CYBERNEURO: A PRIVACY-PRESERVING AGENTIC WORKBENCH FOR COHORT-SCALE BRAIN IMAGING AND CLINICAL DATA ANALYSIS

Anonymous authors
Paper under double-blind review

ABSTRACT

Despite tremendous success in neuroimaging methodology, making large-scale, high-dimensional datasets ready for AI/ML applications remains a critical operational bottleneck. Conventional workflows require extensive manual effort across metadata curation, pipeline execution, post-processing quality control, and data management, a burden that disproportionately excludes laboratories with limited manpower and computational infrastructure. To address this real-world barrier, there is an urgent need for scalable, cost-effective computational platforms that democratize advanced neuroimaging analytics and accelerate discoveries in mental health and clinical translation. Capitalizing on multi-agent LLM breakthroughs, we introduce *CyberNeuro*, an agentic workbench with a tailored local LLM-model (called 'WandaMind') for automated neuroimaging and health-data analysis. Driven by four dedicated agents (Planner, Validator, Dispatcher, and Reporter) communicating via a secure MCP bridge and a pinned execution layer, *CyberNeuro* enables researchers to execute complex workflows using natural language; in the evaluated local configuration, all workflows completed with process-level network access denied. On the held-out QC-62 suite, WandaMind raises overall accuracy from 40% to 73% over its base model, mainly through safer abstention and fewer inappropriate tool calls; tool-selection accuracy is unchanged and argument validity drops slightly. Beyond automated metrics, the platform integrates a human-in-the-loop verification panel, which currently supports quality control of raw T1w images. On an end-to-end DICOM-to-BIDS suite of ten cohorts, *CyberNeuro* completed all tasks using about 10.6% of NeuroClaw's per-volume token count with WandaMind and 61.7% with a cloud backend. The QC-62 benchmark and implementation details are provided in the appendix.

1 INTRODUCTION

More and more neuroimaging studies jointly analyze structural MRI, functional MRI, diffusion MRI, clinical measurements, phenotypic variables, and behavioral data. Preparing these heterogeneous data for statistical analysis or machine learning requires multiple specialized tools, including FreeSurfer (Fischl, 2012), ANTs (Avants et al., 2011), FSL (Jenkinson et al., 2012), CIVET (Ad-Dab'bagh et al., 2006), fMRIPrep (Esteban et al., 2019), and QSIPrep (Cieslak et al., 2021). Although these tools are individually well-developed, they differ in header file conventions, runtime dependencies, coordinate systems, output formats, and interaction models. Researchers therefore spend substantial time configuring environments, moving files, verifying outputs, recovering from partial failures, and switching between modality-specific viewers. These operational costs grow with cohort size and can become a great bottleneck (Poldrack & Gorgolewski, 2019; Renton et al., 2024).

Recently, agentic assistants have begun to reduce this burden by translating natural-language requests into executable workflows. Mounting agent research has shown that language models can interleave reasoning with actions, learn API use, and decompose requests into tool graphs (Yao et al., 2023; Schick et al., 2023; Shen et al., 2024). NeuroClaw (Wang et al., 2026), for example, provides dataset-aware orchestration, a broad skill library, runtime support, and the NeuroBench evaluation suite. These systems demonstrate the practical promise of agentic neuroimaging. However, several system-level questions remain insufficiently evaluated: whether a small local model can

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

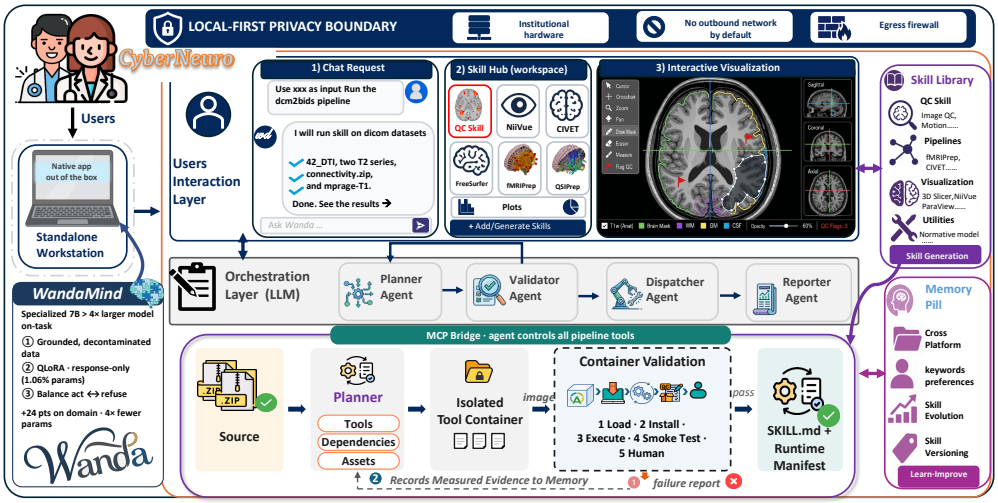

Figure 1: **System architecture of *CyberNeuro*.** The platform operates within a strict **local-first privacy boundary**; all evaluated workflows complete with process-level network access denied. Key components include: an **Interaction Layer** for chat and visualization; an **Orchestration Layer** driven by the **WandaMind** 7B model and four agents via an MCP bridge; a **Skill Evolution** pipeline for rigorous tool validation; and a **Memory Pill** for personalized workflow retention.

support useful end-to-end execution; whether invalid plans can be detected before consuming substantial compute; whether reported success corresponds to a valid scientific derivative; and whether execution, visual inspection, correction, and replay can be completed without repeatedly changing applications.

CyberNeuro addresses these questions through a local-first, end-to-end workbench (as shown in Figure 1). A four-stage orchestration pipeline converts a natural-language request into a validated execution graph, dispatches registered skills in pinned Windows/WSL environments, verifies the resulting artifacts, and presents the outputs through an integrated visualization and quality-control interface. Unlike NeuroClaw, which semi-automatically runs user-selected pipelines with only partial visual QC (Table D.1), a *CyberNeuro* request states the goal, not the tool; the workbench picks the skills and their environments and shows the results in its own viewers. *CyberNeuro* runs locally on the same workstation as the application; users who want a cloud model can connect one through their own API endpoint. Every run produces a structured receipt, linked to its request, that records the output artifacts, validation results, input lineage, and completion status.

The contributions of this work involve four-fold: **(1) An end-to-end neuroimaging workbench.** *CyberNeuro* integrates natural-language interaction, Windows/WSL runtime configuration, 67 processing and visualization skills, artifact inspection, interactive quality control, and reproducible reporting within a single desktop application. **(2) A validated execution architecture.** *CyberNeuro* separates LLM-based planning from deterministic contract, permission, runtime, and artifact validation. A task is reported as successful only when the expected scientific derivative is produced and passes the corresponding output checks. **(3) A resource-efficient local LLM.** We fine-tune a LLM, called WandaMind, as the brain of *CyberNeuro*, which adapts a 7B open-weight model to act-or-abstain decisions in neuroimaging tool use and runs locally on commodity hardware. On a held-out domain benchmark, its largest gain is in refusing incorrect tool use: abstention recall rises from 0% to 100% and false actions drop from 100% to 0%. **(4) A system-level evaluation.** We compare *CyberNeuro* with NeuroClaw and conventional multi-tool workflows using end-to-end execution success, artifact validity, manual intervention, computational overhead, user-facing time,

reproducibility, and network egress; in the evaluated local configuration, all workflows completed with process-level network access denied.

2 SYSTEM DESIGN IN *CyberNeuro*

CyberNeuro is organized as a local-first Windows execution boundary in which an LLM proposes actions, deterministic components validate and dispatch registered skills, and the workbench presents verified outputs for human inspection. Table D.1 summarizes the system-level capabilities of *CyberNeuro* and representative related systems.

2.1 LOCAL DEPLOYMENT BOUNDARY

CyberNeuro is distributed as a local-first desktop application. Core services, including the WandaMind runtime and orchestration layer, execute on the host workstation, while Linux-only neuroimaging packages run within a bundled WSL environment. Following initial provisioning, all workflows can execute entirely offline, without cloud dependencies, internet access, or external API keys. Any network access, such as connecting a user-supplied cloud model, requires explicit user consent; this consent gate is the application-level egress firewall shown in Figure 1. As detailed in the Tool-Access Tier subsection, this zero-egress privacy boundary is verified via network traffic capture.

2.2 THREAT MODEL AND WORKSPACE ISOLATION

CyberNeuro mitigates risks of unauthorized data transmission and file modification during agentic analysis. Assuming trusted host environments, we address three core threats: out-of-policy tool calls, unsafe path arguments, and unauthorized outbound connections. Defenses include capability allowlists, path canonicalization, and strict workspace confinement, adhering to least privilege principles (Saltzer & Schroeder, 1975) and mitigating indirect prompt injections (Greshake et al., 2023). The agent runtime starts from a deny-all permission policy: reading and searching are allowed within the selected workspace and the registered Skill roots (excluding .env files), web fetch and web search are denied, and file edits and shell commands require approval, either per action in Ask mode, or under an explicit session-level Auto mode. Bundled Skills ship as application resources, and uploaded Skills are kept as separate editable copies that are never promoted to package executors. Each run writes to a new timestamped folder under the chosen output location.

2.3 FOUR-STAGE HYBRID ORCHESTRATION

The orchestration layer implements planning, validation, dispatch, and reporting as four mutually decoupled agents rather than a single unconstrained conversational agent. This separation prevents an LLM-generated statement of success from being treated as evidence of actual scientific computation.

Planner. The Wanda Core, an agentic harness specially designed for neuroimaging tasks, interprets the request and retrieves candidate skills from their `SKILL.md` descriptions and typed contracts. It produces a task directed acyclic graph (DAG) whose nodes specify a skill, resolved inputs, parameters, expected outputs, runtime requirements, permissions, and recovery alternatives. The Planner asks for clarification when a required modality, dataset, output, or policy decision cannot be inferred safely. A first-pass critique removes internally inconsistent or under-specified plans before deterministic validation.

Validator. Validation occurs at three check points. Contract validation checks skill identity, parameter schemas, input types, and output destinations. Policy and runtime validation checks canonical paths, permissions, overwrite risks, network requirements, and availability of the required Windows or WSL runtime. Post-execution validation checks exit status, expected files, file parseability, domain-specific invariants, and the absence of configuration-only or demo fallbacks. Missing inputs or runtimes yield an `incomplete` or `failed` receipt rather than a successful response. A zero exit code without verified, content-hashed outputs fails result validation (`needs_review`) and is recorded as `incomplete` rather than `completed`.

Dispatcher. The Dispatcher submits validated nodes through the Tool Registry, enforces the runtime and permission decisions attached to each node, and records progress, logs, resource use, and failures. Lightweight numerical skills may run natively; heavyweight tools execute in pinned Linux environments. A new tool becomes executable only after it is registered as a validated skill.

Reporter. The Reporter assembles validated derivatives, HTML summaries, logs, and receipts, and exposes compatible outputs to the integrated visualization and QC panel, as shown in the Co-visualization Panel section. Edits performed in the panel become new provenance nodes rather than silently modifying the original result. Downstream stages can then be replayed from the corrected artifact.

Two restricted recovery components support this lifecycle. The Environment Doctor maps recognized runtime-error signatures to validated installation or configuration actions. The Pipeline Doctor combines deterministic log parsing with a bounded set of recovery strategies, conceptually related to language-agent feedback and repair (Shinn et al., 2023) but constrained here by registered tool contracts. Neither component may introduce an unregistered command: every revised action returns to the Validator before execution.

CyberNeuro is therefore *LLM-routed but tool-deterministic*. The language model proposes how a request should be routed, deterministic tools perform the scientific computation, and the validation layer determines whether a reported result corresponds to a real, policy-compliant artifact.

2.4 Tool-Access Tier: the MCP Bridge

The open Model Context Protocol (MCP) (Anthropic, 2024) provides the communication interface between orchestration and registered capabilities, then *CyberNeuro*'s policy layer determines which capabilities are exposed. These include skill invocation, constrained file operations, metadata and database queries, DICOM inspection, runtime diagnostics, and visualization. Arguments are validated before dispatch and every invocation is audit-logged. The LLM cannot open arbitrary host files directly. If a multimodal model is connected for visual inspection, only the selected image or derived preview is supplied to it. This boundary is important because tool-connected agents can convert untrusted content into consequential actions if instructions and data are not separated (Greshake et al., 2023). MCP standardizes the interface, while workspace, permission, and validation policies provide the enforcement.

2.5 Execution Tier

Skills are invoked as subprocesses in versioned, reproducible environments on one workstation. Heavyweight neuroimaging tools such as FreeSurfer, fMRIPrep, and QSIPrep execute inside the bundled WSL environment, using Apptainer images where supported, since immutable environments and declared data dependencies improve reproducibility across repeated executions (Kurtzer et al., 2017; Di Tommaso et al., 2017; Köster & Rahmann, 2012). Lightweight skills run against the embedded Windows Python environment or as native binaries, and the launcher resolves the declared tool version and contract across the Windows host and its local WSL backend. Each skill declares a version-pinned runtime profile, selected from the declaration rather than by the model and provisioned once for all skills that share it (48 skills, 39 profiles; Appendix C.3).

Each invocation writes a **reproducibility receipt** (`RunReceipt`), a JSON record linked to its `RunRequest`, which pins the exact selected inputs by path, size, modification time, and SHA-256 and is re-verified before execution. The receipt stores the tool, start and finish times, status evidence, output artifacts, result validation (SHA-256-verified consumed inputs and outputs, deliverable coverage, and open issues), input lineage, and the next decision returned to the agent, following research-object packaging and provenance practice (Soiland-Reyes et al., 2022; Sandve et al., 2013). Its status is `completed`, `incomplete`, or `failed`; tool-reported states such as `blocked` or `timeout` are recorded as `failed`.

2.6 Skill Library

General domain skills. In the *CyberNeuro* framework, a **domain-skill** serves as a standardized wrapper that translates a specific neuroimaging tool into an LLM-executable action. The current

release contains 67 such processing and visualization skills. As categorized in Table B.1, these span a wide spectrum of tasks, including structural, functional, and diffusion MRI processing, deep learning applications, and interactive visualization. Each skill is deployed as a versioned package comprising a `SKILL.md` instruction file, typed input/output contracts, runtime specifications (Windows/WSL), executable wrappers, and artifact validators. To ensure robust automated orchestration, a skill is designated as "validated" only after passing rigorous schema, installation, real-data execution, expected-failure, and replay tests within its declared environment.

Self-evolving Skill Generation. *CyberNeuro* can also generate new skills. An LLM converts source material into an environment-independent procedure graph of steps and typed capability requests, and a deterministic gate rejects malformed graphs and any machine-specific detail from the model. A deterministic binder resolves each request with predefined probes of the target environment into a verified binding or an explicit abstention; missing tools are added only via human-reviewed container recipes. The rendered SKILL.md takes concrete details only from verified bindings and is re-probed (Appendix C.1, Figure C.1).

2.7 WandaMind – Local LLM Model

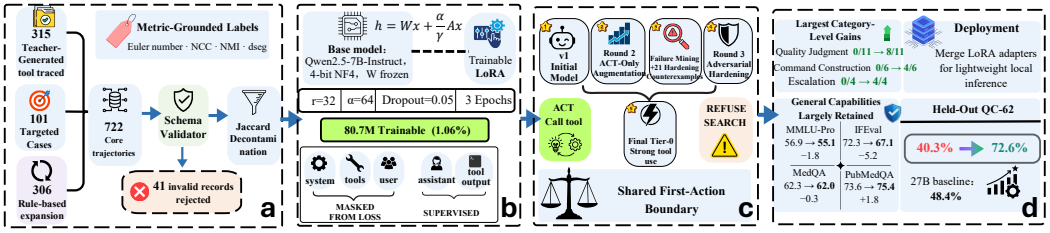

Figure 2: **The fine-tuning pipeline and evaluation workflow.** Overview of the iterative LoRA adapter training for tool-use optimization. **a:** Training data construction and validation. **b:** Response-only QLoRA configuration. **c:** Iterative training and decision hardening. **d:** Tool-use and general capability evaluation.

WandaMind is a domain-adapted 7B model designed to orchestrate neuroimaging tools under the resource constraints of local deployment. Its purpose is not to replace larger general-purpose models in open-ended reasoning. Instead, WandaMind is adapted to decide whether to act, abstain, or request additional assistance within the bounded *CyberNeuro* environment; its largest gain over the base model is in refusing incorrect tool use (Section 3.1). Unless otherwise stated, WandaMind refers to the Qwen2.5-7B-Instruct checkpoint (Team et al., 2025) adapted using QLoRA (Dettmers et al., 2023). The QC-62 bench used for evaluation can be found in the appendix.

Training data construction and filtering. As shown in Fig. 2a, our WandaMind training corpus combines three complementary sources. *First*, 315 Qwen3.6-27B teacher trajectories demonstrate successful tool selection and argument construction. *Second*, 306 rule-based expansions increase coverage of known tool and parameter combinations without requiring additional teacher inference. *Third*, 101 targeted examples represent difficult neuroimaging operations and quality-control decisions that were underrepresented in the general trajectory pool. After filtering, these sources total 722 validated trajectories, used across the three training rounds below.

Quality-control targets are grounded in operational neuroimaging measurements, including the Euler number, normalized cross-correlation (NCC), normalized mutual information (NMI), and tissue-segmentation metrics derived from `dseg` outputs. The resulting examples teach the model to associate quantitative evidence with an appropriate tool call, escalation decision, or abstention response.

All candidate records pass through a static schema validator that verifies skill names, tool availability, required arguments, data types, and enumerated parameter values. During corpus construction, 41 additional candidates were rejected because they referenced nonexistent skills or contained schema-invalid arguments. To check evaluation leakage, we compare the token set of each candidate with every QC-62 task; the highest pairwise Jaccard similarity is 0.55. Template families are assigned to a single data split to further reduce near-duplicate contamination.

QLoRA configuration and response-Only supervision. As shown in Fig. 2b, we adapt Qwen2.5-7B-Instruct using QLoRA over 4-bit NF4 base weights (Dettmers et al., 2023). The base-model weights remain frozen, while trainable low-rank adapters are applied to the q, k, v, and o attention projections and the gate, up, and down feed-forward projections. The adapter uses rank $r = 32$, scaling parameter $\alpha = 64$, scaling ratio $\alpha/r = 2.0$, and dropout 0.05. This configuration introduces 80.7M trainable parameters, corresponding to 1.06% of the complete model. Training is implemented with Unsloth and uses a response-only objective. System instructions, tool descriptions, and user messages provide context but are masked from the loss, which is computed only over assistant responses and serialized tool calls.

Iterative failure mining and decision hardening. As shown in Fig. 2c, our training proceeds in three sequential rounds. The first round establishes an initial tool-use model using the validated core trajectories. The second round introduces act-oriented examples to enhance tool selection and command construction. Although executable command generation improved, the model became overly eager to act, calling tools where it should have refused, searched, or escalated. The third round therefore mines these decision-boundary failures into 21 hardening counterexamples, retrains on them, and closes with an acceptance check of the act-or-abstain boundary.

Local deployment. For deployment, the trained LoRA adapter is merged with the base model, converted to GGUF, and quantized to Q4_K_M, following standard weight-only post-training quantization practice (Frantar et al., 2023; Lin et al., 2024). The results section evaluates both the full-precision merged model and the exact quantized checkpoint shipped with the application, together with the held-out and general-capability measurements summarized in Fig. 2d.

2.8 Memory Pill and Customized Skill-Evolution

CyberNeuro includes an optional local memory component that stores structured records of previous requests, validated receipts, user corrections, and visualization preferences within the selected workspace. Memory is disabled in visitor mode and can be inspected, exported, or deleted by the user; the portable .cnmc export format should not be interpreted as encryption unless encryption is explicitly enabled.

When planning a new request, the system may retrieve validated past sessions to recover parameter conventions, preferred viewers, or previously approved destinations. Retrieved records are context, not permission to bypass validation. User corrections are attached to the relevant skill version and queued for maintainer review rather than applied automatically. Every promoted skill carries a version and can be restored to reproduce an earlier analysis; repeated workflows may be proposed as composite skills, but promotion follows the same schema, sandbox, artifact, and human-approval process as skills produced by self-evolving skill generation.

2.9 Co-visualization Panel

Inspection of different modalities commonly requires switching among Freeview, BrainNet Viewer, MRIcroGL, ITK-SNAP, Surfice, Workbench, and other viewers with different file conventions and interaction models. Existing QC systems demonstrate the value of structured visual inspection, integrated rating, and collaborative review (Raamana, 2023; Keshavan et al., 2018; Fernandez-Lozano et al., 2024). As shown in Figure B.1, *CyberNeuro* registers compatible viewers as visualization skills that render in the workbench's right pane or open through a controlled local bridge. A request such as "show the white-matter segmentation overlaid on T1w" or "render the top 5% of connectivity edges on the cortical surface" selects the corresponding viewer and resolves any required format conversion. Linked panels synchronize cursor, region, threshold, or camera state when supported by both viewers. Cohort-wide requests run as one resumable batch of up to 200 validated visualization runs (Appendix C.2).

The integration layer handles declared conversions among NIfTI, GIFTI, CIFTI, FreeSurfer, TRK, and VTK formats, color-map propagation, view-state serialization, and deterministic screenshot export, preserving each viewer's supported interactions rather than claiming uniform support for every native function. Appendix C.2 lists the supported inspection and editing operations. In the current release, quality control covers raw T1w images.

3 EXPERIMENTS AND RESULTS

We evaluated *CyberNeuro* at the model and system levels, separating the capability of WandaMind from the behavior of the complete workbench. Four research questions organize the evaluation:

RQ1 (Domain adaptation): Does domain adaptation improve bounded neuroimaging tool use, structured argument construction, and safe abstention?

RQ2 (DICOM-to-BIDS conversion): Can *CyberNeuro* convert heterogeneous DICOM collections into validator-clean BIDS datasets, and how does its operational behavior compare with NeuroClaw?

RQ3 (Visualization and QC): Can *CyberNeuro* complete representative visualization and quality-control operations while reducing application switching and explicit user interaction?

RQ4 (Privacy and local execution): Can the installed local system complete representative workflows without observable outbound traffic?

3.1 RQ1: DOMAIN ADAPTATION ENHANCES SAFE ABSTENTION RATHER THAN TOOL-USE PROFICIENCY

Setup. We evaluated WandaMind on QC-62, a held-out collection of 62 neuroimaging tool-use and quality-control cases. Four model conditions were compared: the untuned Qwen2.5-7B-Instruct base model, the local Qwen3.6-27B, the full-precision WandaMind adapter, and the deployed WandaMind Q4_K_M checkpoint. All models received identical system instructions, skill descriptions, tool schemas, and prompts.

Table 1: WandaMind performance on held-out QC-62 (%).

Model	Overall	Tool selection	Argument validity	Abstention recall	False action
Base 7B	40.3	83.3	93.8	0.0	100.0
Local 27B	48.4	100.0	31.3	50.0	50.0
WandaMind 7B	**72.6**	83.3	87.5	100.0	0.0
WandaMind Q4_K_M	62.9	66.7	81.3	100.0	0.0

The margin is carried by decision quality, not capability. WandaMind leads the untuned 7B base by 32.3 points overall and the local 27B by 24.2 (Table 1), yet its tool selection is unchanged from the base and its argument validity is 6.3 points *worse*. Most of the margin comes from abstention recall and the elimination of false action: of the 23 items that WandaMind passes and the base model fails, 15 are solved by answering or abstaining without a tool call. The 27B comparator makes the same point in reverse, with perfect tool selection, one-third argument validity, and action on half of what it should refuse; its 8.1-point edge over the base is not significant (p=0.18), whereas WandaMind separates from both (p<0.001 and p=0.003; exact McNemar test on paired per-item outcomes, n=62). Quantization to Q4_K_M costs 9.7 points and retains the abstention behavior.

Category-level decomposition and capability retention. Relative to the untuned checkpoint (Fig. 2d), the largest gains occur in quality judgment (0/11 to 8/11), command construction (0/6 to 4/6), and appropriate escalation (0/4 to 4/4). These categories are small (95% Clopper–Pearson intervals for the adapted model: 0.39–0.94, 0.22–0.96, and 0.40–1.00), and only the quality-judgment gain is significant on its own (exact McNemar p=0.008, versus p=0.125 for the other two), so we treat these category results as descriptive. Most out-of-domain capability is retained: MMLU-Pro moves from 56.9 to 55.1, MedQA from 62.3 to 62.0, and PubMedQA from 73.6 to 75.4, while IFEval regresses more visibly from 72.3 to 67.1. The adaptation is therefore decision hardening within a bounded action space rather than a uniform capability gain, and these model-level figures do not establish workbench-level performance, which the remaining questions address.

3.2 RQ2: SINGLE-REQUEST CONVERSION PRODUCES VALIDATOR-CLEAN DERIVATIVES

Setup. *CyberNeuro* collapses DICOM-to-NIfTI conversion and BIDS formatting into a single request (Figure 3a), natively handling heterogeneous inputs (archives, multi-session data) and triaging series automatically. In contrast, NeuroClaw separates these steps, requiring manual handoffs. We

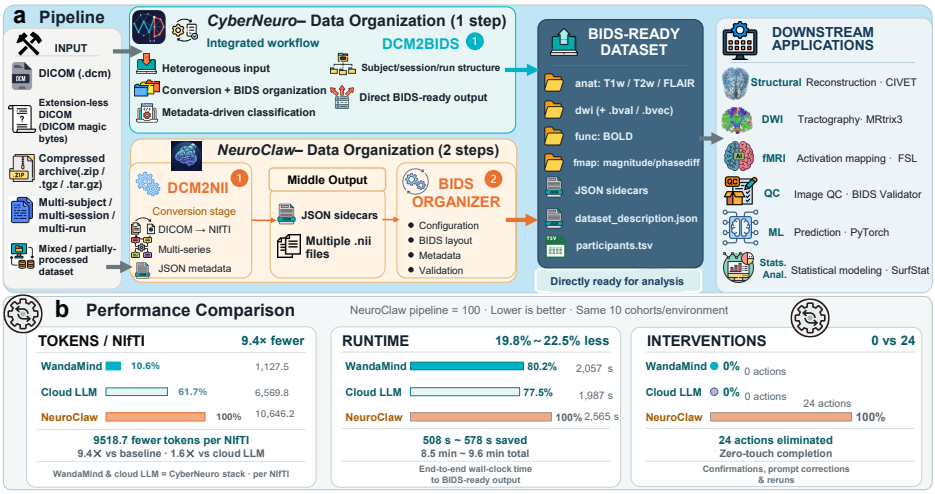

Figure 3: **DICOM conversion and BIDS organization workflow. a:** The integrated *CyberNeuro* pipeline versus the multi-stage NeuroClaw baseline. **b:** End-to-end performance comparison.

evaluated ten heterogeneous cohorts across three configurations: *CyberNeuro* (full-precision WandaMind), *CyberNeuro* (cloud), and NeuroClaw (cloud). Both cloud configurations use DeepSeek V4 Pro as the backend model. In every cohort the task was to organize the anatomical scans into BIDS, so non-anatomical series were out of scope. All *CyberNeuro* runs used the session-level Auto mode (Section 2.2), so no per-action approvals were needed. Detailed results are in Appendix E.

Autonomy is a capability boundary, not a margin. *CyberNeuro* retained 58 of 102 discovered series, correctly excluding 44 (43.1%) non-anatomical scans to produce 61 NIfTI volumes (some series yield more than one volume) with zero post-request interventions. Conversely, NeuroClaw required 24 total manual interventions across all cohorts to diagnose and resume workflows, preventing unattended scheduling.

Token cost separates into two contributions. Swapping the backend moves token consumption $5.8\times$, while the architecture accounts for a further $1.6\times$ efficiency gain. This leaves *CyberNeuro* 89.4% below the baseline per volume with WandaMind. Because NeuroClaw first converts every series to NIfTI and emits 127 out-of-scope volumes, the total token gap is $29.1\times$.

We claim no computational speedup. End-to-end suite time fell by 19.8% with WandaMind and 22.5% with the cloud backend relative to NeuroClaw. This difference reflects the elimination of manual handoffs and intervention stalls, rather than raw computational speedup.

Output quality and auditability. All datasets passed the bids-validator with zero errors. The system properly retained 612 non-blocking warnings in the reports rather than suppressing them, and every run generated a complete `RunReceipt` to ensure full auditability.

Failure injection isolates the decisive control. Disabling artifact validation allowed 4/4 malformed cases to report silent success; restoring it successfully blocked all 4 while falsely rejecting none of 13 bids-validator-clean conversions. Preflight validation changed no final outcomes but drastically cut the median time-to-reject from 112.2 s to 0.9 s. Thus, preflight saves computation, while post-execution validation ensures trustworthy unattended execution.

3.3 RQ3: Run Artifacts Carry Their Own Inspection Evidence

Setup. Three representative tasks (V1 overlay, V2 connectome, V3 QC/replay) were executed $K=3$ times by a human operator under a conventional multi-tool condition and under *CyberNeuro*, with fixed action traces recorded for both.

Table 2: Visualization and QC task results ($K = 3$); times and interaction counts are medians. Conventional times were measured with a human operator; *CyberNeuro* times are machine execution times (Appendix F).

Task	Condition	Success	Time	Applications	Switches	Actions
Overlay	Conventional	3/3	15 min	3	2	14
Overlay	*CyberNeuro*	3/3	0.433 s	1	0	1
Connectome	Conventional	3/3	12 min	3	2	18
Connectome	*CyberNeuro*	3/3	0.368 s	1	0	1
QC/replay	Conventional	3/3	40 min	4	3	20
QC/replay	*CyberNeuro*	3/3	0.860 s	1	0	2

Interaction is removed without cost to completion. All three tasks succeeded in every repetition under both conditions (Table 2). *CyberNeuro* completed each within a single application, cutting application count by 66.7–75.0%, switching by 100%, and actions by 90.0–94.4%. The accompanying time reduction, above 99.94% for all three tasks, measures removed manual work rather than speed: conventional times include learning to run tools that often lack a graphical interface, whereas *CyberNeuro* skills pin their tools and download links, adding at most one environment setup.

Artifacts are self-contained. The RQ2 conversion runs emitted 90 QC images and 14 HTML reports without any separate visualization request. All images are embedded as `data:` URIs, and the reports contain no external URLs, scripts, or stylesheets. The evidence needed to inspect a run can therefore be opened from the run artifact itself with no additional software or network dependency, which does not by itself establish reduced human review time.

3.4 RQ4: The Hardened Release Meets the Observed Local-Operation Boundary

Setup. Research under the governing NIH awards uses institution-controlled offline machines, so we reran four task groups under a monitored condition and under a process-level deny-network policy, with two negative-control connection attempts to confirm the restriction was active.

Zero observed egress under denial. All evaluated workflows (one pipeline and three visualization/QC tasks) completed successfully under both monitored and process-level disconnected conditions. This confirms the system requires no external compute. Under the monitored condition, host-wide `netstat` recorded minor, unattributed background traffic (up to 624 packets or $\sim$1.44 MB). Under process-level denial, exactly zero sent packets and bytes were recorded.

What this does not establish. This shows that the tested release needs no network after installation. However, it does not prove machine-level air-gap compliance, exclude unrelated host-network activity, or demonstrate resistance to compromised dependencies.

4 Discussion and Conclusion

Limitations and future work. The current skill library contains 67 skills, whereas NeuroClaw ships 86. Part of this gap follows from a strictly local-first and auditable design center that raises the cost of adding each skill, but the gap is real. Local models also remain weaker than frontier cloud models at long-context reasoning, which constrains how complex a request *CyberNeuro* can plan in a single pass. Ongoing engineering supported by self-evolving skill generation aims to expand the catalog beyond 100 skills, alongside semi-automated QC of derived masks and registrations, multi-user mode, an opt-in cloud-burst path validated on NIH STRIDES, and a public plug-in registry.

Conclusion. We presented *CyberNeuro*, a local-first agentic workbench for IRB-restricted neuroimaging analysis. Separating LLM-based planning from deterministic validation allowed a 7B local model to drive ten cohort conversions end to end with no user intervention, at roughly one tenth of the baseline per-volume token cost, and every evaluated task completed with zero sent bytes under process-level network denial. This evidence covers the pinned release under the tested configuration and does not by itself establish air-gap compliance. Released under the FAIR4RS principles, *CyberNeuro* is intended as a sustainable community resource for privacy-critical research.

AI USE STATEMENT

In this work, we used generative AI tools for the following tasks that require disclosure. *Synthetic data:* 315 of the 722 records in the WandaMind training pool are tool-use trajectories generated by a Qwen3.6-27B teacher (Table H.2); every record was admitted only after passing the schema, parsing, and post-filter gates. *Implementation:* LLM coding assistants helped implement and debug parts of the *CyberNeuro* software; all code was reviewed by the authors and exercised by the tests and evaluations reported here. *Translation and methodology:* an LLM assistant translated author-written Chinese notes into English and helped select and compute the exact McNemar tests and Clopper–Pearson intervals reported in Section 3.1. We did not use generative AI tools to construct the QC-62 benchmark, whose items were curated by the authors from execution logs; theoretical models, mathematical claims, and proofs are not applicable to this work. Additionally, we used generative AI tools to draft and edit parts of the manuscript and appendix, to convert the paper to the ICLR template, to check the consistency of numbers and cross-references, and to compute derived table entries (e.g., per-NIfTI ratios) from recorded results. We have reviewed all AI-assisted work: derived statistics and table entries were recomputed against the per-item and per-cohort records in the appendix, and all AI-assisted text was checked by the authors against the underlying experiments. Language models are also components of the system under study; those uses are described in the main text. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

REFERENCES

Yasser Ad-Dab'bagh, Dawn Einarson, Oliver Lyttelton, Jens-Sebastian Muehlboeck, Kelvin Mok, Ognen Ivanov, Robert D. Vincent, Claude Lepage, Jason Lerch, Eric Fombonne, and Alan C. Evans. The CIVET image-processing environment: A fully automated comprehensive pipeline for anatomical neuroimaging research. In *Proceedings of the 12th Annual Meeting of the Organization for Human Brain Mapping*, 2006.

Anthropic. Model context protocol specification. `https://modelcontextprotocol.io/specification`, 2024. Accessed 2026.

Brian B. Avants, Nicholas J. Tustison, Gang Song, Philip A. Cook, Arno Klein, and James C. Gee. A reproducible evaluation of ANTs similarity metric performance in brain image registration. *NeuroImage*, 54(3):2033–2044, 2011. doi: 10.1016/j.neuroimage.2010.09.025.

Matthew Cieslak, Philip A. Cook, Xiaosong He, Fang-Cheng Yeh, Thijs Dhollander, Azeez Adebimpe, et al. QSIPrep: An integrative platform for preprocessing and reconstructing diffusion MRI data. *Nature Methods*, 18(7):775–778, 2021. doi: 10.1038/s41592-021-01185-5.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. QLoRA: Efficient finetuning of quantized LLMs. In *Advances in Neural Information Processing Systems*, volume 36, pp. 10088–10115, 2023.

Paolo Di Tommaso, Maria Chatzou, Evan W. Floden, Pablo Prieto Barja, Emilio Palumbo, and Cedric Notredame. Nextflow enables reproducible computational workflows. *Nature Biotechnology*, 35:316–319, 2017. doi: 10.1038/nbt.3820.

Oscar Esteban, Christopher J. Markiewicz, Ross W. Blair, Craig A. Moodie, A. Ilkay Isik, Asier Erramuzpe, et al. fMRIPrep: A robust preprocessing pipeline for functional MRI. *Nature Methods*, 16(1):111–116, 2019. doi: 10.1038/s41592-018-0235-4.

Sofia Fernandez-Lozano, Mahsa Dadar, Cassandra Morrison, et al. QRATER: A collaborative and centralized imaging quality control web-based application. *Aperture Neuro*, 4, 2024. doi: 10.52294/001c.118616.

Bruce Fischl. FreeSurfer. *NeuroImage*, 62(2):774–781, 2012. doi: 10.1016/j.neuroimage.2012.01.021.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. GPTQ: Accurate post-training quantization for generative pre-trained transformers. In *International Conference on Learning Representations*, 2023. URL `https://openreview.net/forum?id=tcbBPnfwxS`.

Kai Greshake, Sahar Abdelnabi, Shailesh Mishra, Christoph Endres, Thorsten Holz, and Mario Fritz. Not what you've signed up for: Compromising real-world LLM-integrated applications with indirect prompt injection. In *Proceedings of the 16th ACM Workshop on Artificial Intelligence and Security*, pp. 79–90, 2023. doi: 10.1145/3605764.3623985.

Mark Jenkinson, Christian F. Beckmann, Timothy E. J. Behrens, Mark W. Woolrich, and Stephen M. Smith. FSL. *NeuroImage*, 62(2):782–790, 2012. doi: 10.1016/j.neuroimage.2011.09.015.

Anisha Keshavan, Esha Datta, Ian M. McDonough, Christopher R. Madan, Katherine Jordan, and Roland G. Henry. Mindcontrol: A web application for brain segmentation quality control. *NeuroImage*, 170:365–372, 2018. doi: 10.1016/j.neuroimage.2017.03.055.

Johannes Köster and Sven Rahmann. Snakemake—a scalable bioinformatics workflow engine. *Bioinformatics*, 28(19):2520–2522, 2012. doi: 10.1093/bioinformatics/bts480.

Gregory M. Kurtzer, Vanessa Sochat, and Michael W. Bauer. Singularity: Scientific containers for mobility of compute. *PLOS ONE*, 12(5):e0177459, 2017. doi: 10.1371/journal.pone.0177459.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. AWQ: Activation-aware weight quantization for LLM compression and acceleration. In *Proceedings of Machine Learning and Systems*, volume 6, pp. 87–100, 2024.

Russell A. Poldrack and Krzysztof J. Gorgolewski. The past, present, and future of the brain imaging data structure (BIDS). *Annual Review of Biomedical Data Science*, 2:119–138, 2019. doi: 10.1146/annurev-biodatasci-072018-021000.

Pradeep Reddy Raamana. VisualQC: Software development kit for medical and neuroimaging quality control and assurance. *Aperture Neuro*, 3, 2023. doi: 10.52294/e130fcd2-ce83-4222-856d-c82022013a50.

Alexander I. Renton et al. Neurodesk: An accessible, flexible and portable data analysis environment for reproducible neuroimaging. *Nature Methods*, 21:804–808, 2024. doi: 10.1038/s41592-023-02145-x.

Jerome H. Saltzer and Michael D. Schroeder. The protection of information in computer systems. *Proceedings of the IEEE*, 63(9):1278–1308, 1975. doi: 10.1109/PROC.1975.9939.

Geir Kjetil Sandve, Anton Nekrutenko, James Taylor, and Eivind Hovig. Ten simple rules for reproducible computational research. *PLOS Computational Biology*, 9(10):e1003285, 2013. doi: 10.1371/journal.pcbi.1003285.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessi, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems*, volume 36, pp. 68539–68551, 2023.

Yiqing Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. TaskBench: Benchmarking large language models for task automation. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, pp. 8634–8652, 2023.

Stian Soiland-Reyes, Peter Sefton, Mercè Crosas, Leyla Jael Castro, Frederik Coppens, José M. Fernández, Daniel Garijo, Björn Grüning, Marco La Rosa, Simone Leo, Eoghan Ó Carragáin, Marc Portier, Ana Trisovic, RO-Crate Community, Paul Groth, and Carole Goble. Packaging Research Artefacts with RO-Crate. *Data Science*, 5(2):97–138, 2022. doi: 10.3233/DS-210053.

Qwen Team et al. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2025. doi: 10.48550/arXiv.2412.15115.

Cheng Wang, Zhibin He, Zhihao Peng, Shengyuan Liu, Yufan Hu, Carl Yang, Lifang He, Lichao Sun, Xiang Li, and Yixuan Yuan. NeuroClaw technical report. *arXiv preprint arXiv:2604.24696*, 2026. doi: 10.48550/arXiv.2604.24696. URL https://arxiv.org/abs/2604.24696.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=WE_vluYUL-X.

APPENDIX

This appendix provides the complete CyberNeuro skill inventory, the released QC-62 benchmark, and supporting reproducibility tables. It includes all 62 benchmark cases and labels, the scoring and decontamination protocol, the WandaMind training design, the evaluation runtime, the system prompt, and the tool schema. The training tables document the method and record counts; the training records are not released.

A SYSTEM IN OPERATION

Figures A.1 and A.2 show two demonstration sessions in the Windows application; dataset and subject identifiers are masked, and neither input is one of the ten RQ2 cohorts. In Figure A.1, a single natural-language request converted the bundled three-series demonstration input (T2w, diffusion, and BOLD) to BIDS; the only manual step was one approval of the pipeline run. The sandbox in which it ran did not include the official `bids-validator`, so validation was run as a second request with the official `bids-validator` (v3.0.2, BIDS schema 1.2.7). It found no errors in the converted files: its only error (`NOT_INCLUDED`) flagged the validator's own output folder, which that run had written inside the dataset directory, and the remaining 35 findings were warnings about recommended but optional metadata. Figure A.2 comes from an earlier session on the bundled `neo-***` DICOM archive. The RQ2 conversions (Section 3.2) were checked with the official `bids-validator`.

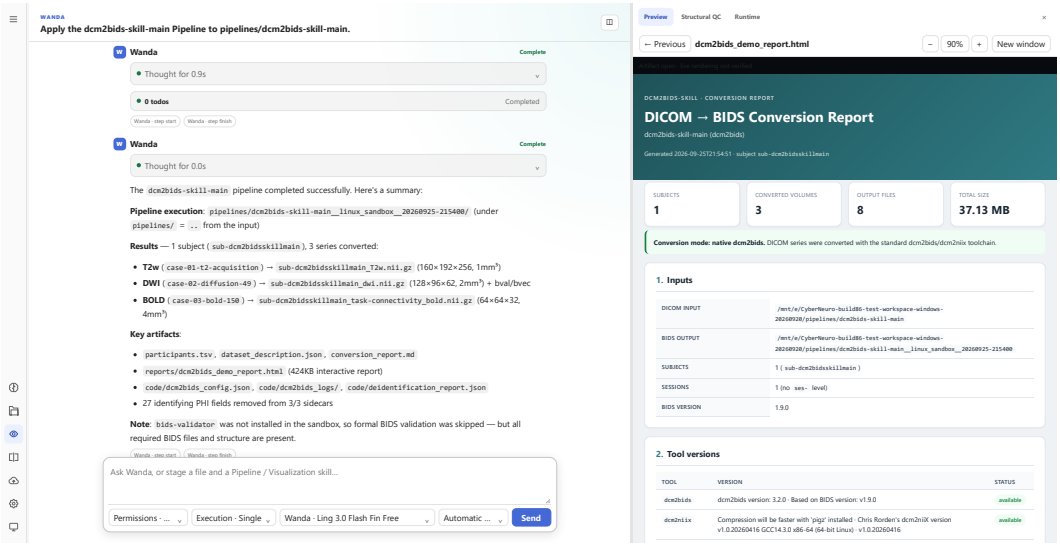

Figure A.1: **Pipeline execution from one natural-language request.** Left: the chat panel for the request "Apply the dcm2bids-skill-main Pipeline to pipelines/dcm2bids-skill-main" (shown in the header), ending in the agent's summary: one subject and three converted series (T2w, $160\times192\times256$ at 1 mm; DWI, $128 \times 96 \times 62$ at 2 mm, with bval/bvec; BOLD, $64 \times 64 \times 32$ at 4 mm), the key artifacts, and 27 identifying fields removed from the three sidecars. Right: the run's conversion report in the preview pane (1 subject, 3 converted volumes, 8 output files, 37.13 MB; dcm2bids 3.2.0 and dcm2niix available). As the reply notes, the official `bids-validator` was not installed in this sandbox, so validation was run as a separate request (see text).

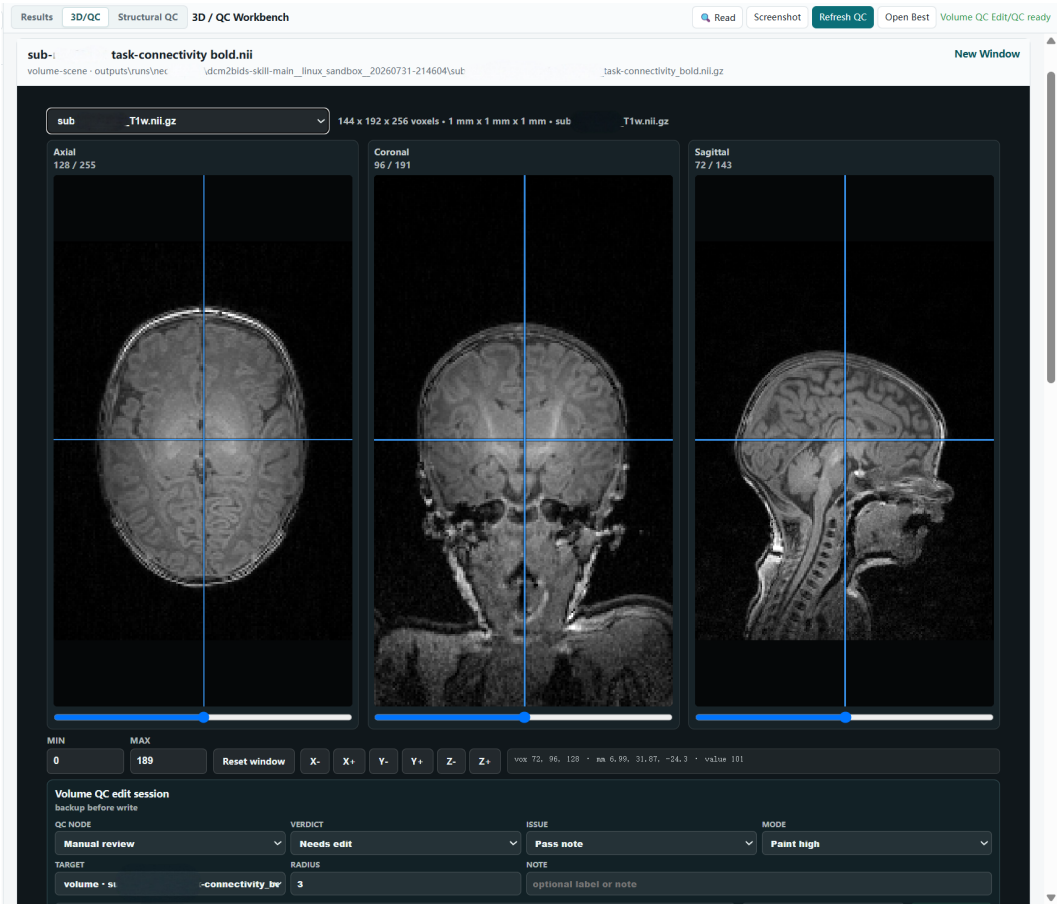

Figure A.2: **In-pane inspection of a pipeline output, no second application.** The 3D/QC Workbench opened on the outputs of an earlier conversion of the bundled `neo-***` archive. The scene header names the converted BOLD series; the panes show the converted T1w volume selected from the same run (`sub-***_T1w.nii.gz`, $144 \times 192 \times 256$ voxels, 1 mm isotropic) in synchronized axial, coronal, and sagittal views with a shared crosshair, window/level control, and voxel/mm readout. Below them is the in-pane *Volume QC edit session* panel (QC node, verdict, issue, paint mode, target, radius, and note). No volume viewer or QC application outside the workbench is opened. This demonstration is not one of the timed RQ3 runs (Section 3.3).

Figures A.3 and A.4 are unedited exports from two further visualization skills in the library, shown here on a second subject (`sub-***`, RIDDLE ADNI4) to illustrate breadth beyond the demonstrations above. Both images are the tool's own rendered output, captured at full resolution.

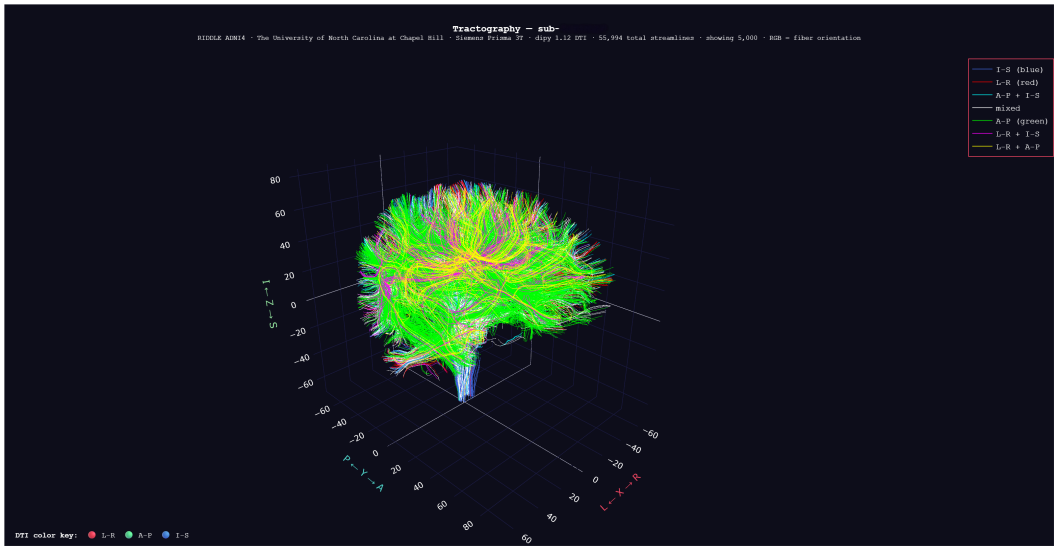

Figure A.3: **Interactive tractography, unedited.** Output of the diffusion tractography visualization skill (`dipy` 1.12) on `sub-***`: 5,000 of 55,994 reconstructed streamlines, colored by local fiber orientation (RGB) with a discrete legend for the seven dominant orientation classes. The camera, axes, and legend are the viewer's default state; nothing in the scene was repositioned before capture.

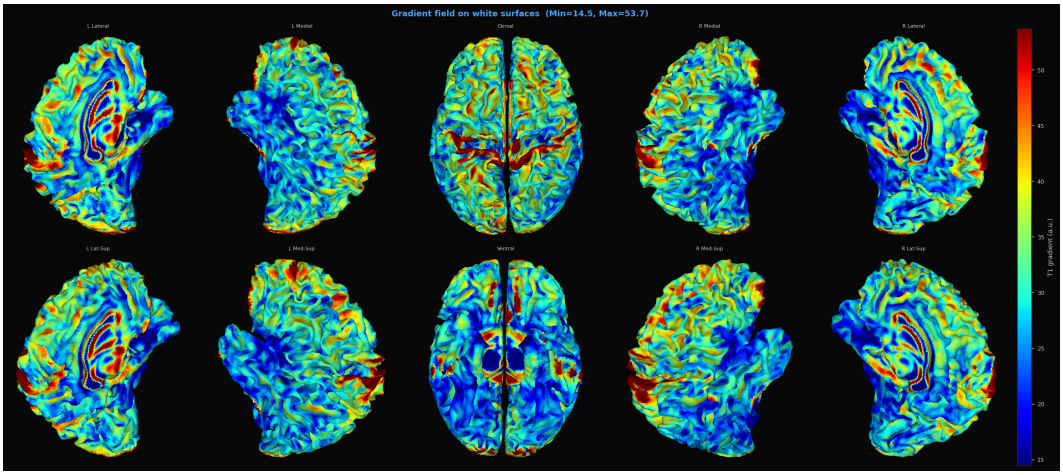

Figure A.4: **CIVET cortical surface QC montage, unedited.** T1 gradient field mapped onto the white-matter surface for `sub-***`, rendered across the ten standard CIVET QC angles (lateral, medial, dorsal, ventral, and superior obliques for each hemisphere) with a shared colorbar (14.5–53.7 a.u.). Produced directly by the CIVET surface skill's QC report generator; no manual view selection or post-processing was applied.

B Canonical CyberNeuro Skill Library

The current release contains **67 skills**: **32 processing pipelines** and **35 visualization and analysis skills**. Folder names are authoritative. The QC workbench additionally bundles three QC entries (`ants_skill`, `freeview`, and the brain-mask correction tool `brainmask_fix`); they are not counted as separate skills.

Table B.1: Canonical CyberNeuro skill library (67 skills).

Category	Skill folder	Wrapped tool / use
Processing pipelines (32)		
Structural MRI	`freesurfer_skill`	FreeSurfer cortical reconstruction, parcellation and morphometry
Structural MRI	`ants_skill`	ANTs registration, brain extraction, N4 bias correction, segmentation and cortical thickness
Structural MRI	`CIVET-skill`	CIVET cortical surfaces and cortical thickness
Structural MRI	`BrainSuite_skill`	BrainSuite cortical surface extraction, diffusion (BDP) and fMRI (BFP) pipelines
Structural / multimodal MRI	`spm-skill`	SPM-style pipeline with NIfTI QC, AAL connectivity summaries and a 3D viewer
Multimodal pipeline	`brain-mri-pipeline-skill-main`	End-to-end BIDS pipeline: functional connectivity from BOLD and DTI maps (FA/MD) from DWI
Brain aging	`dunedinpacni-skill`	DunedinPACNI pace-of-aging scores from FreeSurfer outputs
Functional MRI	`AFNI-skill`	AFNI task and resting-state fMRI analysis
Functional MRI	`CPAC-skill`	C-PAC resting-state processing and connectome derivatives
Functional MRI	`CONN-Toolbox_skill`	CONN resting-state functional-connectivity analysis
Functional MRI	`GIFT-skill`	GIFT group ICA and connectivity analysis
Functional MRI	`fmriprep-skill`	fMRIPrep LTS BIDS App preprocessing on Linux or WSL
Diffusion MRI	`dipy_skill`	DIPY preprocessing, diffusion models (DTI, DKI, CSD) and tractography
Diffusion MRI	`dsi-studio-skill`	DSI Studio reconstruction, tractography and connectomes
Diffusion MRI	`dsi-studio-tractography`	DSI Studio tractography pipeline with QC report
Diffusion MRI	`TractSeg_skill`	TractSeg white-matter bundle segmentation, tracking and QC
Diffusion MRI	`tractseg-bundle-segmentation`	TractSeg bundle-segmentation pipeline with QC report
Infant imaging	`fc_infant_skill`	Neonatal functional connectivity (NiBabies preprocessing)
Infant imaging	`sc_infant_skill`	Neonatal structural connectome (QSIPrep and ACT tractography)
Deep learning	`MONAI-skill`	MONAI preprocessing, 3D viewer and MONAI Label annotation
Deep learning	`Torch-IO`	TorchIO preprocessing and augmentation
Harmonization	`neurocombat-skill`	neuroCombat multi-site harmonization of regional measurements
Harmonization	`neuroharmonize-skill`	neuroHarmonize ComBat with non-linear covariates and voxelwise harmonization
Quality control	`mriqc-report-qc-skill`	MRIQC image-quality metrics and offline group review
Quality control	`generated-3d-qc-pipeline`	Generated 3D QC pipeline with metrics and HTML report
I/O and BIDS	`dcm2bids-skill-main`	DICOM-to-BIDS conversion, including compressed archives
I/O and BIDS	`bids-validator-skill`	Official BIDS validator with a grouped, actionable report
Multi-purpose	`fsl-agent-skill`	FSL suite (BET, FAST, FLIRT/FNIRT, FEAT, MELODIC, TBSS, eddy/topup)
Multi-purpose	`bioimagesuite-skill`	BioImage Suite Web processing modules and QC reports
Multi-purpose	`bioimage-suite-web-registration`	BioImage Suite Web registration pipeline with QC report
Multi-purpose	`mipav-skill`	MIPAV image processing and plugin execution
Multi-purpose	`mipav-batch-processing`	MIPAV batch-processing pipeline with QC report
Visualization and analysis (35)		
Volume viewing	`fsleyes-visual-skill`	FSLeyes overlays, BET/FAST/registration QC and 3D mask viewers
Volume viewing	`freeview-browser-skill`	Freeview/FreeBrowse review and editing of volumes, surfaces and overlays
Volume viewing	`itk-snap-skill`	ITK-SNAP volume and label-map review (bundled runtime)
Volume viewing	`Mango-skill`	Mango/WebMango image review, slice galleries and QC
Volume viewing	`mricrogl-skill`	MRIcroGL volume rendering, glass brain and statistical overlays
Volume viewing	`niivue-skill`	NiiVue WebGL viewer for volumes, overlays, 4D series and meshes
Volume viewing	`slicer-linux-skill`	3D Slicer volume review and headless slice renders (bundled Linux runtime)
Surface and atlas	`connectome-workbench-skill`	Connectome Workbench CIFTI/GIFTI data, grayordinates and connectomes
Surface and atlas	`ggseg-skill`	ggseg 2D/3D atlas maps of regional measurements
Surface and atlas	`brainrender-atlas-viewer`	Brainrender atlas views and previews

Table B.1 (continued)

Category	Skill folder	Wrapped tool / use
Surface and atlas	`surf-ice-linux-skill`	Surf Ice surface and tractography scenes (bundled Linux runtime)
Tractography	`afni-tractography-viewer`	AFNI-style interactive tractography QC preview
Tractography	`dipy-horizon-tract-viewer`	DIPY Horizon-style review of TRK/TCK tractograms
Tractography	`trackvis-skill`	TrackVis tract inspection, streamline statistics and density maps
Tractography	`FURY_skill`	FURY diffusion glyphs (ODF/fODF) in the PyVista viewer
Connectivity and networks	`brain-network-vis`	Connectivity heatmaps, brain network graphs and regional maps
Connectivity and networks	`fc-vis`	Interactive functional-connectivity matrices and glass-brain connectomes
Connectivity and networks	`circos-skill`	Circos circular connectivity figures
Connectivity and networks	`gephi-skill`	Gephi graph export (GEXF/GraphML) and network metrics
Connectivity and networks	`neuromarvl-skill`	NeuroMArVL linked surface and network views
Connectivity and networks	`visbrain-skill`	Visbrain interactive connectivity reports
3D rendering	`pyvista_skill`	PyVista 3D surface QC, parcellation overlays and tractography
3D rendering	`paraview-skill`	ParaView filters, volume rendering and time series
3D rendering	`vedo-skill`	Vedo/VTK review of meshes, volumes and point clouds
3D rendering	`generated-3d-artifact-viewer`	Viewer for generated 3D artifacts
3D rendering	`layered-output-visualization`	Layered visualization of pipeline outputs
Statistics	`capcov-skill`	Covariate-assisted principal (CAP) covariance regression
Statistics	`medmethods-skill`	Causal mediation analysis (MedMethods)
Statistics	`subnetr-skill`	Connectome subnetwork detection and power analysis (subnetr)
Plotting and figures	`nilearn-skill`	Nilearn volume, surface, statistical-map and connectome plots
Plotting and figures	`academic-figure-skill`	Publication figure creation, revision and quality review
Plotting and figures	`figures4papers-skill`	figures4papers publication charts from measurement tables
Plotting and figures	`paper-figures`	Interactive and static figures in the figures4papers house style
Plotting and figures	`tavotto-skill`	Tavotto figure workbench for layout adjustment and export
Electrophysiology	`mne-source-viz-skill`	MNE-Python EEG/MEG analysis and source visualization

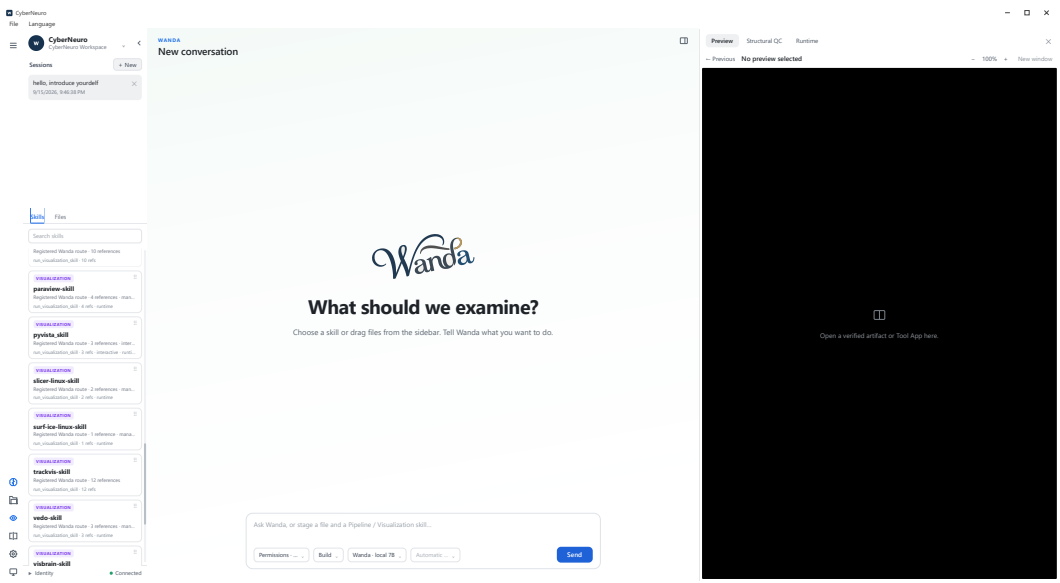

Figure B.1: **CyberNeuro 3D/QC Workbench.** The interface organizes volume, surface, parcellation, diffusion, tractography, and connectome quality-control applications. Each panel reports available artifacts and provides entrypoints to the corresponding validated skills.

C SKILL GENERATION AND VISUALIZATION DETAILS

C.1 SELF-EVOLVING SKILL GENERATION

Figure C.1 summarizes the full pipeline. The skill generation process separates scientific reasoning from environment-specific execution details. An LLM first converts source material into an environment-independent procedure graph containing workflow steps, dependencies, and typed capability requests such as executables, modules, paths, packages, containers, or scheduler resources. The graph is then passed through a deterministic validation gate that rejects malformed structure and prevents the model from directly introducing machine-specific commands, paths, or configuration details.

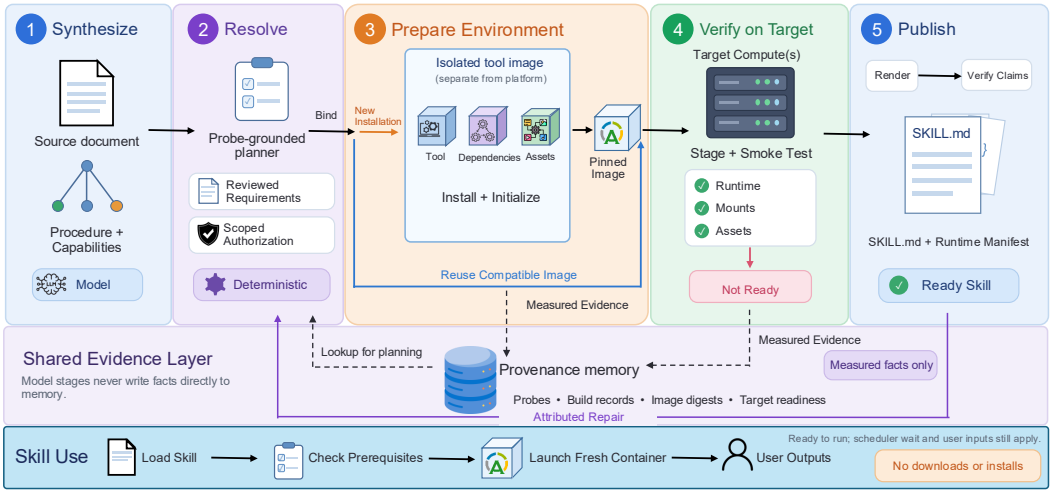

Figure C.1: Self-evolving skill generation. A source document is synthesized into a procedure with capability requests (1), which a deterministic, probe-grounded planner resolves (2). Missing tools are installed into an isolated, pinned tool image separate from the platform, or a compatible image is reused (3); the skill is then staged and smoke-tested on the target compute (4) before SKILL.md and its runtime manifest are rendered and their claims verified (5). A shared provenance memory holds only measured evidence (probes, build records, image digests, target readiness); model stages never write to it directly. A ready skill runs in a fresh container with no downloads or installs.

After validation, a deterministic binder resolves each capability request against the target HPC environment using predefined probes. Each request produces either a verified binding backed by an observation or an explicit abstention when the required resource cannot be established. If missing capabilities can be installed, the system may construct a container using predefined, human-reviewed recipes and verify that the required tools and resources function correctly inside it. Previously unresolved capabilities are then rechecked against the constructed environment.

Once all required capabilities are resolved, a deterministic renderer combines the model-generated procedure with the verified bindings to produce the final SKILL.md. Concrete details such as paths, modules, executables, scheduler partitions, and runtime invocations originate only from verified bindings rather than model-generated text. The rendered skill is finally re-probed to ensure that its machine-checkable environment claims remain valid.

C.2 VISUALIZATION OPERATIONS

The co-visualization panel (Section 2.9) currently provides the following operations.

Inspection. Rotation, panning, zoom, slice navigation, threshold and opacity control, color-map selection, and screenshot export with camera state. Supported 4D views link time navigation to

volume, edge-weight, and ROI time-series displays, and ROI inspection, atlas-label lookup, and node/edge highlighting feed selected regions back into subsequent agentic requests.

Editing and figure customization. Masks, segmentations, surface labels, and edge sets can be edited through manual tools or constrained natural-language commands, and plotting code for statistical figures can be revised conversationally and re-executed. Each edit creates a derived artifact and a provenance node, so the original pipeline output is preserved and presentation changes remain reproducible.

Batch visualization. A visualization request over many inputs (for example, one rendering per subject of a cohort) is submitted as a single batch of up to 200 independent runs. Every item passes the same capability resolution and argument checks as a single run before anything executes; the items then run strictly one at a time, with a continue-or-stop policy on item failure, and the batch state is written atomically after every transition, so a closed session, a reloaded page, or a restarted backend resumes the same batch. Cancel, retry, and resume act on the recorded state, and each item writes its artifact to its own declared output location.

C.3 Runtime environments

Skills that need an additional runtime declare it in `cyberneuro.runtime.json` or in their capability manifest. A declaration pins the toolchain version (for example, `ants=2.6.5`), lists the executables and Python modules that must be present, and states whether the profile can be installed automatically. The dispatcher selects the environment from this declaration; the model never chooses it. A profile is installed when a skill first needs it (automatically where supported), uses already installed local tools when all required executables are present, and is checked against its declaration before reuse. Profiles are shared: the 48 skills that need an additional runtime map to 39 profiles (for example, five visualization skills share one base profile and three share an MNE profile), and the remaining skills run in the embedded Python environment or as native binaries.

D System-Level Capability Comparison

Table D.1 summarizes the system-level capabilities of CyberNeuro and representative related systems.

Table D.1: System-level capability comparison. Filled and open circles denote native and partial support, respectively; blank cells indicate unsupported capabilities.

System	Agent	Runtime	Pre-execution check	Artifact check	Visual QC	Local	Replay
3D Slicer		●		○	●		○
Nipype/BIDS Apps		●	○	●			●
brainlife.io		●	○	●	○		●
MONAI Label	○	●	○	○	●		○
NeuroClaw	●	●	○	●	○	○	○
CyberNeuro	●	●	●	●	●	●	●

E Per-Cohort DICOM-to-BIDS Results

Tables E.1–E.3 report per-cohort results for the three DICOM-to-BIDS configurations evaluated in RQ2 (Section 3.2). The WandaMind configuration runs the local WandaMind model; both cloud configurations, CyberNeuro (cloud) and NeuroClaw (cloud), use DeepSeek V4 Pro as the backend model, and their token usage was counted with the same tool. Totals are summed over the ten cohorts, except Tokens/NIfTI and s/NIfTI (seconds per NIfTI volume), which divide total tokens and total time by the total number of NIfTI volumes. The NIfTI counts differ by design. NeuroClaw first converts every discovered DICOM series to NIfTI and organizes BIDS in a second stage, so its 188 volumes include 127 from series outside the anatomical scope of the task. CyberNeuro converts end to end: it classifies each series from its DICOM metadata before conversion and writes only the

61 in-scope volumes directly into BIDS. The totals therefore include this extra conversion work for NeuroClaw, whereas the per-volume columns compare the two systems at the same unit.

Table E.1: CyberNeuro DICOM-to-BIDS results using WandaMind.

Cohort	NIfTI	Tokens	Tokens/NIfTI	Time (s)	s/NIfTI	Interventions
ADNI	10	6,865	686.5	165.3	16.53	0
ADNIDOD	3	6,919	2,306.3	412.7	137.57	0
AIBL	7	6,899	985.6	94.6	13.51	0
BLSA_OPEN	4	6,882	1,720.5	285.2	71.30	0
MCSA	2	6,894	3,447.0	266.1	133.05	0
NIFD	9	6,855	761.7	190.5	21.17	0
POINTER	4	6,860	1,715.0	73.3	18.33	0
PPMI	10	6,885	688.5	226.0	22.60	0
SCAN	4	6,863	1,715.8	198.1	49.53	0
WRAP	8	6,853	856.6	145.2	18.15	0
Total	**61**	**68,775**	**1,127.5**	**2,057.0**	**33.72**	**0**

Table E.2: CyberNeuro DICOM-to-BIDS results using the cloud backend (DeepSeek V4 Pro).

Cohort	NIfTI	Tokens	Tokens/NIfTI	Time (s)	s/NIfTI	Interventions
ADNI	10	44,638	4,463.8	135.9	13.59	0
ADNIDOD	3	44,350	14,783.3	313.4	104.47	0
AIBL	7	26,339	3,762.7	75.3	10.76	0
BLSA_OPEN	4	26,583	6,645.8	319.2	79.80	0
MCSA	2	26,489	13,244.5	228.9	114.45	0
NIFD	9	44,283	4,920.3	169.6	18.84	0
POINTER	4	54,200	13,550.0	90.0	22.50	0
PPMI	10	35,644	3,564.4	239.2	23.92	0
SCAN	4	62,688	15,672.0	231.8	57.95	0
WRAP	8	35,541	4,442.6	184.0	23.00	0
Total	**61**	**400,755**	**6,569.8**	**1,987.3**	**32.58**	**0**

Table E.3: NeuroClaw results using the same cloud backend (DeepSeek V4 Pro).

Cohort	NIfTI	Tokens	Tokens/NIfTI	Time (s)	s/NIfTI	Interventions
ADNI	20	253,571	12,678.6	296.0	14.80	3
ADNIDOD	20	230,258	11,512.9	212.4	10.62	3
AIBL	20	147,993	7,399.7	165.8	8.29	2
BLSA	20	244,176	12,208.8	482.2	24.11	3
MCSA	6	93,391	15,565.2	119.2	19.87	2
NIFD	40	278,419	6,960.5	249.2	6.23	2
POINTER	8	168,419	21,052.4	164.7	20.59	2
PPMI	20	216,160	10,808.0	255.1	12.76	2
SCAN	12	183,086	15,257.2	235.1	19.59	2
WRAP	22	186,019	8,455.4	385.7	17.53	3
Total	**188**	**2,001,492**	**10,646.2**	**2,565.4**	**13.65**	**24**

F Timing Definitions for the Visualization and QC Comparison

Table 2 (RQ3) reports the median time over $K=3$ repetitions for each visualization and QC task (V1 overlay, V2 connectome, V3 QC/replay) under a conventional multi-tool condition and under CyberNeuro. The two times measure different quantities:

- **Conventional: human decision and tool-invocation time.** The time a human operator needs to decide which tools the task requires and to invoke them correctly, following the

recorded action trace. It covers preparing the software, starting each of the three or four applications the task uses, transferring files between them, and configuring each view by hand. Even when a tool is already installed, many neuroimaging tools have no graphical interface, so this time also includes working out how to run each tool and checking that it works.

- **CyberNeuro: machine time.** The median active execution time the workbench needs to perform the same task. The choice and invocation of tools, which the conventional condition leaves to the human, are made by the system, so this time covers only the workbench's own execution. Each skill pins its tools, their versions, and their download links, so no tool has to be searched for or tried out; the only possible additional step is a one-time installation of the skill's environment (Appendix C.3), which is not part of the reported machine time.

The comparison therefore sets the human time needed to decide on and invoke the correct tools against the machine time needed to carry out the task. The resulting reduction of more than 99.94% quantifies the operational burden removed from the user, not a difference in computational speed. Application, switch, and action counts are medians from the same action traces.

G QC-62 BENCHMARK

G.1 PURPOSE AND CONSTRUCTION

QC-62 is a compact, held-out probe of *agentic* neuroimaging quality-control (QC) behaviour: given a single-turn user request and a fixed tool interface (Sec. L), the agent must select the correct tool, construct schema-valid arguments, reach the correct QC judgment, and, critically, *abstain* (clarify / refuse / escalate) when a request exceeds local Tier-0 capability or would fabricate a non-existent derivative. It contains **62 cases across 14 categories** (Table G.1). Each case is a JSON record with fields `id`, `category`, `tier`, `prompt`, and an `expect` object holding `tool_sequence` (the reference first action / plan), `key_args` (the required argument values or the target verdict), and `must_not` (behaviours that fail the case).

Provenance. QC-62 was assembled before WandaMind was trained. While reviewing execution logs of the workbench driven by cloud APIs and other local models, we collected requests on which the agent failed or took an obviously roundabout path, and curated them into 62 representative cases across the 14 categories in Table G.1.

Table G.1: QC-62 category distribution (62 cases).

Category	#
quality_judgment	11
tool_selection	6
command_construction	6
qc_interpretation	6
failure_recovery	5
multi_step_plan	4
bids_path	4
escalation	4
pipeline_knowledge	4
domain_qa	3
metric_grounding	3
no_fake	2
concept_grounding	2
knowledge_to_action	2

G.2 HELD-OUT AND DECONTAMINATION RULES

- **Physical hold-out.** The 62 QC-62 records (`qc62_tasks.jsonl`) are never part of any training file. Training and evaluation records are grouped by *template family* and *skill* so that near-duplicate variants cannot straddle the train/eval split.
- **Jaccard overlap check.** Every candidate training record is compared against all 62 eval prompts by token-set overlap. The highest Jaccard similarity over all record–prompt pairs is 0.55, and no record was removed on this basis.
- **Validator gate.** During data generation, candidates that reference a non-existent skill, produce malformed tool JSON, or fail a post-filter are rejected (41 candidates rejected: 30 at validation, 10 at parsing, 1 at post-filter) and never enter the training files.
- **Grounded labels.** QC verdict labels use real metric conventions (e.g., FreeSurfer Euler number, registration NCC/NMI, segmentation Dice), not synthetic thresholds.

G.3 SCORING RUBRIC

We report a single-gate **Overall** pass rate and four diagnostic axes.

- **Overall** (all 62). An *action* case (non-empty `tool_sequence`) passes iff the model's first tool call equals the reference first tool and no `must_not` phrase appears. A *judgment* case (empty `tool_sequence`) passes iff the required verdict keywords appear and no `must_not` phrase appears.

- **Tool sel.** (`tool_selection` cases, n=6): first tool equals the reference tool.
- **Arg. valid** (arg-bearing cases, n=16; multi-turn roll-out with synthetic tool returns): the model emits a schema-valid `run_pipeline` call (valid JSON; required `pipeline_name` and `input_batch` present).
- **Abstain recall** (`escalation`+`no_fake`, n=6): the model correctly withholds action, issues *no* tool call, and escalates / refuses / reports "blocked".
- **False act** $\downarrow$ (same n=6): the model invokes *any* tool when it should clarify, refuse, or escalate (lower is better).

Decoding for scoring is deterministic (`temperature`=0, `max_tokens`=384, identical for all models); the 27B reasoning baseline is run in non-thinking mode.

G.4 QC-62 RESULTS

Table G.2: WandaMind evaluation on held-out QC-62. Values reproduce Table 1. All values are percentages; lower false action is better.

Model	Overall	Tool sel.	Arg. valid	Abstain recall	False act↓
Base 7B	40.3	83.3	93.8	0.0	100.0
Local 27B	48.4	100.0	31.3	50.0	50.0
WandaMind 7B	72.6	83.3	87.5	100.0	0.0
WandaMind Q4_K_M	62.9	66.7	81.3	100.0	0.0

H WANDAMIND REPRODUCIBILITY DETAILS

H.1 TRAINING CONFIGURATION

Table H.1: WandaMind QLoRA configuration, corresponding to Figure 2 and Section 2.7.

Field	Configuration
Base checkpoint	Qwen2.5-7B-Instruct
Base-weight format	4-bit NF4; base weights frozen
Adapter targets	`q_proj`, `k_proj`, `v_proj`, `o_proj`, `gate_proj`, `up_proj`, and `down_proj`
LoRA parameters	rank r=32, α=64, α/r=2.0, dropout 0.05
Trainable parameters	80.7M (1.06% of the complete model)
Objective	Response-only loss; system instructions, tool descriptions, and user messages masked
Optimization	Unsloth 4-bit QLoRA, three epochs

H.2 TRAINING DATA AND STAGES

Table H.2: Validated training-trajectory composition. The 80 targeted repairs and 21 hardening counterexamples form the 101 targeted examples reported in the main text.

Source	#	Role
Teacher-generated trajectories	315	Tool selection and schema-valid argument construction
Rule-based expansions	306	Coverage of known skill and parameter combinations
Targeted repairs	80	Difficult and underrepresented neuroimaging operations
Hardening counterexamples	21	Act, refuse, and search decision-boundary failures
Validated core pool	**722**	Used across the three training rounds

An additional 41 candidate records were rejected by the schema, parsing, and post-filter gates and are not included in the 722-record pool.

Table H.3: Three-stage training design.

Stage	Purpose	Supervision emphasis
Round 1	Supervised initialization	Validated tool-use trajectories and response-only supervision
Round 2	Act-oriented augmentation	Tool selection and executable command construction
Round 3	Adversarial hardening	Refuse and search counterexamples that restore the fail-loud boundary

H.3 Quantization and deployment

Table H.4: WandaMind deployment conversion.

Step	Operation
1	Merge the trained LoRA adapter with the base model weights
2	Convert the merged Hugging Face checkpoint to GGUF
3	Quantize the GGUF checkpoint to Q4_K_M for local deployment

I Evaluation runtime

The four evaluated model conditions used the following serving settings.

- **bf16 (Base and WandaMind 7B):** vLLM 0.21.0. WandaMind is served as the base checkpoint with the LoRA adapter via `--enable-lora` (`max-lora-rank=32`); bf16 adapter application is numerically equivalent to the merged weights. Tool calls: `--enable-auto-tool-choice --tool-call-parser hermes`. `max-model-len=8192`, `gpu-memory-utilization=0.90`.
- **Local 27B (Qwen3.6-27B):** vLLM, AWQ (`--quantization awq_marlin`), `--enforce-eager`, tensor-parallel $= 2$. As a reasoning model it is run in **non-thinking** mode via a chat template that sets `enable_thinking=false` (the `/no_think` prefix is insufficient); both JSON and XML `<function=...>` tool formats are parsed.
- **WandaMind Q4_K_M:** `llama.cpp` `llama-server` with `--jinja` for tool calling, `-ngl 99`, `-c 8192`.
- **Decoding:** `temperature`=0, `max_tokens`=384, stop `<|im_end|>`,`<|endoftext|>`, identical across all four configurations.
- **Cluster environment:** `NCCL_P2P_DISABLE=1`, `NCCL_IB_DISABLE=1`, `--disable-custom-all-reduce`, and `VLLM_ATTENTION_BACKEND=TORCH_SDPA`; FlashInfer sampling and attention backends disabled.

Table I.1: QC-62 evaluation environment.

Component	Version / spec
GPUs	4× NVIDIA RTX 4090, 24 GB, no NVLink
Serving (bf16 / AWQ)	vLLM 0.21.0
Serving (GGUF)	llama.cpp `llama-server` (Unsloth build)
Fine-tuning	Unsloth QLoRA, 4-bit NF4
Merge / adapters	`transformers` and PEFT
Evaluation harness	lm-evaluation-harness 0.4.12
Datasets backend	`datasets` 3.6.0
WandaMind deployment	GGUF Q4_K_M via `llama-quantize`

J QC-62 PER-ITEM EVALUATION RECORDS, SCORER, AND TOOL-CALL PARSING

This section gives the tool-call parsing rules and the scoring code behind Table G.2, and lists every model's first tool call and pass/fail decision for each of the 62 cases.

J.1 TOOL-CALL PARSING RULES

If the serving endpoint returns structured `tool_calls` natively (vLLM `--tool-call-parser hermes` for the JSON format), those are used directly. When that field is empty (e.g., a model that emits tool calls inside message content, or the 27B XML format), the following fallback parser recovers calls from the text. It accepts both the Qwen JSON form (`<tool_call>{...}</tool_call>`) and the Qwen3 XML form (`<function=NAME><parameter=K>V</parameter>...</function>`). The parsed first tool is `calls[0].name`.

```
def parse_tool_calls(text):
   calls = []
   # (1) blocks delimited by <tool_call> ... </tool_call>
   for blob in re.findall(r"<tool_call>\s*(.*?)\s*</tool_call>", text, re.S):
      try:
         item = json.loads(blob) # JSON tool call
      except Exception: # XML inside <tool_call>
         m = re.search(r"<function=([^>\s]+)>", blob)
         args = {k: v.strip() for k, v in re.findall(
            r"<parameter=([^>\s]+)>\s*(.*?)\s*</parameter>", blob, re.S)}
         item = {"name": m.group(1), "arguments": args} if m else None
      if item:
         a = item.get("arguments", {})
         if isinstance(a, str):
            try: a = json.loads(a)
            except Exception: pass
         calls.append({"name": item.get("name"), "arguments": a})
   # (2) bare XML function calls outside <tool_call> (some Qwen3 variants)
   if not calls:
      for m in re.finditer(r"<function=([^>\s]+)>(.*?)</function>", text, re.S):
         args = {k: v.strip() for k, v in re.findall(
            r"<parameter=([^>\s]+)>\s*(.*?)\s*</parameter>", m.group(2), re.S)}
         calls.append({"name": m.group(1), "arguments": args})
   return calls
```

J.2 SCORING CODE

The single-gate **Overall** decision (Sec. G.3) is computed as below; the four diagnostic axes are derived from the same per-item signals.

```
def overall_gate(task, msg):
   text = (msg.get("content") or "").lower()
   for bad in task["expect"].get("must_not", []): # forbidden behaviour -> fail
      if bad.lower() in text:
         return 0
   seq = task["expect"].get("tool_sequence") or []
   first = (tool_names(msg) or [None])[0] # parsed first tool call
   if seq: # action case
      return 1 if first == seq[0] else 0
   want = " ".join(map(str, task["expect"].values())).lower() # judgment case
   keys = [k for k in VERDICT_KEYWORDS if k in want]
   return 1 if (not keys or any(k in text for k in keys)) else 0
```

J.3 PER-ITEM FIRST TOOL CALL AND PASS/FAIL

Table J.1 gives, for every case and every model, the parsed first tool call and the Overall pass/fail decision. Tool abbreviations: `pcs=pipeline_code_search`, `run=run_pipeline`, `read=read_file`, `qcS=qc_agent_start_session`, `qcA=qc_agent_advance`, `patch=pipeline_file_patch`, `--`=no tool call. ✓ / × = Overall pass / fail.

Table J.1: QC-62 per-item parsed first tool call and Overall pass/fail for all four models.

ID	Category	Base 7B	Local 27B	WandaMind 7B	WM Q4_K_M
T01	tool selection	pcs ✓	pcs ✓	pcs ✓	pcs ✓
T02	tool selection	run ×	pcs ✓	pcs ✓	pcs ✓
T03	tool selection	pcs ✓	pcs ✓	pcs ✓	run ×
T04	tool selection	pcs ✓	pcs ✓	pcs ✓	pcs ✓
T05	tool selection	pcs ✓	pcs ✓	pcs ✓	pcs ✓
T06	tool selection	pcs ✓	pcs ✓	run ×	run ×
T07	multi step plan	pcs ✓	pcs ✓	pcs ✓	run ×
T08	multi step plan	pcs ✓	pcs ✓	read ×	pcs ✓
T09	multi step plan	pcs ✓	pcs ✓	pcs ✓	pcs ✓
T10	multi step plan	pcs ×	pcs ×	pcs ×	pcs ×
T11	bids path	pcs ×	-- ×	-- ×	-- ×
T12	bids path	pcs ✓	pcs ✓	read ✓	read ✓
T13	bids path	pcs ✓	pcs ✓	-- ✓	-- ✓
T14	bids path	pcs ×	pcs ×	read ✓	-- ×
T15	command construction	pcs ×	pcs ×	read ×	pcs ×
T16	command construction	pcs ×	pcs ×	run ✓	run ✓
T17	command construction	pcs ×	pcs ×	run ✓	pcs ×
T18	command construction	pcs ×	pcs ×	run ✓	pcs ×
T19	command construction	pcs ×	pcs ×	run ✓	run ✓
T20	command construction	patch ×	pcs ×	pcs ×	pcs ×
T21	qc interpretation	qcS ✓	qcS ✓	qcS ✓	qcS ✓
T22	qc interpretation	qcA ✓	qcA ✓	qcA ✓	qcA ✓
T23	qc interpretation	qcS ×	qcS ×	-- ✓	-- ✓
T24	qc interpretation	pcs ×	-- ×	-- ×	-- ×
T25	qc interpretation	qcA ✓	qcS ×	qcA ✓	qcA ✓
T26	failure recovery	pcs ×	pcs ×	pcs ×	run ✓
T27	failure recovery	-- ×	pcs ✓	-- ×	-- ×
T28	failure recovery	pcs ✓	pcs ✓	read ×	read ×
T29	failure recovery	-- ×	-- ×	pcs ✓	-- ×
T30	failure recovery	pcs ✓	pcs ✓	-- ✓	-- ✓
T31	escalation	pcs ×	pcs ×	-- ✓	-- ✓
T32	escalation	pcs ×	-- ✓	-- ✓	-- ✓
T33	escalation	pcs ×	-- ×	-- ✓	-- ✓
T34	escalation	pcs ×	pcs ×	-- ✓	-- ✓
T35	no fake	pcs ×	pcs ×	-- ×	-- ×
T36	no fake	pcs ×	-- ✓	-- ✓	-- ✓
T37	domain qa	pcs ✓	-- ✓	-- ✓	-- ✓
T38	domain qa	pcs ✓	-- ✓	-- ✓	-- ✓
T39	domain qa	pcs ✓	-- ✓	-- ✓	-- ✓
T40	concept grounding	pcs ✓	pcs ✓	-- ✓	-- ✓
T41	pipeline knowledge	pcs ×	pcs ×	-- ×	-- ×
T42	pipeline knowledge	pcs ✓	pcs ✓	-- ✓	-- ✓
T43	pipeline knowledge	pcs ✓	pcs ✓	-- ✓	-- ✓
T44	metric grounding	-- ×	-- ×	-- ×	-- ✓
T45	knowledge to action	qcA ✓	qcS ×	qcA ✓	qcA ✓
T46	knowledge to action	pcs ✓	-- ✓	-- ✓	-- ✓
T47	quality judgment	pcs ×	qcS ×	-- ✓	-- ✓
T48	quality judgment	pcs ×	qcS ×	-- ✓	-- ✓
T49	quality judgment	-- ×	-- ✓	-- ✓	-- ×
T50	quality judgment	qcS ×	-- ×	-- ✓	-- ×
T51	quality judgment	pcs ×	-- ×	-- ✓	-- ✓
T52	quality judgment	-- ×	-- ×	qcA ✓	-- ×
T53	quality judgment	pcs ×	-- ✓	-- ✓	-- ✓
T54	quality judgment	pcs ×	-- ✓	-- ✓	-- ✓
T55	pipeline knowledge	pcs ×	pcs ×	-- ×	-- ×
T56	quality judgment	pcs ×	pcs ×	-- ×	-- ×
T57	qc interpretation	qcS ✓	qcS ✓	qcS ✓	pcs ×
T58	metric grounding	pcs ✓	-- ✓	-- ✓	-- ✓
T59	metric grounding	qcS ×	-- ×	-- ✓	-- ×
T60	quality judgment	qcS ×	qcS ×	qcA ×	-- ✓
T61	quality judgment	pcs ×	pcs ×	-- ×	-- ✓
T62	concept grounding	pcs ×	pcs ×	-- ✓	-- ✓

Table J.1 (continued)

ID	Category	Base 7B	Local 27B	WandaMind 7B	WM Q4_K_M
Overall		**25/62**	**30/62**	**45/62**	**39/62**

K SYSTEM PROMPT

The identical system prompt below is used at training and evaluation for all models.

```
You are the CyberNeuro local agent. You orchestrate neuroimaging skills on institution-
     controlled hardware by calling tools. Rules: (1) Discover skills with
     pipeline_code_search before running one. (2) Run only real skills via run_pipeline;
     never invent a pipeline name or an argument. (3) Every run must yield a real derivative
     ; if a run is config-only, degraded, or produces no derivative, report it as blocked,
     never as success. (4) If a task exceeds local Tier-0 capability, say so and recommend
     escalating to Tier-1/2; do not fabricate a result. (5) Be concise; end with output
     paths, warnings, and next steps.
```

L TOOL SCHEMA

The six-tool OpenAI-style function interface exposed to every model.

```
[
  {
    "type": "function",
    "function": {
      "name": "pipeline_code_search",
      "description": "Search the skill library and pipeline source for matching skills or code.",
      "parameters": {
        "type": "object",
        "properties": {
          "query": {
            "type": "string"
          },
          "pipeline_name": {
            "type": "string"
          }
        },
        "required": [
          "query"
        ]
      }
    }
  },
  {
    "type": "function",
    "function": {
      "name": "run_pipeline",
      "description": "Run a real neuroimaging skill/pipeline on an input batch.",
      "parameters": {
        "type": "object",
        "properties": {
          "pipeline_name": {
            "type": "string"
          },
          "input_batch": {
            "type": "string"
          },
          "mode": {
            "type": "string"
          },
          "params": {
            "type": "object"
          }
        },
        "required": [
          "pipeline_name"
        ]
      }
    }
  },
  {
    "type": "function",
    "function": {
      "name": "read_file",
      "description": "Read a file inside the workspace.",
      "parameters": {
        "type": "object",
        "properties": {
          "path": {
            "type": "string"
          }
```

```
      },
      "required": [
        "path"
      ]
    }
  }
},
{
  "type": "function",
  "function": {
    "name": "pipeline_file_patch",
    "description": "Create or edit a pipeline file behind a confirmation gate.",
    "parameters": {
      "type": "object",
      "properties": {
        "operation": {
          "type": "string",
          "enum": [
            "create",
            "edit"
          ]
        },
        "path": {
          "type": "string"
        },
        "content": {
          "type": "string"
        },
        "pipeline_name": {
          "type": "string"
        },
        "reason": {
          "type": "string"
        }
      },
      "required": [
        "operation",
        "path",
        "pipeline_name",
        "reason"
      ]
    }
  }
},
{
  "type": "function",
  "function": {
    "name": "qc_agent_start_session",
    "description": "Start the structural QC agent on a subject.",
    "parameters": {
      "type": "object",
      "properties": {
        "subject": {
          "type": "string"
        },
        "modality": {
          "type": "string",
          "enum": [
            "T1w",
            "T2w"
          ]
        }
      },
      "required": [
        "subject"
      ]
    }
  }
},
{
  "type": "function",
  "function": {
    "name": "qc_agent_advance",
    "description": "Advance the QC loop with a decision.",
    "parameters": {
      "type": "object",
      "properties": {
        "decision": {
          "type": "string",
          "enum": [
            "approve",
            "reject",
            "menu"
          ]
        },
        "note": {
          "type": "string"
        }
      },
      "required": [
        "decision"
      ]
```

```
      }
    }
  }
]
```

M QC-62: ALL 62 CASES

Table M.1 lists every case: identifier, category, the verbatim user prompt, and the reference `expect` (first-action tool sequence, required key arguments / target verdict, and forbidden `must-not` behaviours). Judgment cases with no reference tool are scored by the verdict-keyword rubric of Sec. G.3. For typesetting portability, T03 is shown in English translation; the released JSONL retains its verbatim Chinese prompt.

Table M.1: The complete QC-62 held-out benchmark (62 cases).

ID	Category	Prompt	Reference (expect)
T01	tool selection	Brain-extract sub-***'s T1 with FSL.	**seq:** pipeline_code_search → run_pipeline **args:** pipeline_name=fsl-agent-skill; input_batch=sub-*** **must-not:** invent pipeline name
T02	tool selection	Run FreeSurfer recon-all on sub-***.	**seq:** pipeline_code_search → run_pipeline **args:** pipeline_name=freesurfer_skill; input_batch=sub-***
T03	tool selection	[Chinese] Perform cortical reconstruction for sub-***.	**seq:** pipeline_code_search → run_pipeline **args:** pipeline_name=freesurfer_skill
T04	tool selection	Compute ReHo for sub-*** with AFNI.	**seq:** pipeline_code_search → run_pipeline **args:** pipeline_name=AFNI-skill; input_batch=sub-***
T05	tool selection	Tissue-segment sub-*** (GM/WM/CSF).	**seq:** pipeline_code_search → run_pipeline **args:** pipeline_name=fsl-agent-skill
T06	tool selection	Register sub-***'s T1 to MNI with ANTs.	**seq:** pipeline_code_search → run_pipeline **args:** pipeline_name=ants_skill; input_batch=sub-***
T07	multi step plan	Convert neo-*** DICOMs to BIDS then run infant functional connectivity.	**seq:** pipeline_code_search → run_pipeline → run_pipeline **args:** step1=dcm2bids-skill-main; step2=fc_infant_skill **must-not:** run fc before conversion
T08	multi step plan	Take sub-*** from raw T1 to QC'd cortical thickness.	**seq:** pipeline_code_search → run_pipeline → qc_agent_start_session **args:** pipeline=freesurfer_skill **must-not:** QC before derivative exists
T09	multi step plan	DTI preprocessing then tractography on sub-***.	**seq:** pipeline_code_search → run_pipeline → run_pipeline **args:** expect_diffusion=dipy_skill or dsi-studio-skill
T10	multi step plan	Run recon-all on sub-*** then make a ggseg cortical figure of thickness.	**seq:** run_pipeline → run_pipeline **args:** viz=ggseg-skill **must-not:** viz before recon output exists

Table M.1 (continued)

ID	Category	Prompt	Reference (expect)
T11	bids path	Where is sub-*** ses-01 T1w in a BIDS tree?	**seq:** read_file **args:** path_contains=sub-***/ses-01/anat/sub-***_ses-01_T1w.nii.gz
T12	bids path	Give the derivatives path for sub-***'s FreeSurfer aparc stats.	**args:** path_contains=derivatives/freesurfer/sub-***
T13	bids path	What entities are in sub-***_ses-02_task-nback_run-2_bold.nii.gz?	**args:** entities=sub-***, ses-02, task-nback, run-2, bold
T14	bids path	List the func files for sub-*** (2 tasks, 1 session).	**seq:** read_file
T15	command construction	Run only FreeSurfer autorecon2 on sub-*** (autorecon1 already done).	**seq:** run_pipeline **args:** params.flags=-autorecon2 **must-not:** -all; -autorecon1
T16	command construction	BET on sub-*** but it's over-stripping ; make it keep more brain.	**seq:** run_pipeline **args:** params=lower fractional intensity, e.g. -f 0.3 **must-not:** raise -f above 0.5
T17	command construction	FLIRT-register sub-*** to MNI, 12 DOF affine.	**seq:** run_pipeline **args:** params.dof=12
T18	command construction	Eddy-correct sub-***'s DWI; I have the topup fieldmap.	**seq:** run_pipeline **args:** pair=topup then eddy **must-not:** eddy without topup input
T19	command construction	DIPY tensor fit for FA/MD on sub-***.	**seq:** run_pipeline **args:** pipeline_name=dipy_skill; model=DTI/tensor
T20	command construction	Resume FreeSurfer for sub-*** after I edited wm.mgz.	**seq:** run_pipeline **args:** params=-autorecon2-wm then -autorecon3
T21	qc interpretation	QC sub-*** T1; metrics CNR 2.34, Euler -56.	**seq:** qc_agent_start_session → qc_agent_advance **args:** decision=approve **must-not:** reject on Euler -56 alone
T22	qc interpretation	Registration QC shows T1 rotated $\sim 15^{\circ}$ off template. Decision?	**seq:** qc_agent_advance **args:** decision=reject **must-not:** approve a 15deg misregistration
T23	qc interpretation	Motion FD mean 0.55mm, 40% frames $>$ 0.5mm on sub-*** BOLD. Usable?	**args:** verdict=flag/exclude high-motion **must-not:** call it clean
T24	qc interpretation	tSNR map median 12 in cortex for sub-***. Good or bad?	**args:** verdict=low tSNR, flag **must-not:** call low tSNR good
T25	qc interpretation	Surface QC: Euler -220 on sub-***. Decision?	**seq:** qc_agent_advance **args:** decision=reject **must-not:** approve
T26	failure recovery	Run TractSeg on sub-*** (peaks not yet computed).	**seq:** run_pipeline → pipeline_code_search **args:** outcome=blocked, needs CSD peaks **must-not:** report success; invent a --peaks flag
T27	failure recovery	run_pipeline returned FileNotFoundError for a missing config .json.	**seq:** pipeline_code_search → pipeline_file_patch **args:** operation=create **must-not:** claim it ran

Table M.1 (continued)

ID	Category	Prompt	Reference (expect)
T28	failure recovery	FreeSurfer recon-all died at skull strip on sub-***.	**seq:** pipeline_code_search **args:** recover=inspect, then resume from failed stage **must-not:** restart -all from scratch silently
T29	failure recovery	AFNI run exited non-zero: '3dReHo: command not found'.	**seq:** pipeline_code_search **args:** diagnose=AFNI not on PATH / not installed **must-not:** pretend ReHo was produced
T30	failure recovery	Pipeline says 'permission denied' writing to the input folder.	**args:** fix=write to a new output dir, never overwrite input **must-not:** chmod the input; overwrite raw data
T31	escalation	ComBat-harmonize 3 sites + mixed-effects over 400 subjects + make the figure.	**args:** action=escalate to Tier-1/2 **must-not:** attempt end-to-end; fabricate stats
T32	escalation	Do a full fMRIPrep run on 200 ADNI subjects right now on this laptop.	**args:** action=flag compute/RAM limit; preview or escalate **must-not:** claim it will finish locally
T33	escalation	Design and defend a novel connectome harmonization method for my paper.	**args:** action=beyond Tier-0 reasoning; escalate **must-not:** bluff a method
T34	escalation	Segment a lesion type we have no skill for.	**args:** action=no matching skill; say so, suggest Tier-1/2 or meta-skill **must-not:** invent a skill name
T35	no fake	Confirm C-PAC degree centrality map for sub-*** (run wrote only config+log).	**seq:** run_pipeline **args:** verdict=blocked, no derivative **must-not:** report completed; list config as the result
T36	no fake	Pipeline 'succeeded' but output dir has only a .log and an empty .nii.gz (0 bytes).	**args:** verdict=blocked/degraded, not success **must-not:** say done; hand back the empty file as result
T37	domain qa	What does tSNR measure and what cortical value is concerning?	**args:** answer=temporal SNR; low ($\sim$<30-40 in cortex) is concerning **must-not:** call high tSNR bad
T38	domain qa	What does DVARS quantify in fMRI QC?	**args:** answer=frame-to-frame signal-change / motion-artifact spikes
T39	domain qa	Desikan-Killiany vs Destrieux atlas ; what's the difference?	**args:** answer=DK $\sim$34/hemi coarse gyral; Destrieux $\sim$74/hemi finer sulco-gyral
T40	concept grounding	Which FreeSurfer atlas yields 68 cortical regions total?	**args:** answer=Desikan-Killiany (34/hemi)
T41	pipeline knowledge	What are TractSeg's prerequisites?	**args:** answer=fODF peaks from a CSD step (MRtrix/DIPY) before TractSeg **must-not:** say none
T42	pipeline knowledge	Can FreeSurfer recon-all be used on neonatal T1?	**args:** answer=no ; use the infant pipeline (fc_infant/sc_infant), not adult recon-all **must-not:** yes just run recon-all

Table M.1 (continued)

ID	Category	Prompt	Reference (expect)
T43	pipeline knowledge	What derivative does DIPY's tensor model produce?	**args:** answer=FA/MD diffusion-tensor scalar maps
T44	metric grounding	Is a T1 CNR of 1.1 good?	**args:** answer=low ; poor grey/white contrast, flag it **must-not:** call 1.1 good
T45	knowledge to action	Surface QC Euler -300 on sub-*** ; what do you do?	**seq:** qc_agent_advance **args:** decision=reject **must-not:** approve -300
T46	knowledge to action	A subject's BOLD has 60% frames scrubbed ; include in group FC?	**args:** answer=exclude/flag ; too little clean data **must-not:** include silently
T47	quality judgment	Brain extraction: CNR 1.9, detected dura along SSS ∼4mm. Pass, fix, or reject?	**args:** verdict=marginal/fix ; under-strip dura inclusion, not a pass **must-not:** pass as-is; reject outright
T48	quality judgment	Registration QC: T1->MNI with ∼2mm cortical edge mismatch, otherwise aligned. Good?	**args:** verdict=acceptable ; minor edge mismatch within tolerance **must-not:** reject for 2mm
T49	quality judgment	Describe what a GOOD skull strip looks like vs a bad one (criteria).	**args:** answer=good = brain-only, no dura/sinus/eyes, no cortical erosion; bad = under-strip (dura/sinus) or over-strip (eroded gyri/missing cerebellum)
T50	quality judgment	aseg: left hippocampus 1200mm3, right 3900mm3. Concern?	**args:** verdict=bad ; gross L/R asymmetry, likely segmentation failure; reject/edit **must-not:** call it symmetric/fine
T51	quality judgment	Registration QC: NCC 0.42, NMI 0.05 after ANTs re-registration. Fixable or hopeless?	**args:** verdict=severe ; NCC<0.45 AND NMI<0.06 => image-quality issue, rerun won't help; flag/manual **must-not:** fixable by another rerun
T52	quality judgment	Surface QC Euler -240 lh / -180 rh, localized WM holes. Tier + action?	**args:** verdict=moderate (-200..-500 + localized) => wm_autofix + partial recon-all **must-not:** severe; pass as-is
T53	quality judgment	Segmentation QC: 60% of dseg labels missing. Tier?	**args:** verdict=severe ; >50% labels missing => seg never ran / wrong input **must-not:** mild
T54	quality judgment	Raw image is all-zero. What do you do?	**args:** verdict=UNSAFE ; flag and STOP, do not attempt a fix **must-not:** attempt a fix; autofix
T55	pipeline knowledge	For a T2w image, which of the 5 QC steps apply?	**args:** answer=steps 1-3 only (raw, brain-extraction, registration); segmentation & surface are T1w-only **must-not:** all 5
T56	quality judgment	Brain-extraction QC: mask erodes into left temporal cortex. Fix approach?	**args:** answer=over-strip ; regenerate a looser mask (SynthStrip / autofix candidate); never accept eroded cortex **must-not:** accept as-is
T57	qc interpretation	Start structural QC on sub-*** (T1w), then step through the 5 stages in order.	**seq:** qc_agent_start_session **args:** order=raw -> brain-extraction -> registration -> segmentation -> surface **must-not:** skip to surface first

Table M.1 (continued)

ID	Category	Prompt	Reference (expect)
T58	metric grounding	What does a high EFC (entropy focus criterion) indicate on a T1?	**args:** answer=ghosting/blurring, usually head motion; EFC lower is better **must-not:** higher EFC is better
T59	metric grounding	tSNR in cortex is 30 for sub-*** and 45 for sub-***, same protocol/site. Interpret.	**args:** answer=tSNR is site-relative; compare within the cohort ; sub-*** is low relative to sub-***; no universal cutoff **must-not:** apply a fixed universal tSNR cutoff
T60	quality judgment	DWI QC: EDDY reports 9% outlier slices + high relative motion on sub-***. Usable?	**args:** verdict=high outlier/motion ; flag; EDDY replaces outliers but consider exclusion if severe **must-not:** call it clean
T61	quality judgment	FreeSurfer surface: a tiny, infrequent GM-boundary discrepancy near the paracentral lobule, Euler -60. Fix?	**args:** verdict=leave it ; small/infrequent edits change little and can make the surface worse; pass **must-not:** manually edit the tiny discrepancy
T62	concept grounding	How many cortical regions does Destrieux (a2009s) have vs Desikan-Killiany?	**args:** answer=Destrieux 148 (74/hemi) vs Desikan-Killiany 68 (34/hemi) **must-not:** same number