qwen 3B Succ.	Gemma 4B Succ.	Δ Succ.	Qwen 3B Mean	Gemma 4B Mean	Δ Mean
Text baseline	217	258	+41	0.562	0.594	+0.032
Vision baseline	243	250	+7	0.627	0.568	-0.059
Text retrieval	226	222	-4	0.582	0.484	-0.098
Vision retrieval	270	289	+19	0.692	0.699	+0.006

Table 6: Cross-family comparison between `Qwen2.5-VL-3B-Instruct` and `Gemma 3 4B`. We attempted the full six-condition Gemma matrix, but the two reranker conditions exceeded the single-A10 GPU memory budget when the reranker was loaded alongside the Gemma generator. Among the completed conditions, Gemma confirms the strongest 3B Qwen trend: vision-grounded retrieval is the best-performing setting.

Config	Constr.	Feas.	Adapt.	Judge
Text baseline	0.0238	0.9723	0.3222	0.351
Vision baseline	0.0931	0.9067	0.2917	0.418
Text retrieval	0.2577	0.6623	0.2932	0.656
Vision retrieval	0.0715	0.8279	0.3437	0.638
Text retr. + rerank	0.2454	0.6831	0.3121	0.600
Vision retr. + rerank	0.0746	0.8249	0.3489	0.610

Table 7: Secondary heuristic rescoring of the main 2B VLM outputs. Lower constraint violation is better; higher feasibility, adaptation quality, and judge score are better. The heuristic evaluator favors the vision-retrieval variants more strongly than the LLM judge.

Comparison	Fixed	Broke	Net
Vision vs. text baseline	82	56	+26
Text retrieval vs. text baseline	160	41	+119
Vision retrieval vs. vision baseline	129	44	+85
Text reranker vs. text retrieval	56	78	-22
Vision reranker vs. vision retrieval	49	59	-10

Table 8: Paired changes across the same 390 test examples. "Fixed" counts examples changed from failure to success; "broke" counts examples changed from success to failure.

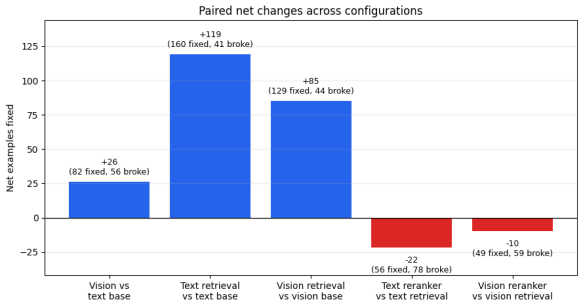

Figure 8: Net paired changes across configurations. Positive values indicate more failures fixed than successes broken; negative values indicate a net regression.

6.7 Qualitative Error Patterns

Manual inspection of outputs and judge notes revealed four common patterns. Table 9 summarizes representative cases from the main 2B test runs.

First, baseline outputs often continue the original step without giving a concrete substitute. This failure is especially common when the disrupted step names a common ingredient such as water, salt, oil, or butter.

Second, retrieval helps by injecting explicit substitution patterns. For example, retrieved cases often contain templates such as replacing milk with water or plant milk, replacing butter with oil, or replacing olive oil with another neutral oil. These analogies make it easier for the model to produce an actionable adaptation.

Third, reranking can break otherwise useful retrieval contexts. In some cases, the raw retrieved examples support a valid adaptation, but the reranked set pushes the generator toward a superficially similar yet constraint-violating continuation. This matches the paired results, where reranking fixes some examples but breaks more than it fixes in the 2B setting.

Fourth, some judge failures are likely false negatives caused by conservative guardrail checks. For example, one output diagnosed missing salt and proposed using soy sauce as a substitute, but the guardrail still marked it as a failure because the output mentioned the missing ingredient. This suggests that the reported scores may underestimate valid adaptations, especially for retrieval systems that explicitly discuss the missing ingredient before replacing it.

7 Conclusion and Discussion

This project evaluates procedural adaptation under visual missing-ingredient disruptions using a YouCook2-derived benchmark and a family of open vision-language models. The 2B Qwen experiments showed that retrieval was the strongest inter-

Pattern	Example	Observed behavior	Interpretation
Text retrieval fixes baseline failure	`01ae801d9f222d50_5`	The text baseline fails on a missing-salt disruption, while text retrieval proposes replacing salt with a small amount of soy sauce or omitting it.	Retrieved cases provide explicit substitution language that the baseline often lacks.
Vision fixes baseline failure	`00f74d3a9f32143e_9`	The text baseline fails on a missing-pepper disruption, while the vision baseline proposes paprika as a substitute.	Visual grounding can help the model produce a concrete adaptation even without retrieval.
Reranking breaks a vision-retrieval success	`01ae801d9f222d50_3`	Vision retrieval succeeds, but the reranked vision run returns a final plan that still says to heat oil in a pan despite oil being unavailable.	Reranking can select examples that are relevant on the surface but do not lead to a constraint-respecting final plan.
Conservative guardrail failure	`03078f7ab0a7257a_1`	The model says to replace unavailable water with broth, stock, or another cooking liquid, but the guardrail still marks the case as using water.	The reported scores are likely conservative because some valid replacements mention the unavailable ingredient while explaining the substitution.

Table 9: Representative qualitative examples from the main Qwen2-VL-2B test runs on the 390-example visual missing-ingredient test set.

vention, especially in the text-only setting. However, the 3B Qwen scaling study refined that conclusion: while retrieval remained valuable, larger model capacity substantially improved the vision-conditioned baselines and shifted the best overall configuration to vision retrieval with reranking. A cross-family Gemma 3 4B comparison further strengthened the main trend. Although the Gemma reranker conditions could not be completed on a single A10 due to GPU memory limits, the completed Gemma runs showed that vision retrieval again outperformed both baselines and clearly outperformed text retrieval. The broader takeaway is that the relative value of retrieval, vision, and reranking depends on model capacity and model family rather than being fixed across scales.

7.1 Why The Best Configuration Changes With Scale

At 2B, the best system is text retrieval rather than vision retrieval. This does not mean visual information is useless; the vision baseline is still stronger than the text baseline. Instead, it suggests that a small VLM is often bottlenecked by knowing a suitable procedural workaround, so retrieved cases provide the most useful extra information. At 3B, the picture changes: both non-retrieval baselines improve sharply, and the strongest setting becomes vision retrieval with reranking. This suggests that larger models can make better use of visual context, so retrieval works best when it complements rather than replaces visual grounding. The text-based conditions remain important because they show that the influence of retrieval is scale-dependent rather than universally dominant.

7.2 Why Reranking Changes With Scale

At 2B, reranking likely optimizes for query-document relevance rather than for the downstream usefulness of an adaptation. A reranked example can share the same dish, ingredient, or surface context while containing a less appropriate workaround. In contrast, raw dense retrieval may preserve more diverse adaptation cases. This is consistent with the 2B paired results, where reranking fixes some examples but breaks more than it fixes. At 3B, however, reranking helps in the vision setting while still hurting in the text setting. This suggests that reranking quality interacts with both modality and model capacity: once the generator is strong enough to integrate image context effectively, a more selective retrieved set can become helpful. The Gemma experiments add an important practical qualification: even when reranking may be conceptually useful, deploying a VLM generator and a cross-encoder reranker on the same 24 GB A10 can itself be infeasible.

7.3 Practical Implications

For procedural assistants, these results suggest that a compact retrieval library of prior adaptations can substantially improve robustness without fine-tuning the generator, but the best way to use re-

trieval depends on the base model. Smaller VLMs benefit strongly from explicit procedural analogies, while somewhat larger VLMs appear to extract more value from visual grounding and can combine that grounding with retrieved examples more effectively. However, the retrieval objective should be aligned with adaptation usefulness, not only semantic similarity.

7.4 Future Work

Future work should address five concrete directions. First, human evaluation would help validate the LLM judge and calibrate the conservative guardrail, especially for cases where a model mentions an unavailable ingredient while proposing a valid substitute. Second, retrieval and reranking objectives should be redesigned around adaptation usefulness rather than generic semantic similarity. Third, the benchmark should be extended beyond missing ingredients to other disruption types, including missing tools, failed steps, incorrect objects, and environmental constraints. Fourth, the scaling results should be pushed further within and across families. In particular, a natural next step is to complete the full six-condition Gemma matrix by moving the reranker off the generator GPU or by using higher-memory hardware, so that Gemma can be compared to Qwen under fully matched conditions. Fifth, broader cross-family multimodal comparisons should be added, including additional open VLMs such as DeepSeek-VL2 (Wu et al., 2024), PaliGemma 2 (Google, 2024), and other vision-language model families once the required hardware and serving constraints are addressed.

8 Limitations

This project has several limitations. First, evaluation uses an LLM judge rather than human evaluation. The judge provides scalable and consistent scoring, but procedural adaptation admits multiple valid answers, and human judgments would better capture practical usefulness.

Second, the missing-ingredient guardrail is conservative. It can mark outputs as failures when they mention the missing ingredient even while proposing a valid substitute. This makes the final scores useful for comparison, but likely conservative in absolute terms.

Third, evaluator choice matters. The secondary heuristic rescoring of the main 2B VLM outputs did not produce the same ranking as the LLM judge, especially for the text-retrieval variants. This suggests that some conclusions are robust only at the level of broad trends, and that finer-grained comparisons depend on what kinds of errors the evaluator is designed to penalize.

Fourth, the reported experiments focus on cooking procedures from YouCook2 and on missing-ingredient visual disruptions. The results may not generalize automatically to other domains, disruption types, or higher-stakes settings.

Fifth, although we added a full `Qwen2.5-VL-3B-Instruct` scaling comparison and a partial cross-family comparison with `Gemma 3 4B`, the study still covers only a narrow slice of the model landscape. Other VLM families and larger-scale evaluations may show different trade-offs between vision, retrieval, and reranking.

Sixth, the infrastructure budget constrained which comparisons could be completed. On a single A10 GPU, the Gemma baseline and retrieval conditions ran successfully, but the two Gemma reranker conditions exceeded the memory budget when the reranker was loaded alongside the VLM generator. This means the current cross-family comparison is informative but not perfectly symmetric across all six conditions.

Finally, the reranking result should be interpreted as evidence about the tested reranker and query design, not as a general claim that reranking is always harmful. A reranker trained or prompted specifically for adaptation usefulness may behave differently.

Acknowledgments

We thank Masudur Rahman for mentorship, feedback, and guidance throughout this project, including helpful suggestions on the experimental design, refinement of our ideas, and feedback on the presentation of the work. We also acknowledge Purdue's Gilbreth cluster for providing the compute resources used in this project.

References

Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Cristian Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Danny Ho, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Eric Jang, Ruben Ruano, Kai Jeffrey, and 20 others. 2022. Do as I can, not as I say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, and 12 others. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901.

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, Louis Rouillard, Thomas Mesnard, Geoffrey Cideron, Jean-Bastien Grill, Sabela Ramos, Edouard Yvinec, Michelle Casbon, Etienne Pot, Ivo Penchev, and 3 others. 2025. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*.

Google. 2024. Paligemma 2 model card. https://huggingface.co/google/paligemma2-3b-pt-448. Hugging Face model card.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pages 6769–6781.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with PagedAttention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*.

Patrick Lewis, Ethan Perez, Aleksandras Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems*, volume 33, pages 9459–9474.

Rodrigo Nogueira and Kyunghyun Cho. 2019. Passage re-ranking with BERT. In *arXiv preprint arXiv:1901.04085*.

Mohit Shridhar, Jesse Thomason, Daniel Gordon, Yonatan Bisk, Winson Han, Roozbeh Mottaghi, Dieter Fox, and Ali Farhadi. 2020. ALFRED: A benchmark for interpreting grounded instructions for everyday tasks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10740–10749.

Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Yang Fan, Kai Dang, Mengfei Du, Xuancheng Ren, Rui Men, Dayiheng Liu, Chang Zhou, Jingren Zhou, and Junyang Lin. 2024. Qwen2-VL: Enhancing vision-language model's perception of the world at any resolution. *arXiv preprint arXiv:2409.12191*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, pages 24824–24837.

Zhiyu Wu, Xiaokang Chen, Zizheng Pan, Xingchao Liu, Wen Liu, Damai Dai, Huazuo Gao, Yiyang Ma, Chengyue Wu, Bingxuan Wang, Zhenda Xie, Yu Wu, Kai Hu, Jiawei Wang, Yaofeng Sun, Yukun Li, Yishi Piao, Kang Guan, Aixin Liu, and 8 others. 2024. DeepSeek-VL2: Mixture-of-experts vision-language models for advanced multimodal understanding. *arXiv preprint arXiv:2412.10302*.

Shitao Xiao, Zheng Liu, Peitian Zhang, and Niklas Muennighoff. 2023. C-Pack: Packaged resources to advance general Chinese embedding. *arXiv preprint arXiv:2309.07597*.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas Griffiths, Yuan Cao, and Karthik Narasimhan. 2023a. Tree of thoughts: Deliberate problem solving with large language models. In *Advances in Neural Information Processing Systems*, volume 36, pages 11809–11822.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023b. ReAct: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*.

Luowei Zhou, Chenliang Xu, and Jason J. Corso. 2018. Towards automatic learning of procedures from web instructional videos. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 7590–7598.