

Tool Quality, Not Framework Complexity: Evaluating RLM Vision, Code, and Memory Pipelines for Radiology Applications

Jonas Schewior, Ansari Hamid Raja Gulam,
Mohamed Ashfaq Anwerdeen, Andreas Wendemuth

Cognitive Systems, Otto von Guericke University Magdeburg, Germany
{jonas.schewior, hamid.ansari, mohamed.anwerdeen, andreas.wendemuth}@ovgu.de

Abstract

It remains unclear whether agentic augmentation (iterative tool use, code execution, or retrieved memory) improves over single-pass VLM inference for medical image interpretation. We present a *systematic quantitative* evaluation of the Recursive Language Model (RLM) framework on a multimodal medical task, comparing three pipeline architectures (P_1 direct vision, P_2 iterative RLM tool use, P_3 model-written code agent) across four retrieval-augmented memory conditions on 100 CheXpert Plus chest X-rays, scoring with CheXbert (14-label classification) and GREEN (radiology-specific LLM judge). Three findings emerge: (1) direct vision outperforms both agentic variants; the RLM loop consumes $7\text{--}9\times$ more tokens without adding task-relevant information when its only tool calls the same VLM; (2) retrieval consistently helps, but the optimal pool size depends on metric and pipeline: top-5 is the sweet spot for direct-vision report quality (GREEN +37 %), while 24 retrieved cases further benefit pathology classification (CheXbert +50 %) and can improve GREEN for agentic pipelines; and (3) qualitative analysis shows code agents with a pretrained classifier succeed when it returns confident pathology scores but fail on normal cases where it returns noise-floor outputs, illustrating that *tool quality, not framework complexity, determines outcome*.

1 Introduction

Conversational AI that reasons about medical images must produce not just classification labels but coherent, grounded explanations: a physician trainee asking “*What findings are visible in this chest X-ray?*” needs an answer that walks through the evidence and connects observations to clinical knowledge. Delivering this reliably motivates a fundamental architectural question: what kind of image-reasoning pipeline should underpin a medical AI agent?

Vision-language models (VLMs) such as MedGemma (Sønderby et al., 2026) have demonstrated impressive performance on medical image understanding benchmarks, processing images and generating structured radiology report text in a single forward pass. However, this passive, one-shot inference paradigm has well-known limitations: models cannot revisit their own analysis, cannot invoke external tools to inspect specific image regions, and cannot recover from early misinterpretations. At the same time, agentic frameworks, systems where LLMs iteratively plan, execute actions, and refine outputs, have produced substantial gains on complex reasoning tasks (Yao et al., 2023; Shinn et al., 2023). Yet these two developments have rarely been brought together in a controlled evaluation, particularly in the medical imaging domain.

The Recursive Language Model (RLM) (Zhang et al., 2026) is a recently proposed framework that treats the prompt as a variable in an external Python REPL environment, allowing a language model to programmatically inspect, decompose, and recursively process its inputs. RLM achieves state-of-the-art performance on long-context benchmarks requiring sustained reasoning, and its REPL environment naturally supports tool use via arbitrary Python code. Crucially, RLM has to date been evaluated on text-based tasks; its application to multimodal or medical domains is largely unexplored.

Concurrent with our work, Mayumu et al. (2026) propose RVLM, extending RLM with vision-specific primitives and an adaptive iteration-depth router, with qualitative evaluation on brain MRI and chest X-ray. We complement this with a systematic quantitative ablation over pipeline architectures and retrieval-augmented memory (Lewis et al., 2020) for radiology report generation. We compare three architectures: P_1 (direct vision, non-agentic baseline), P_2 (VLM inside the RLM loop

with an `analyze_image` tool that issues vision API calls iteratively), and P_3 (model-written code agent with full Python REPL access, optionally guided toward `torchxrayvision` for quantitative pixel-level analysis).

We cross both models and all three pipelines with four memory conditions, ranging from no retrieval to whole-pool context of 99 cases, yielding a complete $2 \times 4 \times 4$ experimental matrix of 32 cells (treating P_3 -base and P_3 -xrv as two prompt-level ablations of P_3). Experiments are run on a 100-patient subset of CheXpert Plus (Chambon et al., 2024), an extension of CheXpert (Irvin et al., 2019), across two VLMs: MedGemma-27B (Sellersgren et al., 2026) and Gemma-3-27B (Google DeepMind, 2025). Generated reports are evaluated with CheXbert (Smit et al., 2020) (14-label classification) and GREEN (Ostmeier et al., 2024) (radiology-specific LLM judge).

2 Related Work

Agentic LLM frameworks. Yao et al. (2023) introduced ReAct, which interleaves reasoning traces with action steps, the conceptual ancestor of P_2 ’s iterative vision loop. Shinn et al. (2023) extended this with Reflexion, where agents use verbal memory of past failures for self-correction. Wang et al. (2024) showed that code execution as an action space outperforms text-only agents on multi-step tasks; P_3 embeds this insight in the RLM framework. RLM (Zhang et al., 2026) treats the prompt as a Python variable, enabling recursive sub-problem decomposition in an external REPL, achieving up to 130% relative improvement over context-compaction baselines on long-context text benchmarks.

Code-based visual reasoning. ViperGPT (Surís et al., 2023) and VisProg (Gupta and Kembhavi, 2023) use LLM-generated Python programs that call vision subroutines to answer visual questions without end-to-end multimodal training. P_3 is the medical-domain, RLM-powered counterpart: it replaces single-shot program generation with iterative REPL execution over chest X-rays.

Agentic medical imaging. Erdur et al. (2026) show that an LLM orchestrating external segmentation tools can overcome the lack of 3D reasoning in standard VLMs for brain MRI, motivating our investigation on 2D chest X-rays. Zhou et al. (2024) demonstrate that even frontier mod-

els fall well short of radiologist performance on findings detection, underscoring the need for architectural improvements beyond simply scaling capacity. Bannur et al. (2023) demonstrate that incorporating prior images and reports at training time improves radiology report generation, complementary to our inference-time retrieval approach. MedGemma (Sellersgren et al., 2026), our primary base model, is Google DeepMind’s medical VLM trained on large-scale medical image-text corpora. Concurrently, Mayumu et al. (2026) extend RLM with vision primitives and an adaptive iteration-depth controller (RRouter), with qualitative evaluation on brain MRI and chest X-ray; our work complements this with a controlled quantitative ablation across explicit retrieval-augmented memory conditions.

Evaluation and retrieval. We adopt GREEN (Ostmeier et al., 2024) as our primary metric: a generative LLM judge validated against radiologist agreement that outperforms all prior NLG metrics and is used in the ReXrank leaderboard (Zhang et al., 2024). Jeong et al. (2024), building on the RAG framework (Lewis et al., 2020), showed that retrieved medical context reduces hallucination but degrades quality when retrieved documents are dissimilar, motivating our memory scaling conditions. CheXpert Plus (Chambon et al., 2024) provides paired radiographs and free-text reports; CheXbert (Smit et al., 2020) extracts 14-label pathology annotations from generated text, giving a complementary classification signal to GREEN.

3 System Overview and Pipeline Architectures

We evaluate three pipeline architectures across four retrieval-augmented memory conditions, forming a complete $2 \times 4 \times 4$ experimental matrix of 32 cells (§4). Every pipeline receives the same input, a query chest X-ray image x , a task prompt q , and a (possibly empty) memory context $\mathcal{M}$ of retrieved similar cases, and produces a structured radiology report comprising numbered findings and an impression, following a single locked output format (§3.2). The three pipeline designs differ only in the inference mechanism applied between input and output; model weights are held constant across pipelines. Figure 1 gives an overview of all three designs.

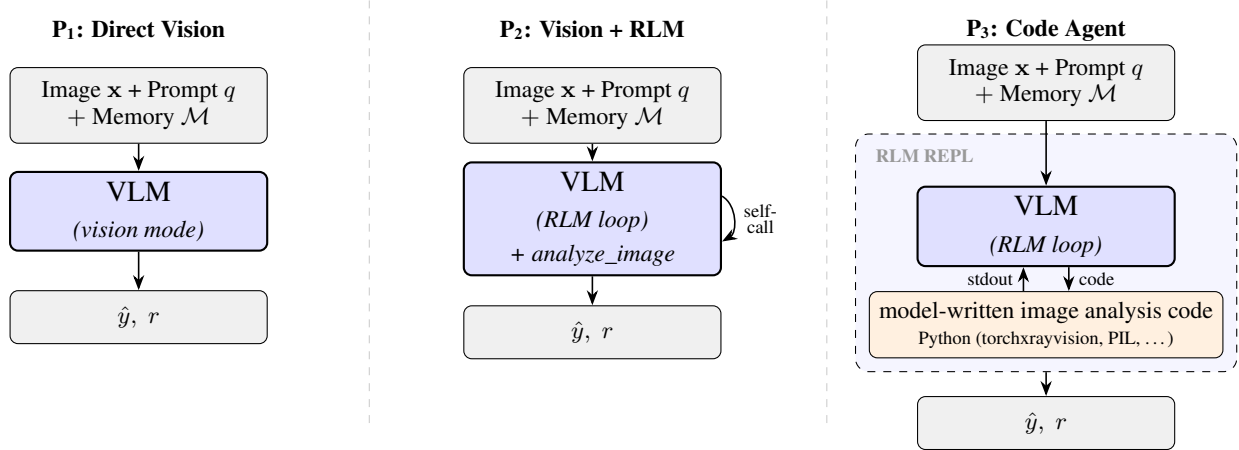


Figure 1: The three pipeline architectures. All receive the same image x , prompt q , and memory context $\mathcal{M}$, and produce numbered findings $\hat{y}$ plus impression r . P₁: single VLM forward pass (baseline). P₂: VLM inside the RLM loop with iterative `analyze_image` self-calls (Zhang et al., 2026). P₃: VLM in RLM REPL with full Python access (e.g. `torchxrayvision`).

3.1 Memory and Retrieval Layer

Before any pipeline is invoked, a memory layer assembles a set of prior cases from the CheXpert Plus validation split into the context $\mathcal{M}$, which is prepended to the task prompt for all three pipelines. Each included case contributes its free-text radiology report to $\mathcal{M}$; patient-level leakage is prevented (no case from the query patient is ever included).

We study four memory conditions (§4.2) that differ in both *how many* cases are included and *how they are selected*. The targeted condition uses MedSigLIP (Sellergren et al., 2026) image embeddings indexed with FAISS (Johnson et al., 2021) to retrieve the five most visually similar cases from the 100-patient dev pool (leave-one-patient-out). The broad-context conditions instead inject a fixed, query-independent set of patients (drawn from a seed-42 random order), presenting them to the model ordered by cosine similarity but without similarity-based filtering.

3.2 Locked Output Format

All three pipelines share a single locked prompt template that instructs the model to produce a structured radiology report: a *Findings* section in which each finding appears on its own numbered line (e.g. *1. Cardiomegaly.*), followed by a concise *Impression*. This format is the single source of truth for all experimental conditions; its integrity is enforced by a byte-level fingerprint assertion at run time. The numbered format serves two purposes: it allows the GREEN radiology judge to identify individual findings unambiguously, and it allows CheXbert to

extract pathology labels from the same generated text, so both metrics score identical outputs. Full verbatim prompt texts for all three pipelines are given in Appendix B.

3.3 P₁: Direct Vision (Baseline)

P₁ is the non-agentic baseline: the query image x , task prompt q , and memory context $\mathcal{M}$ are concatenated and passed to the VLM in a single forward pass, producing the structured radiology report with no iterative refinement, tool use, or self-reflection.

3.4 P₂: Vision + RLM

P₂ places the VLM inside the Recursive Language Model (RLM) framework (Zhang et al., 2026), equipped with an `analyze_image` tool. This tool accepts an image path and returns a three-sentence qualitative description by making a real vision API call (`base64 PNG` $\rightarrow$ model endpoint). Inside the RLM REPL loop, the model calls `analyze_image` one or more times, reasons over the returned text descriptions, and may issue additional calls, for example, to re-examine a specific finding, before producing a final report, when the RLM sets `answer["ready"] = True`.

3.5 P₃: Model-Written Code Agent

P₃ is the most autonomous design. The model has the freedom to write and execute arbitrary Python code inside the RLM REPL, including libraries such as `torchxrayvision` (Cohen et al., 2021), `Pillow`, and `NumPy` for quantitative pixel-level analysis. Critically, no pixel analysis tool

is pre-provided: the model must author all image-reading code itself, deciding which features to compute and how. This tests whether full code-writing autonomy produces richer radiology outputs than either P_1 or P_2 . We evaluate two prompt variants that differ only in the EXAMPLE block seeding the model’s first REPL step. P_3 -base provides a generic numpy/PIL starter that computes pixel statistics (mean intensity, quadrant means, opacity fraction), basic image features the model must interpret as radiological findings. P_3 -xrv instead shows a torchxrayvision DenseNet-121 recipe, guiding the model toward a pretrained chest X-ray classifier as its primary analysis tool. Both variants share identical system-prompt structure, output-format requirements, and kickoff message; only the EXAMPLE block differs (full prompts in Appendix B).

4 Experimental Setup

4.1 Dataset

We use **CheXpert Plus** (Stanford AIMI) (Cham-bon et al., 2024), a large chest radiograph collection pairing frontal-view X-rays with both free-text radiology reports and 14-label CheXbert pathology annotations. All experiments use the frontal-only validation split (202 frontal images from 200 patients), of which we use the first 100 studies in the split’s default patient ordering as a development subset (*dev-100*) for the current evaluation. The ground-truth report text provides the reference for GREEN scoring; the 14 CheXbert labels provide the reference for classification scoring. Label policy follows the CheXpert Plus recommendation: uncertain labels (−1) are treated as positive and unlabelled observations as negative, a conservative convention that maximises recall for rare pathologies and matches the original CheXpert challenge protocol (Irvin et al., 2019).

4.2 Memory Conditions

We study four memory conditions that differ in both *how many* prior cases the model receives and *how those cases are selected* (Table 1). Patient-level leakage is prevented across all conditions: no case from the query patient’s own studies is ever included in the context.

none provides no context (ablation baseline). static_valid retrieves the top-5 most similar cases via FAISS cosine search over the 100-patient dev pool (leave-one-patient-out) (Lewis et al.,

Condition	Selection	Pool	Reports
none	-	0	0
static_valid	Top-5 cosine sim. (FAISS)	100	5
wholepool_quarter	All in seed-42 pool	25	24
wholepool_full	All in seed-42 pool	100	99

Table 1: Memory conditions. *Selection* describes how cases enter the context; *Pool* is the number of candidate patients; *Reports* is the count injected into the prompt. In all non-zero conditions, cases are presented ordered by descending MedSigLIP cosine similarity to the query image, but without numerical scores.

2020), the only condition where pool membership is query-driven. The two wholepool conditions instead inject a fixed, query-independent set of patients drawn from a seed-42 random ordering: wholepool_quarter uses all 24 patients from the first quarter of that order, while wholepool_full uses all 99, testing whether broad reference context illustrating report format and vocabulary, rather than visual similarity, benefits generation (Jeong et al., 2024).

4.3 Models

We evaluate the three pipelines on two vision-language models (Table 2). MedGemma-27B API runs were completed via KISSKI (AI Service Centre for Sensitive and Critical Infrastructures) (medgemma-27b-it); when it was unavailable or for runs requiring longer context windows, models were served locally (cells marked †/‡ in Tables 3 and 4). Specifically, all P_3 -base MedGemma cells (‡) and the wholepool_full cells for P_2 and P_3 -xrv MedGemma (†) used local vLLM; all other MedGemma conditions used the API endpoint. All Gemma-3-27B runs were served locally. Local serving used vLLM (Kwon et al., 2023) v0.8.4 on NVIDIA A40 GPUs (48 GB VRAM) with bitsandbytes INT4 quantization (≈ 14 GB per 27B model). Full serving commands are in Appendix C.

Model	Context	Vision Encoder	Domain
MedGemma-27B	128K	MedSigLIP-400M	Medical imaging
Gemma-3-27B	128K	SigLIP-400M	General

Table 2: Models evaluated. Both share the same 27B parameter count and 128K context window but differ in domain. Both cover all three pipelines and all four memory conditions.

4.4 Evaluation Metrics

CheXbert (classification). We extract 14 pathology labels from each generated report using

CheXbert (Smit et al., 2020), a BERT-based chest X-ray label extractor, and compute micro-F1, macro-F1, and accuracy against the ground-truth CheXpert Plus annotations. CheXbert scoring is deterministic and requires no LLM judge.

GREEN (report quality). We score each generated report against the reference radiologist text using GREEN (Ostmeier et al., 2024): $\text{score} = \text{matched} / (\text{matched} + \text{significant errors})$, where a judge LLM classifies findings into six error categories following the original GREEN prompt template verbatim. We use GREEN-RadLlama2-7b (Ostmeier et al., 2024), the model the GREEN paper was trained and validated against, as our primary judge (temperature 0, served locally on a dedicated A40). All GREEN scores should be interpreted as *relative* comparisons across conditions; see §5 for discussion.

Failure handling. Runs that fail to produce a parseable output (timeout, REPL error, or format violation) are excluded from metric computation; all reported means are over successfully completed runs only. Per-condition completion counts are given in §5; for most conditions this affects only P₃.

Efficiency. We record total token usage and RLM iteration counts for P₂ and P₃.

5 Results

Tables 3 and 4 report GREEN and CheXbert micro-F1 scores across all pipeline–memory–model combinations. Table 5 quantifies the computational cost of each pipeline. We identify seven consistent findings.

Direct vision outperforms agentic augmentation on both metrics. P₁ (direct vision, no RLM loop) achieves the highest GREEN and CheXbert scores for both models across both metrics. For MedGemma-27B, the best single cell, P₁ with top-5 retrieved cases, yields GREEN = 0.303 and CheXbert micro-F1 = 0.506. Introducing the RLM loop in P₂ drops performance (GREEN = 0.172, micro-F1 = 0.441 at the same condition), and the code agent P₃ drops further still on GREEN. The ranking P₁ > P₂ > P₃ holds for MedGemma across both metrics and for Gemma-3 on GREEN; the P₁–P₂ gap is most pronounced at non-zero memory conditions, where bootstrap confidence intervals do not overlap, and narrowest at memory=none (full

per-cell 95% CIs in Figure 2). The exception is Gemma-3 on CheXbert, where P₃-xrv consistently outperforms P₂ across all four memory conditions (e.g. 0.338 vs. 0.298 at none, 0.326 vs. 0.312 at wholepool_quarter), while both remain well below P₁ (0.411).

Code scaffolding helps classification but not text quality. Within the code agent family, providing a torchxrayvision example snippet in the prompt (P₃ xrv) consistently outperforms the base variant (P₃ base) on CheXbert for Gemma-3-27B (the only fair comparison, since both use local vLLM): +9 pp at none (0.338 vs. 0.248), shrinking to +3 pp at wholepool_full (0.324 vs. 0.290). On GREEN, the two variants are virtually indistinguishable (≤ 0.02 difference across all conditions), confirming that a code template aids pathology identification but does not improve the quality of the generated report text.

CheXbert and GREEN diverge most for code-agent pipelines. Across all pipelines, CheXbert F1 scores are substantially higher than GREEN for the same condition (e.g. P₁ + Top-5 MedGemma: CheXbert micro-F1 = 0.506 vs. GREEN = 0.303). This gap is partly structural: CheXbert awards credit for correctly absent labels (true negatives over 14 binary slots), while GREEN scores only positive findings, but the divergence is *largest* for P₃ base, where CheXbert F1 remains non-trivial (~ 0.20 – 0.29) while GREEN approaches zero. The reason is that CheXbert ignores pixel-statistic “findings” (mean intensity, opacity fraction) that carry no pathology label, whereas GREEN’s RadLlama2 judge penalises them as significant errors. This metric asymmetry implies that label-extraction metrics alone can mask systematic output-format failures in code-agent pipelines, motivating the use of a natural-language judge alongside CheXbert when evaluating agentic radiology systems.

Retrieval helps; optimal pool size depends on metric and pipeline. Retrieval consistently improves P₁ on both metrics and P₂ on CheXbert; for P₃ the effect is mixed (Table 4). For P₁ (direct vision) on GREEN, Top-5 (static_valid) is the clear sweet spot: MedGemma rises from 0.221 to 0.303 (+37 %) and Gemma-3 from 0.172 to 0.260 (+51 %); wholepool_full is the weakest non-zero P₁ condition, consistent with prior evidence that low-similarity documents degrade generation (Jeong et al., 2024) (cosine similarity falls

Table 3: GREEN mean scores (judge: GREEN-RadLlama2-7b) across pipeline, model, and memory condition. **Bold** = best cell per model column pair. WP-Qtr = wholepool_quarter (24 reports); WP-Full = wholepool_full (99 reports). MedGemma P₃(base) column used local vLLM throughout (§); cells marked † also used local vLLM. All Gemma-3 cells used local vLLM (see §4).

Memory	P ₁ Direct Vision		P ₂ Vision + RLM		P ₃ Code (base)		P ₃ Code (xrv)	
	MedGemma	Gemma-3	MedGemma	Gemma-3	MedGemma [‡]	Gemma-3	MedGemma	Gemma-3
None	0.221	0.172	0.178	0.138	0.003	0.066	0.045	0.067
Top-5	0.303	0.260	0.172	0.161	0.030	0.065	0.080	0.060
WP-Qtr	0.266	0.248	0.168	0.148	0.023	0.080	0.090	0.063
WP-Full	0.266	0.199	0.220 [†]	0.175	0.028	0.072	0.044 [†]	0.090

Table 4: CheXbert micro-F1 (14-label) across pipeline, model, and memory condition. **Bold** = best cell per model column pair. WP-Qtr / WP-Full, †, ‡ as in Table 3.

Memory	P ₁ Direct Vision		P ₂ Vision + RLM		P ₃ Code (base)		P ₃ Code (xrv)	
	MedGemma	Gemma-3	MedGemma	Gemma-3	MedGemma [‡]	Gemma-3	MedGemma	Gemma-3
None	0.433	0.274	0.409	0.298	0.237	0.248	0.303	0.338
Top-5	0.506	0.394	0.441	0.307	0.204	0.260	0.299	0.329
WP-Qtr	0.496	0.411	0.470	0.312	0.235	0.270	0.236	0.326
WP-Full	0.463	0.325	0.415 [†]	0.303	0.242	0.290	0.230 [†]	0.324

Table 5: Per-query cost at memory=none. Tokens = mean total tokens per query; Time = mean seconds per query. P₃ (base) and P₃ (xrv) rows for Gemma-3 enable a direct comparison (both local vLLM).

Pipeline	Model	Tokens	Time (s)	RLM iters
P ₁ Direct	MedGemma	634	18.5	-
	Gemma-3	588	23.0	-
P ₂ Vis+RLM	MedGemma	5,647	53.3	2.4
	Gemma-3	4,472	135.1	2.0
P ₃ Code (base)	MedGemma [‡]	9,656	322.0	3.0
	Gemma-3	17,882	238.0	6.0
P ₃ Code (xrv)	MedGemma	30,871	188.6	8.7
	Gemma-3	14,790	179.0	5.4

from ~ 0.75 at rank 1 to ~ 0.31 at rank 99). For P₂ and P₃, however, broader context pools are competitive or superior to Top-5 in several conditions (e.g. P₃-xrv MedGemma: WP-Qtr = 0.090 vs. Top-5 = 0.080; P₂ MedGemma: WP-Full = 0.220 vs. Top-5 = 0.172), suggesting agentic pipelines tolerate broader, less targeted context better than direct vision does. CheXbert also benefits from retrieval, and for Gemma-3 from larger pools: MedGemma P₁ micro-F1 rises from 0.433 (none) to 0.506 at Top-5 (+17%), while Gemma-3 P₁ rises from 0.274 to 0.394 at Top-5 (+44%) and further to 0.411 at wholepool_quarter (+50%).

Medical pre-training amplifies report quality. MedGemma-27B consistently outperforms Gemma-3-27B on GREEN across all conditions (e.g. 0.303 vs. 0.260 for P₁ + Top-5), confirming

that domain-specific pre-training is reflected in radiology text quality as judged by the radiological LLM evaluator.

Complexity incurs large computational costs.

Table 5 shows the per-query cost at memory=none. P₁ generates a report in ≈ 600 tokens and 18-23 seconds. P₂ uses ~ 2 -2.4 RLM iterations, consuming 7-9 $\times$ more tokens and 2-6 $\times$ more time. For Gemma-3, P₃ (base) uses 17,882 tokens and 238 s with 6 iterations, while P₃ (xrv) uses 14,790 tokens and 179 s with 5.4 iterations; for Gemma-3 the xrv template reduces token use and latency by ≈ 17 -25%. For MedGemma, the higher iteration count of P₃-xrv (8.7 vs. 3.0) instead *increases* token consumption (30,871 vs. 9,656), driven by context-overload and wrong-fence failure modes detailed in §6. Both code agent variants consume roughly 20-30 $\times$ more tokens than P₁, confirming that P₁ is the Pareto-optimal pipeline across both quality and efficiency dimensions.

Reliability. Generation success rates are high for P₁ ($\geq 99\%$) and P₂ ($\geq 91\%$) across all conditions. P₃-xrv MedGemma-27B shows elevated REPL failures at none (86/100 successful) and wholepool_quarter (91/100), likely due to wrong-fence and repeated-error failures detailed in §6; P₃-base MedGemma-27B is 100% at none but degrades sharply with context length, reaching 70% at wholepool_full. Gemma-3-27B maintains $\geq 95\%$ completion across all P₃ condi-

tions. Failure-adjusted GREEN means (assigning GREEN = 0 to incomplete runs) are within 0.01 of completed-only means for the highest-failure cells (P₃-base MedGemma wholepool_full: 0.028 completed vs. 0.020 adjusted; P₃-xrv MedGemma wholepool_full: 0.046 vs. 0.036), confirming that survivorship bias does not materially inflate the already-low P₃ GREEN scores.

Contextual comparison with published systems.

Our evaluation is not directly comparable to published benchmarks; we use a 100-patient subset, a structured findings-plus-impression output format, and a locally hosted GREEN-RadLlama2-7b judge, but rough context is informative. On the ReXrank CheXpert Plus leaderboard (Zhang et al., 2024), fine-tuned specialist systems achieve GREEN scores of 0.147-0.306 (findings section), with GPT-4V zero-shot at 0.152. Our best zero-shot result of GREEN = 0.303 (P₁+Top-5, MedGemma-27B) falls in the upper range of that leaderboard despite using no task-specific fine-tuning, though the small test set makes the estimate noisy. This comparison should be interpreted with caution: differences in test-set size, output format, fine-tuning status, and possibly GREEN judge version make absolute ranking claims unreliable.

6 Analysis

Reliability gap and failure taxonomy. P₁ achieves 99-100% completion across all conditions; P₂ achieves ≥91%. P₃ reaches only 70-100% depending on model and memory, with MedGemma-27B showing the steepest degradation: completion falls from 100% at none to 70% at wholepool_full for the base variant. Trace inspection reveals two dominant failure modes. The first is *context overload*: at wholepool_full, 10 out of 30 MedGemma P₃-base failures produce zero iterations, the model never emits a first response before timing out during prefill, and a further 15 produce only prose planning text across 2-4 iterations without ever issuing a ````repl` code block, as if attending to 99 prior cases exhausts the model’s effective generation budget. Gemma-3-27B is entirely immune to this pattern, achieving 100% completion under wholepool_full for both P₃ variants, suggesting its context-window handling is more robust than MedGemma’s at this scale. The second failure mode is a *wrong-fence no-op loop*: 11 out of 100 P₃-xrv/MedGemma/none traces emit ````python` delimiters instead of the

required ````repl`, so no code is ever executed despite the model generating syntactically correct Python in every iteration. The model receives no REPL output, reproduces the same response in the next iteration, and exhausts the 11-iteration cap without submitting an answer (P₃-xrv/MedGemma/none, patient 64542).

What P₃ actually computes: pixels versus pathology probabilities.

The two P₃ variants diverge sharply in their code strategies. P₃-base converges on a stereotyped three-iteration pattern for MedGemma: load the image, compute mean intensity and left/right/upper/lower quadrant means and an opacity fraction, then transcribe those pixel statistics into clinical-sounding prose. The resulting reports contain no genuine pathology labels, only reformulations of brightness values (e.g., “significant difference between upper (0.409) and lower (0.542) portions... requires clinical correlation”, patient 64542), which directly explains the near-zero GREEN scores (0.003-0.030) and weak CheXbert F1 (0.20-0.24) for P₃-base/MedGemma. P₃-xrv produces a qualitatively different pipeline: Gemma-3-27B adopts torchxrayvision in 96-100% of traces, correctly decomposing the preprocessing across iterations (load → normalize → crop → infer) and using the DenseNet-121 pathology probability vector as grounding evidence (e.g. patient 64548: Effusion 0.911, Atelectasis 0.896, Cardiomegaly 0.819; P₃-xrv/Gemma-3/none). MedGemma’s torchxrayvision adoption degrades sharply with context length, from 75% at none to 32% at wholepool_quarter, as the growing prior-case text crowds out the example code in the system prompt. An emergent behaviour: P₃-base/Gemma-3 spontaneously imports torchxrayvision in ~35% of static_valid traces despite the numpy-only example prompt, suggesting Gemma-3 infers from the prior-case context that specialised medical imaging libraries are available.

Where the RLM loop breaks: two pathological iteration patterns.

Among P₃-xrv/MedGemma traces with ≥8 iterations, two recurring failure patterns account for the majority of degraded cases. (1) *Wrong-fence no-op* (11 traces): as described above, the model plans torchxrayvision usage in internal reasoning but emits ````python` fences; no code executes and the plan repeats unchanged to the iteration cap. (2) *Repeated-error deadlock* (observed in traces reaching the 11-iteration cap,

	Case A - Pathological (patient 64673) Model: Gemma-3-27B	Case B - Normal/stable (patient 64708) Model: MedGemma-27B
Reference	Globular enlarged cardiac silhouette consistent with <i>cardiomegaly</i> . Left basilar <i>atelectasis</i> . Possible small left <i>pleural effusion</i> .	No interval change. No evidence of pneumonia.
P₁ Direct Vision (memory=None)	GREEN = 0.00 Findings: 1. Right and left chest tubes present. 2. Moderate right pneumothorax. 3. Small left pneumothorax. [...] Cardiac silhouette normal. No pleural effusion. Impression: Bilateral pneumothoraces, likely related to chest tubes.	GREEN = 1.00 Findings: 1. Azygos lobe present. 2. Lungs clear; no effusion, pneumothorax, or consolidation. 3. Right port-a-cath, tip in lower SVC. Impression: Normal chest radiograph except azygos lobe.
P₃-xrv Code Agent (memory=None)	GREEN = 0.60 DenseNet: Cardiomegaly 0.87, Effusion 0.86, Atelectasis 0.82 Findings: 1. Atelectasis. 2. Pleural effusion. 3. Cardiomegaly. 4. Lung opacity. Impression: Findings suggestive of atelectasis, effusion, and cardiomegaly.	GREEN = 0.00 DenseNet: all 18 pathologies $\approx$ 0.50 (noise floor) Findings: 15. Atelectasis. 15. Consolidation. [...] 14 pathologies, all numbered "15," due to model-introduced indexing error] Impression: Multiple potential findings; correlation with clinical findings necessary.

Table 6: Qualitative comparison of P₁ and P₃-xrv on two contrasting cases (memory=None). Case A: DenseNet returns confident pathology scores; P₃-xrv correctly identifies all three reference findings (GREEN = 0.60) while P₁ hallucinates chest tubes (GREEN = 0.00). Case B: DenseNet returns near-0.5 noise-floor outputs for a normal chest; P₃-xrv reports 14 spurious findings (GREEN = 0.00) while P₁ is correct (GREEN = 1.00).

e.g., patient 64718): the model writes `C, H, W = img.shape` after `XRayCenterCrop()`, which returns a 2D (H, W) array for single-channel input, raising `ValueError: not enough values to unpack (expected 3, got 2)`. This error fires identically across 8 consecutive iterations with minimal code variation; the model never diagnoses the fix (`normalize(img, 255)[None, ...]`) that Gemma-3 applies correctly on its first attempt. Gemma-3 avoids both patterns by decomposing the pipeline correctly across iterations and fixing API mismatches on the first attempt.

Model-framework compatibility as a prerequisite. The P₃ reliability gap between MedGemma and Gemma-3 reveals a prerequisite that precedes tool quality: the backbone model must reliably execute the RLM REPL contract. P₃-xrv/MedGemma reaches only 86% completion even at none (zero context pressure), exhibiting the wrong-fence failure described above; Gemma-3 never exhibits this failure. Domain fine-tuning may therefore erode the general instruction-following robustness that the RLM framework, designed around code-executing agents, requires as a prerequisite. This creates a practical tension: the model best suited for medical image interpretation (MedGemma on GREEN) is less reliably capable of operating the REPL loop, suggesting that *model-framework compatibility* is as important as tool quality for agentic deployment.

P₂ as an expensive transcription step. Trace analysis clarifies why P₂ underperforms P₁ despite accessing the same image. In 81-100% of

traces the loop completes in exactly two iterations: call `analyze_image()`, then transcribe the result. Since `analyze_image()` is itself a vision call to the same VLM, P₂ adds no new visual evidence, only a code-agent framing that constrains output toward list-style enumeration rather than direct narrative synthesis, and a transcription step that can drop qualifications from the raw description. P₂ would add value only if its tools could access information unavailable to the base VLM (region-of-interest cropping, DICOM metadata, retrieval), which the current setup does not provide. Table 6 illustrates these dynamics on two concrete patients: DenseNet provides reliable signal when real pathologies are present (Case A, GREEN 0.60 for P₃-xrv vs. 0.00 for P₁), but returns near-0.5 noise-floor outputs for all 18 pathologies on normal cases (Case B). The code agent, lacking any confidence threshold, reports every label as a finding, a failure pattern we term *threshold insensitivity*, inverting the advantage and yielding GREEN = 0.00 where P₁ scores 1.00.

7 Conclusion

We presented a systematic quantitative evaluation of RLM (Zhang et al., 2026) for chest X-ray report generation, concurrent with but complementary to Mayumu et al. (2026).

Our main findings are threefold. First, direct vision (P₁) consistently outperforms agentic alternatives on both GREEN and CheXbert: in P₂, the RLM loop adds overhead without adding task-relevant information when its only tool issues a further call to the same underlying VLM; in P₃ with-

out effective tool guidance, model-written pixel-statistics code similarly fails to improve over single-pass inference. Second, top-5 retrieved memory reliably improves P_1 on *both* metrics (up to +0.08 GREEN, +44 % CheXbert micro-F1). The two metrics point to different optimal budgets: top-5 is the sweet spot for report text quality (larger pools introduce low-similarity noise that degrades GREEN), while ~ 24 retrieved cases can further benefit pathology classification (CheXbert) for non-domain-adapted models. Third, guiding P_3 toward a pretrained chest X-ray classifier (Cohen et al., 2021) improves CheXbert F1 by +9 pp over pixel-statistics baselines, though GREEN remains limited by threshold insensitivity and over-reporting.

These results suggest that RLM’s iterative reasoning framework holds genuine promise for medical imaging, but its benefit is conditioned on tool quality: the bottleneck is not the framework but what the REPL agent can access. Future work should explore better-calibrated code agent prompts, adaptive context truncation for large memory pools, tools providing information unavailable to the base VLM (DICOM metadata, region-of-interest detectors, anatomy segmentation), and iterative pipelines (P_2) equipped with per-region tools to enable step-by-step clinical reasoning for applications such as educational AI scenarios.

Limitations

This study has several limitations that should be considered when interpreting our findings.

Not intended for clinical use. This work is a research evaluation of pipeline architectures for automated radiology report generation; it is **not** intended for, nor validated for, real clinical use or patient care. The underlying models and tools, including torchxrayvision (Cohen et al., 2021), which its authors explicitly caution is not intended for clinical deployment, are research artefacts evaluated on a small held-out set under controlled conditions. Results should be interpreted solely as evidence about pipeline design choices, not as evidence of clinical safety or efficacy.

Scale. We evaluate on a 100-patient subset of CheXpert Plus (frontal views only). While sufficient to demonstrate the methodology, results may not fully generalise to the complete validation set or to lateral-view or multi-view inputs. Future work should look into increasing the patient set used for

the experiments.

LLM-as-judge bias. GREEN relies on GREEN-RadLlama2-7b as the judge LLM. LLM judges are known to exhibit stylistic biases; we mitigate this by using the dedicated judge the GREEN paper was trained and validated against, and by reporting CheXbert classification scores, which are deterministic and require no judge. GREEN scores should be interpreted as relative comparisons across conditions, not absolute benchmarks against published numbers.

Compute cost of agentic pipelines. P_2 and P_3 incur substantially higher inference costs than P_1 due to iterative REPL calls. Use in resource-constrained research settings may favour simpler architectures regardless of accuracy gains; our token and iteration counts quantify this trade-off.

P_2 as a baseline. Our comparison between P_1 and P_2 is not compute-matched: P_2 uses $7\text{--}9\times$ more tokens per query by design. This means the observed quality drop for P_2 cannot be attributed solely to the RLM loop’s reasoning capacity. It also reflects the overhead of a multi-turn framework evaluated in a zero-shot setting without prompt tuning. A more optimised RLM prompt or stopping criterion may narrow the gap. We plan to improve on this in the future.

Tool quality vs. prompt steering in P_3 . P_3 -base and P_3 -xrv share identical system-prompt structure and differ only in their EXAMPLE block: one seeds the model with numpy/PIL pixel statistics, the other with torchxrayvision. The observed performance difference between these variants therefore conflates *tool quality* (which library is available) with *prompt steering* (the example biases the model toward a particular code strategy). Future ablations should hold the EXAMPLE constant while varying tool availability to disentangle these effects.

Reliance on an unreviewed framework. The RLM framework (Zhang et al., 2026) is, at the time of writing, an arXiv preprint that has not yet undergone formal peer review and is still being actively revised. We therefore greatly encourage independent replication as the framework matures.

Ethics Statement

This work adheres to the [ACL Code of Ethics](#).

Data. All experiments use CheXpert Plus (Chambon et al., 2024), a publicly available, de-identified chest radiograph dataset released by Stanford AIMI under its standard data-use agreement. No patient-identifiable information was accessed, collected, or stored at any point. No human subjects were recruited, annotated, or otherwise involved in this research.

Models and tools. We evaluate publicly available pre-trained models (MedGemma-27B, Gemma-3-27B, GREEN-RadLlama2-7b, torchxrayvision DenseNet-121) accessed through their respective published APIs or model hubs under their stated licences. No new model training or fine-tuning was performed.

Environmental impact. Running 32 experimental cells on 100 patients each on A40 GPUs incurred a non-trivial but modest compute footprint; we report token counts and runtimes in Table 5 to support reproducibility and cost estimation for follow-on work.

Acknowledgments

This work was supported by KISSKI (AI Service Centre for Sensitive and Critical Infrastructures, <https://kisski.gwdg.de>), which provided the compute infrastructure used to run the experiments.

The authors used **Claude Sonnet 4.6** (Anthropic, 2026) as an AI-assisted writing and coding tool during the preparation of this manuscript, for drafting and refining text, generating L^AT_EX code, and supporting literature organisation. All scientific contributions, experimental design, analysis, and conclusions are entirely the authors’ own work.

References

- Anthropic. 2026. Claude Sonnet 4.6. <https://www.anthropic.com>. Large language model.
- Shruthi Bannur, Stephanie Hyland, Qianchu Liu, Fernando Pérez-García, Maximilian Ilse, Daniel C. Castro, Benedikt Boecking, Harshita Sharma, Kenza Bouzid, Anja Thieme, Anton Schwaighofer, Maria Wetscherek, Matthew P. Lungren, Aditya Nori, Javier Alvarez-Valle, and Ozan Oktay. 2023. *Learning to Exploit Temporal Structure for Biomedical Vision-Language Processing*. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15016–15027, Vancouver, BC, Canada. IEEE.
- Pierre Chambon, Jean-Benoit Delbrouck, Thomas Sounack, Shih-Cheng Huang, Zhihong Chen, Maya Varma, Steven QH Truong, Chu The Chuong, and Curtis P. Langlotz. 2024. *CheXpert plus: Augmenting a large chest x-ray dataset with text radiology reports, patient demographics and additional image formats*. *Preprint*, arXiv:2405.19538.
- Joseph Paul Cohen, Joseph D. Viviano, Paul Bertin, Paul Morrison, Parsa Torabian, Matteo Guarrera, Matthew P. Lungren, Akshay Chaudhari, Rupert Brooks, Mohammad Hashir, and Hadrien Bertrand. 2021. *TorchXRayVision: A library of chest X-ray datasets and models*. *arXiv preprint*. ArXiv:2111.00595 [eess.IV].
- Ayhan Can Erdur, Daniel Scholz, Jiazhen Pan, Benedikt Wiestler, Daniel Rueckert, and Jan C. Peeken. 2026. *Agentic Large Language Models for Training-Free Neuro-Radiological Image Analysis*. *arXiv preprint*. ArXiv:2604.16729 [cs.CV].
- Google DeepMind. 2025. Gemma 3 model card. https://ai.google.dev/gemma/docs/core/model_card_3. Accessed: 2026-06-13.
- Tanmay Gupta and Aniruddha Kembhavi. 2023. *Visual Programming: Compositional visual reasoning without training*. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14953–14962, Vancouver, BC, Canada. IEEE.
- Jeremy Irvin, Pranav Rajpurkar, Michael Ko, Yifan Yu, Silvana Ciurea-Ilcus, Chris Chute, Henrik Marklund, Behzad Haghgoo, Robyn Ball, Katie Shpanskaya, Jayne Seekins, David A. Mong, Safwan S. Halabi, Jesse K. Sandberg, Ricky Jones, David B. Larson, Curtis P. Langlotz, Bhavik N. Patel, Matthew P. Lungren, and Andrew Y. Ng. 2019. *CheXpert: A Large Chest Radiograph Dataset with Uncertainty Labels and Expert Comparison*. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):590–597.
- Minbyul Jeong, Jiwoong Sohn, Mujeen Sung, and Jae-woo Kang. 2024. *Improving medical reasoning through retrieval and self-reflection with retrieval-augmented large language models*. *Bioinformatics*, 40(Supplement_1):i119–i129.
- Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2021. *Billion-scale similarity search with GPUs*. *IEEE Transactions on Big Data*, 7(3):535–547. ArXiv:1702.08734.
- 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 29th Symposium on Operating Systems Principles (SOSP)*.
- Patrick Lewis, Ethan Perez, Aleksandra 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 (NeurIPS)*.

- Nicanor Mayumu, Zeenath Khan, Melodena Stephens, Patrick Mukala, and Farhad Oroumchian. 2026. [RVLM: Recursive Vision-Language Models with Adaptive Depth](#). *arXiv preprint*. ArXiv:2603.24224 [cs.CV].
- Sophie Ostmeier, Justin Xu, Zhihong Chen, Maya Varma, Louis Blankemeier, Christian Bluethgen, Arne Edward Michalson Md, Michael Moseley, Curtis Langlotz, Akshay S Chaudhari, and Jean-Benoit Delbrouck. 2024. [GREEN: Generative Radiology Report Evaluation and Error Notation](#). In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 374–390, Miami, Florida, USA. Association for Computational Linguistics.
- Andrew Sellergren, Sahar Kazemzadeh, Tiam Jaroensri, Atilla Kiraly, Madeleine Traverse, Timo Kohlberger, Shawn Xu, Fayaz Jamil, Cían Hughes, Charles Lau, Justin Chen, Fereshteh Mahvar, Liron Yatziv, Tiffany Chen, Bram Sterling, Stefanie Anna Baby, Susanna Maria Baby, Jeremy Lai, Samuel Schmidgall, and 62 others. 2026. [Medgemma technical report](#). *Preprint*, arXiv:2507.05201.
- Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language Agents with Verbal Reinforcement Learning. In *Advances in Neural Information Processing Systems (NeurIPS)*.
- Akshay Smit, Saahil Jain, Pranav Rajpurkar, Anuj Pareek, Andrew Ng, and Matthew Lungren. 2020. [Combining automatic labelers and expert annotations for accurate radiology report labeling using BERT](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1500–1519. ArXiv:2004.09167.
- Dídac Surís, Sachit Menon, and Carl Vondrick. 2023. [ViperGPT: Visual Inference via Python Execution for Reasoning](#). In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 11854–11864, Paris, France. IEEE.
- Xingyao Wang, Yangyi Chen, Lifan Yuan, Yizhe Zhang, Yunzhu Li, Hao Peng, and Heng Ji. 2024. [Executable code actions elicit better LLM agents](#). In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 50208–50232. PMLR.
- Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. [ReAct: Synergizing Reasoning and Acting in Language Models](#). *arXiv preprint*. ArXiv:2210.03629 [cs.CL].
- Alex L. Zhang, Tim Kraska, and Omar Khattab. 2026. [Recursive Language Models](#). *arXiv preprint*. ArXiv:2512.24601 [cs.AI].
- Xiaoman Zhang, Hong-Yu Zhou, Xiaoli Yang, Oishi Banerjee, Julián N. Acosta, Josh Miller, Owen Huang, and Pranav Rajpurkar. 2024. [ReXrank: A Public Leaderboard for AI-Powered Radiology Report Generation](#). *arXiv preprint*. ArXiv:2411.15122 [cs.CV].
- Yiliang Zhou, Hanley Ong, Patrick Kennedy, Carol C. Wu, Jacob Kazam, Keith Hentel, Adam Flanders, George Shih, and Yifan Peng. 2024. [Evaluating GPT-4V \(GPT-4 with Vision\) on Detection of Radiologic Findings on Chest Radiographs](#). *Radiology*, 311(2):e233270.

A Bootstrap Confidence Intervals for GREEN Scores

Figure 2 shows 95% bootstrap confidence intervals (2,000 resamples over per-patient GREEN scores) for all 32 experimental cells, organised by pipeline. Filled dots show failure-adjusted means (GREEN = 0 assigned to incomplete runs); open circles show means computed over completed runs only. The figure supports the two reliability claims in §5: (1) the $P_1 > P_2$ gap is clear and non-overlapping at non-zero memory conditions, while intervals overlap at memory=none; (2) P_3 GREEN scores remain near zero regardless of model or memory condition, and failure adjustment changes them only marginally.

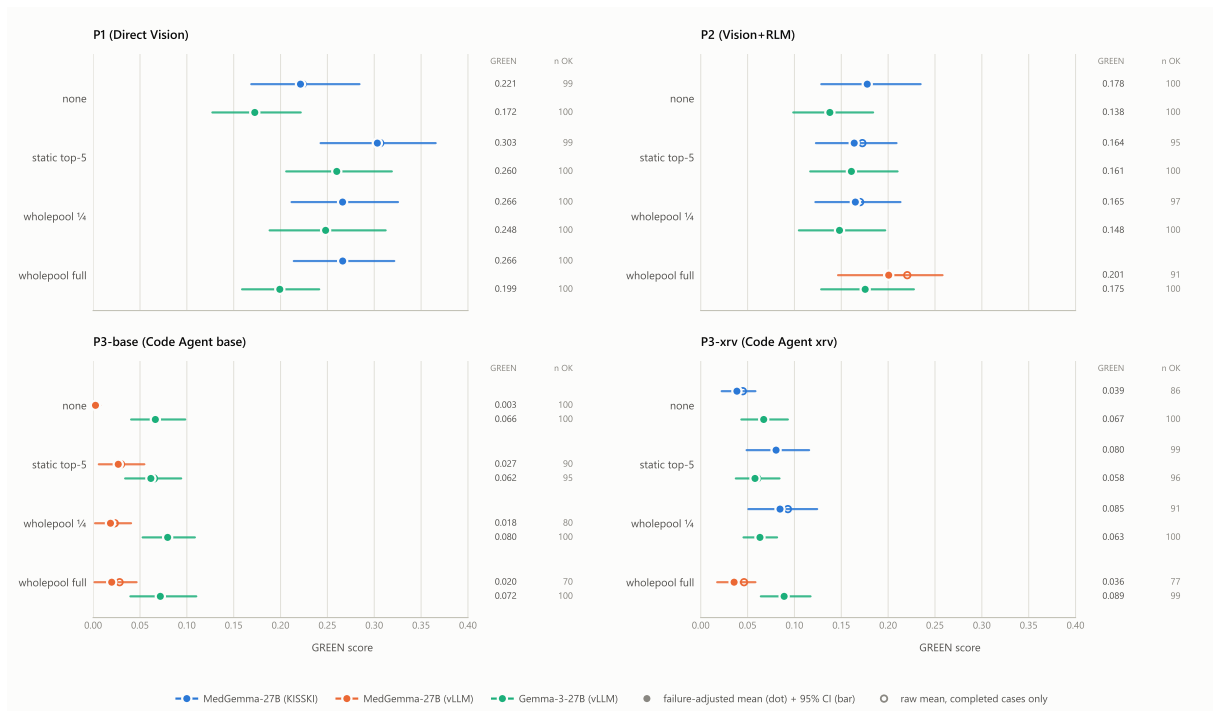


Figure 2: 95% bootstrap confidence intervals (2,000 resamples) for GREEN scores across all 32 experimental cells. Each panel corresponds to one pipeline; rows within a panel are memory conditions. **Filled dot**: failure-adjusted mean (failed runs scored as GREEN = 0). **Open circle**: mean over completed runs only. **Error bars**: 95% CI. Blue = MedGemma-27B via KISSKI (API); orange = MedGemma-27B local vLLM (INT4); green = Gemma-3-27B local vLLM. n_{ok} gives the number of successfully completed runs out of 100.

B Prompt Templates

All prompts are given verbatim, byte-identical to the source files. The placeholder `{retrieved_cases}` is filled at runtime (“Case 1: ...” ... “Case N: ...”, or “None provided.” for none). The output-format block is shared across all pipelines and enforced via a SHA-256 fingerprint assertion at run time.

P₁: Direct Vision

Sent as a single multimodal user message; the query image is prepended as a base64 PNG content block.

Listing 1: P₁ user message

```
[IMAGE: base64 PNG of query chest X-ray]

You are an expert radiologist interpreting a single frontal chest X-ray.

Similar prior cases (from other patients, for reference only):
{retrieved_cases}

Task: write a radiology report for the X-ray shown.
- Use two sections, exactly: "Findings:" then "Impression:".
- In the Findings section, list each finding on a separate numbered line (e.g. "1. Cardiomegaly."). One finding per line. State each finding concisely and specifically.
- Include both positive findings and important negatives (e.g. pleural effusion, pneumothorax, consolidation, edema, cardiomegaly, atelectasis, support devices).
- Describe only what is visible in THIS X-ray. Do not copy the prior cases.
- Output only the report. No preamble, no closing remark, no markdown.
```

P₂: Vision + RLM

Listing 2: P₂ system prompt (sent once at loop start)

```
You are a clinical radiology AI assistant interpreting ONE frontal chest X-ray (the query image) inside an RLM Python REPL. You CANNOT see the image directly. You have ONE tool, analyze_image(), which sends the query X-ray to a vision model and returns its imaging findings as text
-- this is your way to 'see' the image.

Write ALL Python inside ```repl code blocks -- these are the ONLY blocks that execute. Put the code DIRECTLY inside the ```repl fence
-- do NOT write a ```python fence and do NOT nest any code fence inside
```repl (only ```repl runs; a nested fence is a syntax error). Use print() to see output between turns.

Work in two phases:
PHASE 1 -- EXAMINE: in a ```repl block, call findings = analyze_image() and print(findings) to obtain the query X-ray's imaging findings.
PHASE 2 -- REASON & REPORT: reason as a radiologist over those vision findings, together with the similar prior cases below, and write the
```

```
radiology report.

Your final report is YOUR clinical reasoning -- NOT a restatement of the analyze_image() text. Interpret what the vision findings and the prior cases imply about THIS patient. The vision findings describe THIS X-ray and are your PRIMARY evidence; use the prior cases only as supporting context -- do NOT assert a finding the vision read does not support, and do not copy the prior cases.

Submit from inside a ```repl block by setting answer["content"] to the report, then answer["ready"] = True. Do NOT set answer["ready"] = True until you have actually called analyze_image().

The report MUST use exactly two sections, "Findings:" then "Impression:". In the Findings section, list each finding on a separate numbered line (e.g. "1. Cardiomegaly."), one finding per line, each stated concisely and specifically. Include positive findings and the important negatives (e.g. pleural effusion, pneumothorax, consolidation, edema, cardiomegaly, atelectasis, support devices), in plain standard radiology language, present tense. Describe only what is visible in THIS X-ray; do not copy the prior cases. Output only the report: no preamble, no closing remark, no markdown. This is ONE query and ONE report -- do NOT rank, compare, or describe multiple patients.

Similar prior cases (from other patients, for reference only -- do not copy them):
{retrieved_cases}
```

Listing 3: P<sub>2</sub> kickoff message (first turn); the REPORT TASK block is the same LOCKED\_PROMPT as P<sub>1</sub>

```
You have access to analyze_image() to examine the query chest X-ray. Call it from a ```repl block, read the returned findings, then write the radiology report into answer["content"] and set answer["ready"] = True.

Example:
```repl
findings = analyze_image()
print(findings)
```

===== REPORT TASK =====
You are an expert radiologist interpreting a single frontal chest X-ray.

Similar prior cases (from other patients, for reference only):
{retrieved_cases}

Task: write a radiology report for the X-ray shown.
- Use two sections, exactly: "Findings:" then "Impression:".
- In the Findings section, list each finding on a separate numbered line (e.g. "1. Cardiomegaly."). One finding per line.
- Include both positive findings and important negatives.
- Describe only what is visible in THIS X-ray. Do not copy prior cases.
- Output only the report. No preamble, no closing remark, no markdown.
```

Listing 4: analyze\_image() vision tool prompt (sent with the query image to the same model)

```
You are a radiologist. Describe the radiographic findings visible in
```

this single frontal chest X-ray. Report the relevant positive findings and the pertinent negatives (for example pleural effusion, pneumothorax, consolidation, edema, cardiomegaly, atelectasis, and support devices). Be specific and factual and describe only what is visible in this image.

### P<sub>3</sub>: Code Agent

Both variants share the same structure and kickoff; only the EXAMPLE block differs.

#### Listing 5: P<sub>3</sub>-base system prompt (numpy/PIL example)

You are a clinical radiology AI interpreting ONE frontal chest X-ray (the query image) inside an RLM Python REPL. You cannot passively "look" at the image -- but your CODE is your eyes: by writing Python that opens and analyses the pixel data you can extract real, quantitative evidence about THIS X-ray. There is no vision tool; every observation must come from code you write and run yourself.

A variable named QUERY\_IMAGE\_PATH is ALREADY defined in the REPL and holds the absolute path to the query image PNG. Use it directly; do NOT reassign it or invent a filename -- that file does not exist. A full scientific-Python stack is installed: numpy, scipy, PIL, opencv, cv2, scikit-image (skimage), matplotlib, monai, torch, and torchxrayvision (pretrained chest-X-ray classifiers).

EXAMPLE -- ONE simple approach to get you started. You decide whether to use it, extend it, or analyse the image a different way (for instance with a pretrained classifier). Put code like this inside a ```repl block:

```
import numpy as np
from PIL import Image
img = np.asarray(
 Image.open(QUERY_IMAGE_PATH).convert('L'),
 dtype=np.float32) / 255.0
h, w = img.shape
print('size', img.shape,
 'mean_intensity', round(float(img.mean()), 3))
print('left_mean', round(float(img[:, :w//2].mean()),
3))
print('right_mean', round(float(img[:, w//2:].mean()),
3))
print('upper_mean', round(float(img[:h//2, :].mean()),
3))
print('lower_mean', round(float(img[h//2:, :].mean()),
3))
print('opacity_fraction',
 round(float((img > 0.55).mean()), 3))
```

These are generic pixel statistics (brightness, left/right and upper/lower asymmetry, opacity burden) -- NOT a diagnosis. It is up to YOU to compute whatever evidence you need and to interpret it as radiological findings.

Write ALL Python inside ```repl code blocks -- these are the ONLY blocks that execute. Put the code DIRECTLY inside the ```repl fence -- do NOT write a ```python fence and do NOT nest any code fence inside ```repl (only ```repl runs; a nested fence is a syntax error). Use print() to see output between turns.

Work in two phases:  
PHASE 1 -- ANALYSE: write and run code to extract real

evidence from the image; print what you find.  
PHASE 2 -- REASON & REPORT: reason as a radiologist over the evidence  
YOUR code produced, together with the similar prior cases below, and write the radiology report.

Rules:

- Base the report on evidence your code actually computed from THIS image. Do NOT submit a disclaimer such as 'the image cannot be assessed' -- running your analysis IS your inspection.
- Do NOT set answer["ready"] = True until your code has produced concrete image-derived evidence (probabilities, measurements, or statistics -- not merely the image dimensions).

Submit from inside a ```repl block by setting answer["content"] to the report, then answer["ready"] = True.

The report MUST use exactly two sections, "Findings:" then "Impression:". In the Findings section, list each finding on a separate numbered line (e.g. "1. Cardiomegaly."), one finding per line, each stated concisely and specifically. Include positive findings and the important negatives, in plain standard radiology language, present tense. Output only the report: no preamble, no closing remark, no markdown. This is ONE query and ONE report.

Similar prior cases (from other patients, for reference only -- do not copy them):  
{retrieved\_cases}

#### Listing 6: P<sub>3</sub>-xrv EXAMPLE block (replaces the numpy block above; rest of system prompt identical)

EXAMPLE -- ONE possible approach. You decide whether to use it, adapt it, or analyse the image a different way. Write every line at the TOP LEVEL of the ```repl block (NO leading indentation); variables persist across turns. For example:

```
import torch, torchxrayvision as xrv, numpy as np
from PIL import Image
img = np.asarray(
 Image.open(QUERY_IMAGE_PATH).convert('L'),
 dtype=np.float32)
img = xrv.datasets.normalize(img, 255)[None, ...]
img = xrv.datasets.XRayResizer(224)(
 xrv.datasets.XRayCenterCrop()(img))
x = torch.from_numpy(img).unsqueeze(0).float()
model = xrv.models.DenseNet(
 weights='densenet121-res224-all').eval()
probs = model(x)[0].detach().numpy()
print([(n, round(float(v), 3))
 for n, v in zip(model.pathologies, probs)])
```

This prints a probability per pathology (Atelectasis, Effusion, Cardiomegaly, ...). You could also measure intensity, asymmetry, or opacity with numpy/scipy/skimage, or combine methods.

#### Listing 7: P<sub>3</sub> kickoff message (identical for both variants)

Begin. In a ```repl block, analyse the chest X-ray at QUERY\_IMAGE\_PATH with your own code; then reason over your results together with the prior cases in your instructions and write the radiology report into answer["content"], then set answer["ready"] = True.

## C Local vLLM Serving Configuration

All locally-served models used vLLM (Kwon et al., 2023) v0.8.4 on NVIDIA A40 (48 GB) GPUs. The two 27B generation models required bitsandbytes INT4 quantization to fit a single 48 GB GPU; the 7B GREEN judge required no quantization. Each model was pinned to its own GPU and port via CUDA\_VISIBLE\_DEVICES:

```
MedGemma-27B (port 8000, GPU 0)
CUDA_VISIBLE_DEVICES=0 vllm serve \
 google/medgemma-27b-it \
 --served-model-name medgemma-27b-it-vllm \
 --quantization bitsandbytes \
 --load-format bitsandbytes \
 --max-model-len 131072 \
 --gpu-memory-utilization 0.92 \
 --port 8000 --enforce-eager

Gemma-3-27B (port 8001, GPU 1)
CUDA_VISIBLE_DEVICES=1 vllm serve \
 google/gemma-3-27b-it \
 --served-model-name gemma-3-27b-it-vllm \
 --quantization bitsandbytes \
 --load-format bitsandbytes \
 --max-model-len 131072 \
 --gpu-memory-utilization 0.92 \
 --port 8001 --enforce-eager

GREEN judge (port 8003, GPU 3)
CUDA_VISIBLE_DEVICES=3 vllm serve \
 StanfordAIMI/GREEN-RadLlama2-7b \
 --served-model-name green-radllama2-7b \
 --port 8003
```

--enforce-eager disables CUDA graph capture, which would otherwise OOM during warm-up due to the additional transient memory required on top of bitsandbytes dequantization buffers (Kwon et al., 2023). INT4 weights occupy  $\approx 14$  GB per 27B model, leaving  $\approx 30$  GB for the KV cache; at --max-model-len 131072 this is sufficient for the wholepool\_full condition’s worst-case prompt length ( $\approx 6$ K memory tokens) but may require --kv-cache-dtype fp8 for very long RLM iteration chains. MedGemma/Gemma 3 are gated repositories; Hugging Face authentication (huggingface-cli login) and licence acceptance are required before vllm serve can download the weights.

## D Full Evaluation Results

Tables 7–10 report complete per-memory, per-model evaluation metrics for all four pipelines. *Compl.* = completion rate; *Tokens* = mean total RLM tokens per query; *Iters* = mean RLM iterations (for  $P_1$ , which has no loop). CheXbert-All = 14-label F1; CheXbert-5 = 5-label F1; GREEN evaluated by GREEN-RadLlama2-7b.

Table 7: Full results for **P<sub>1</sub>: Direct Vision** across all memory conditions and models. All MedGemma cells used the KISSKI API endpoint; no local vLLM was required for P<sub>1</sub>.

| Mem.    | Model    | $n_{ok}$ | Compl. | Time (s) | Tokens | Iters | CheXbert-All |       | CheXbert-5 |       | GREEN |
|---------|----------|----------|--------|----------|--------|-------|--------------|-------|------------|-------|-------|
|         |          |          |        |          |        |       | $\mu$ F1     | MF1   | $\mu$ F1   | MF1   |       |
| None    | MedGemma | 99       | 99%    | 18.5     | 634    | -     | 0.433        | 0.302 | 0.446      | 0.414 | 0.221 |
| None    | Gemma-3  | 100      | 100%   | 23.0     | 588    | -     | 0.274        | 0.155 | 0.161      | 0.147 | 0.172 |
| Top-5   | MedGemma | 99       | 99%    | 49.1     | 1,291  | -     | 0.506        | 0.340 | 0.494      | 0.457 | 0.303 |
| Top-5   | Gemma-3  | 100      | 100%   | 22.2     | 1,255  | -     | 0.394        | 0.251 | 0.338      | 0.330 | 0.260 |
| WP-Qtr  | MedGemma | 100      | 100%   | 18.2     | 3,474  | -     | 0.496        | 0.296 | 0.457      | 0.385 | 0.266 |
| WP-Qtr  | Gemma-3  | 100      | 100%   | 22.6     | 3,439  | -     | 0.411        | 0.246 | 0.362      | 0.306 | 0.248 |
| WP-Full | MedGemma | 100      | 100%   | 17.5     | 14,315 | -     | 0.463        | 0.289 | 0.460      | 0.414 | 0.266 |
| WP-Full | Gemma-3  | 100      | 100%   | 27.3     | 14,273 | -     | 0.325        | 0.178 | 0.277      | 0.254 | 0.199 |

Table 8: Full results for **P<sub>2</sub>: Vision + RLM** across all memory conditions and models. † = MedGemma WP-Full cell served via local vLLM.

| Mem.    | Model                 | $n_{ok}$ | Compl. | Time (s) | Tokens | Iters | CheXbert-All |       | CheXbert-5 |       | GREEN |
|---------|-----------------------|----------|--------|----------|--------|-------|--------------|-------|------------|-------|-------|
|         |                       |          |        |          |        |       | $\mu$ F1     | MF1   | $\mu$ F1   | MF1   |       |
| None    | MedGemma              | 100      | 100%   | 53.3     | 5,647  | 2.4   | 0.409        | 0.272 | 0.390      | 0.338 | 0.178 |
| None    | Gemma-3               | 100      | 100%   | 135.1    | 4,472  | 2.0   | 0.298        | 0.156 | 0.129      | 0.129 | 0.138 |
| Top-5   | MedGemma              | 95       | 95%    | 86.0     | 10,365 | 3.8   | 0.441        | 0.299 | 0.406      | 0.343 | 0.172 |
| Top-5   | Gemma-3               | 100      | 100%   | 155.1    | 5,964  | 2.1   | 0.307        | 0.182 | 0.139      | 0.140 | 0.161 |
| WP-Qtr  | MedGemma              | 97       | 97%    | 78.0     | 16,748 | 3.8   | 0.470        | 0.315 | 0.478      | 0.425 | 0.168 |
| WP-Qtr  | Gemma-3               | 100      | 100%   | 173.8    | 11,406 | 2.5   | 0.312        | 0.180 | 0.176      | 0.166 | 0.148 |
| WP-Full | MedGemma <sup>†</sup> | 91       | 91%    | 234.1    | 47,501 | 4.4   | 0.415        | 0.266 | 0.386      | 0.338 | 0.220 |
| WP-Full | Gemma-3               | 100      | 100%   | 179.6    | 31,364 | 2.5   | 0.303        | 0.157 | 0.127      | 0.129 | 0.175 |

Table 9: Full results for **P<sub>3</sub>-base: Code Agent (base)** across all memory conditions and models. ‡ = all MedGemma cells served via local vLLM.

| Mem.    | Model                 | $n_{ok}$ | Compl. | Time (s) | Tokens | Iters | CheXbert-All |       | CheXbert-5 |       | GREEN |
|---------|-----------------------|----------|--------|----------|--------|-------|--------------|-------|------------|-------|-------|
|         |                       |          |        |          |        |       | $\mu$ F1     | MF1   | $\mu$ F1   | MF1   |       |
| None    | MedGemma <sup>‡</sup> | 100      | 100%   | 322.0    | 9,656  | 3.0   | 0.237        | 0.093 | 0.212      | 0.167 | 0.003 |
| None    | Gemma-3               | 100      | 100%   | 237.9    | 17,882 | 6.0   | 0.248        | 0.123 | 0.239      | 0.207 | 0.066 |
| Top-5   | MedGemma <sup>‡</sup> | 90       | 90%    | 361.9    | 10,087 | 2.9   | 0.204        | 0.090 | 0.143      | 0.142 | 0.030 |
| Top-5   | Gemma-3               | 95       | 95%    | 364.1    | 24,861 | 6.6   | 0.260        | 0.150 | 0.208      | 0.187 | 0.065 |
| WP-Qtr  | MedGemma <sup>‡</sup> | 80       | 80%    | 426.1    | 11,715 | 2.6   | 0.235        | 0.138 | 0.211      | 0.189 | 0.023 |
| WP-Qtr  | Gemma-3               | 100      | 100%   | 346.3    | 37,153 | 6.9   | 0.270        | 0.145 | 0.263      | 0.257 | 0.080 |
| WP-Full | MedGemma <sup>‡</sup> | 70       | 70%    | 514.1    | 43,278 | 4.4   | 0.242        | 0.122 | 0.188      | 0.183 | 0.028 |
| WP-Full | Gemma-3               | 100      | 100%   | 434.0    | 98,263 | 7.7   | 0.290        | 0.149 | 0.306      | 0.290 | 0.072 |

Table 10: Full results for **P<sub>3</sub>-xrv: Code Agent (xrv)** across all memory conditions and models. † = MedGemma WP-Full cell served via local vLLM.

| Mem.    | Model                 | $n_{ok}$ | Compl. | Time (s) | Tokens | Iters | CheXbert-All |       | CheXbert-5 |       | GREEN |
|---------|-----------------------|----------|--------|----------|--------|-------|--------------|-------|------------|-------|-------|
|         |                       |          |        |          |        |       | $\mu$ F1     | MF1   | $\mu$ F1   | MF1   |       |
| None    | MedGemma              | 86       | 86%    | 188.6    | 30,871 | 8.7   | 0.303        | 0.267 | 0.397      | 0.382 | 0.045 |
| None    | Gemma-3               | 100      | 100%   | 179.0    | 14,790 | 5.4   | 0.338        | 0.223 | 0.404      | 0.396 | 0.067 |
| Top-5   | MedGemma              | 99       | 99%    | 146.8    | 33,279 | 8.1   | 0.299        | 0.231 | 0.388      | 0.373 | 0.080 |
| Top-5   | Gemma-3               | 96       | 96%    | 250.6    | 24,469 | 6.8   | 0.329        | 0.216 | 0.411      | 0.404 | 0.060 |
| WP-Qtr  | MedGemma              | 91       | 91%    | 243.7    | 53,225 | 8.4   | 0.236        | 0.155 | 0.332      | 0.313 | 0.090 |
| WP-Qtr  | Gemma-3               | 100      | 100%   | 251.9    | 31,427 | 6.3   | 0.326        | 0.207 | 0.381      | 0.379 | 0.063 |
| WP-Full | MedGemma <sup>†</sup> | 77       | 77%    | 647.5    | 97,383 | 8.7   | 0.230        | 0.166 | 0.301      | 0.290 | 0.044 |
| WP-Full | Gemma-3               | 99       | 99%    | 350.7    | 96,323 | 8.2   | 0.324        | 0.203 | 0.397      | 0.390 | 0.090 |