
Seed2.1 Model Card: Agentic Intelligence for Productivity

Bytedance Seed

1 Introduction

As ByteDance’s flagship model family, Seed has been developed with a long-standing commitment to understanding users’ real needs, supporting a wide range of work and life scenarios, and empowering users to think, create, and work with greater confidence.

Through the previous cycle of user feedback and model iteration, the Seed team has developed a deeper understanding of what truly matters for LLM-based agents as productivity tools. Their core value does not lie in scaling interactions alone, but in creating genuine productivity value for each individual user. A capable agent should not merely respond to prompts; it should help users navigate complex decisions, complete demanding tasks, and produce reliable outcomes in real workflows.

Seed2.1 marks a critical turning point for the Seed model family. For the first time, complex needs from daily life, professional productivity, and frontier exploration have been placed at the center of our model development priorities, ahead of traffic-driven general usage alone. This shift has led to substantial improvements in both intelligence and reliability, enabling Seed2.1 to better serve as a trusted partner across real-world scenarios.

In this release, we introduce two Seed2.1 model variants: **Seed2.1 Turbo** and **Seed2.1 Pro**. Seed2.1 Turbo is designed for efficient, high-throughput product scenarios, while Seed2.1 Pro targets stronger reasoning, agentic execution, and complex productivity workflows. Together, they form a practical model family for both everyday user assistance and demanding professional tasks.

As model capabilities continue to evolve, we observe two important changes in how they should be evaluated. First, model evaluation is moving beyond static benchmarks and becoming increasingly tied to real user experience. Second, model performance is becoming more deeply integrated with the harness, tools, and product environment in which the model operates. In other words, the value of an agent should be measured not only by isolated capability scores, but also by whether it can perform reliably in the actual product workflows that users depend on.

To address these changes, the Seed team adopted a product-driven development and evaluation methodology for Seed2.1:

1. The Seed team continuously optimized Seed2.1 Turbo and Seed2.1 Pro through dogfooding with internal users, external invited testers, and crowd-testing developers.
2. The Seed team constructed product-driven evaluation targets around real user experience:
 - (a) In coding scenarios, we prioritized direct user experience in products such as Claude Code and Trae, focusing on whether the models can solve realistic engineering tasks rather than only perform well on standalone coding benchmarks.
 - (b) In general-agent scenarios, we continuously expanded benchmarks from real bad cases returned by users. In parallel, we conducted large-scale interviews with enterprise users and domain experts to identify unmet needs and convert them into evaluation tasks.

- The Seed team deeply used Seed models themselves to participate in model iteration, including evaluation construction, failure analysis, data cleaning, data synthesis, training iteration, and infrastructure optimization. Although this capability is still at an early stage, it reveals the emerging potential of self-evolving model development: stronger models can help build better evaluations, produce higher-quality data, improve training workflows, and accelerate both model and organizational evolution.

2 General Agent

This section evaluates Seed2.1’s broader agentic capability beyond coding, focusing on whether the model can plan, use tools, process files, maintain task state, and produce final deliverables in realistic professional workflows. The evaluation combines public agent benchmarks with Seed-developed evaluations, covering workspace-level file reasoning, slide generation, enterprise document reasoning, economically valuable deliverables, and market-validated startup workflows.

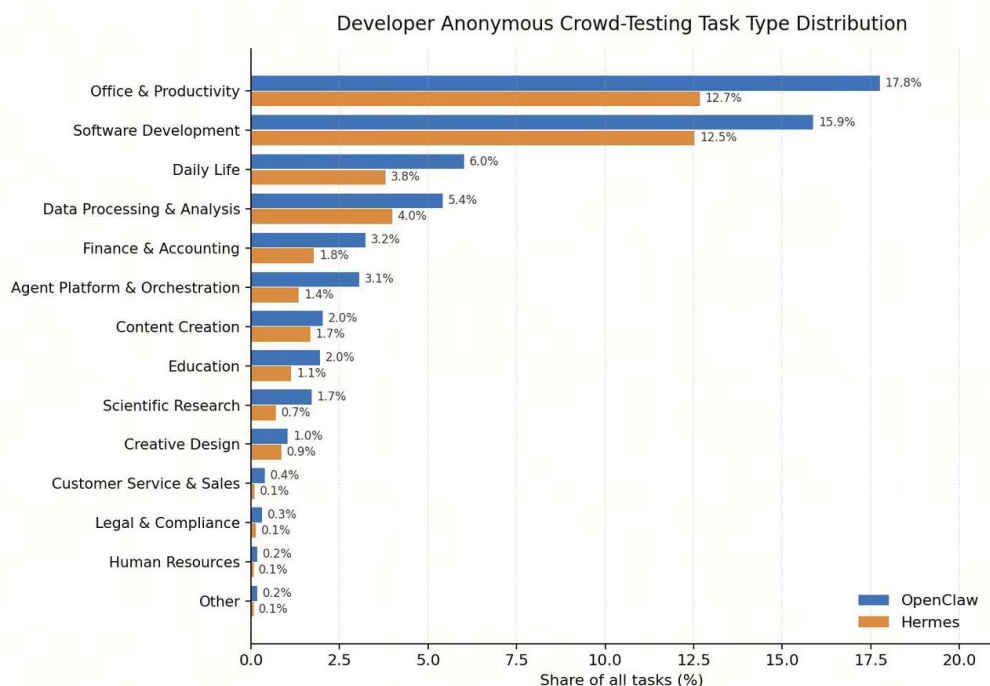


Figure 1 Distribution of Anonymous Crowd Evaluation Tasks.

2.1 Crowdsourced Agent Evaluation

During the iterative development of the Seed2.1 model series, we conducted multiple rounds of large-scale anonymous crowdsourced evaluations with Agent users. Developer feedback collected from these evaluations was continuously incorporated into subsequent model improvements. As a result, Seed2.1 achieved substantial gains in General Agent capabilities, as shown in Figure 1.

Based on anonymous developer feedback, Seed2.1 demonstrated a preference ranking superior to Claude Opus 4.6. In particular, across a wide range of real-world tasks, including software development, code security, and hardware-oriented source retrieval, Seed2.1 consistently showed strong capabilities in understanding complex instructions, accurately incorporating user-provided materials, and delivering end-to-end outputs that are executable, practical, and production-ready, as shown in Figures 2, 3, and 4.

- **Stock-strategy backtesting web app.** Figure 2 presents a software-development case where the model was asked to build a Tonghuashun-style stock-strategy backtesting web app. Seed2.1 planned a full-stack implementation with a FastAPI backend, a Python backtest engine, and an ECharts-based frontend.

After multiple public market-data sources were unavailable or incomplete, it reconstructed an offline-safe historical dataset, implemented multiple strategies including moving-average, MACD, KDJ, RSI, and BOLL variants, and debugged several non-trivial issues such as uninitialized rolling-window indicators, incorrect snapshot statistics, marker-coordinate errors, and pandas truth-value bugs. The final deliverable was a running backtesting workspace with K-line charts, MA overlays, volume and MACD sub-charts, equity curves, strategy controls, trade logs, and WebSocket-based bar-by-bar replay.

- **Android memory-leak refactoring.** Figure 3 shows a code-security and refactoring case where the

MODEL CARD · CASE STUDY

Building a stock-strategy backtesting web app

Software Development · agent harness: Hermes

Task Overview

User: Build a stock-strategy backtesting web app with a Tonghuashun-style trading UI. It must load a sample stock's full price history since its IPO, run a user-given trading strategy, and play back the entire backtest process — not just print a final verdict. Use one stock as the example.

Assistant · end-to-end agent workflow reconstructed from the execution trace

[Stage 1] Planning & Environment Probing

```
todo(plan = "Full-stack app: FastAPI backend + Python backtest engine + ECharts front-end; real data first")
terminal("python3 --version && pip3 list | grep -E \"fastapi|akshare|pandas|numpy\"")
terminal("pip3 install akshare -q")
» fastapi / numpy / pandas already present; akshare install times out under the restricted-network sandbox.
Reasoning: plans the whole stack up front and decides the deliverable must be a running app, not a stub.
```

[Stage 2] Real-Data Acquisition — try every real source first

```
terminal("curl push2his.eastmoney.com/api/qt/stock/kline/get?secid=1.600519&kit=101 ...")
terminal("curl web.ifzq.gtimg.cn/appstock/app/fqkline/get?param=sh600519,day,2001-08-27,...")
terminal("curl quotes.money.163.com/service/chddata.html?code=0600519&start=20010827...")
execute_code("curl money.finance.sina.com.cn/.../CN_MarketData.getKLineData?symbol=sh600519&datalen=6000")
» EastMoney / Tencent / 163 return empty or are blocked; Sina works but truncates to ~461 of the ~6000 trading days needed.
Reasoning: "simulated data has no reference value for a backtest" — exhausts 5 public market-data APIs before giving up on a live feed.
```

[Stage 3] Honest Data Reconstruction & Backtest Engine

```
write_file("generate_data.py") # rebuild 600519 from IPO 2001-08 (~31 yuan) to 2021 peak (~2600) by real milestones
terminal("python3 generate_data.py") -> 6,210 trading days generated, price path matches the real history
write_file("backtest.py") # 8 strategies: dual-MA, MA60-trend, MA-cross, MACD, KDJ, RSI, BOLL, custom
terminal("python3 backtest.py")
» Win-rate is only -22%: buy-and-hold is +3742% yet the strategy shows -98% — the logic must be wrong, not the market.
Reasoning: refuses to fake numbers; reconstructs a realistic, offline-safe dataset, then treats the absurd result as a bug to chase.
```

[Stage 4] Root-cause Debugging, UI & Full Verification

```
execute_code("inspect df[[ma5, ma20]].head(25)") -> earliest bars have MA=0 (rolling window not warmed up)
patch("backtest.py") # guard NaN / zero indicators -> kill the false golden-cross signals
patch("generate_data.py") # lower daily volatility 1.8% -> 0.8% so MAs stop crossing on noise
write_file("server.py") # FastAPI HTTP + WebSocket bar-by-bar replay
write_file("templates/index.html") # Tonghuashun dark UI, red-up/green-down, ECharts K-line + equity curve
patch x3 # fix empty engine.snapshots (stats), date-based buy/sell marker coords, RSI pandas Series truth-value bug
terminal("for s in ma_cross dual_ma macd kdj rsi boll ...; curl /api/backtest?strategy=$s")
» All 8 strategies run clean. Trend strategies (dual-MA +25%/yr, Sharpe 2.78) beat the oscillators — the expected pattern on a long-uptrend stock, which confirms the engine is actually correct, not just running.
```

[Generated Content] — the actual running product



Full Tonghuashun-style backtest workspace: K-line + MA overlays, volume / MACD sub-charts, equity curve, strategy panel, live trade log and bar-by-bar replay with buy/sell markers.

Result A working FastAPI + ECharts app over the stock's full history (6,210 trading days), 8 strategies, and a WebSocket bar-by-bar replay showing buy/sell markers, an equity curve and a trade log.
Developer acceptance: rated above expectations for a highly faithful UI and a complete feature set; got there through several rounds of self-driven iteration and refinement.

Figure 2 The case of Seed2.1 with Software Development task.

model was asked to audit and harden an Android Activity implementation with severe memory-leak issues. Seed2.1 first grounded its diagnosis in the uploaded `OldDownloadActivity.java`, identified two strong-reference leak paths from the non-static `Handler` and anonymous background `Runnable`, and then delivered a complete refactored `SecureDownloadActivity.java`. The fix introduced a static `Handler` with `WeakReference`, an interrupt-aware static background runnable, liveness checks before UI updates, precise exception logging, and explicit cleanup in `onDestroy`. This case demonstrates Seed2.1's ability to turn code-audit findings into a complete, lifecycle-aware implementation rather than only providing high-level suggestions.

- **Hardware Q&A with official source retrieval.** Figure 4 shows a hardware source-retrieval case on the nRF54L15 DK, where the user required exact GPIO pin assignments, solder-bridge operations, and resistor reference-designator changes. When the official online manual was blocked by Cloudflare and secondary sources were inconsistent, Seed2.1 avoided guessing and instead downloaded the official PCA10156 hardware files. It parsed the schematic, BOM, and assembly drawing to identify that P1.00/P1.01 are connected by default to the 32.768 kHz crystal through XL1/XL2, that freeing them requires cutting SB3/SB4 and shorting SB5/SB6, and that using P1.02/P1.03 as GPIO requires moving R21 to R33 and R22 to R34. This case highlights Seed2.1's ability to recover from web-access failures, trace claims to official engineering artifacts, and produce directly actionable hardware instructions.

2.2 High-Economic-Value Tasks

2.2.1 Benchmark Overview

The general-agent evaluation suite combines public benchmarks and Seed-developed evaluations to assess complementary aspects of professional productivity. Public benchmarks cover workspace-level file dependency reasoning, slide generation, expert-level professional tasks, enterprise document reasoning, economically valuable deliverables, and long-horizon cross-application workflows. Specifically, Workspace Bench evaluates realistic workspace tasks with large-scale heterogeneous file dependencies [51]; PresentBench evaluates

MODEL CARD · CASE STUDY

Auditing & hardening a memory-leaking Android Activity

Code Security / Refactoring · agent harness: Hermes

Task Overview

User: Act as a senior Android architect. The uploaded OldDownloadActivity.java crashes when the user quickly closes the page and leaks the Activity (it can't be GC'd). (1) Pinpoint the 2 strong-reference leak paths from the Handler / anonymous inner classes; (2) refactor with a static inner class + WeakReference (or a lifecycle-aware component) so no background thread keeps a strong reference to UI or Context after onDestroy; (3) add main-thread + isFinishing()/isDestroyed() liveness checks before touching UI; (4) never swallow exceptions — catch precisely and Log.e the full stack. Deliver an audit report first, then the complete refactored SecureDownloadActivity.java.

Assistant · end-to-end agent workflow reconstructed from the execution trace

[Stage 1] Locate & Read the Faulty Source

```
search_files(path="/home/claude-user", pattern="OldDownloadActivity.java", target="files")
read_file(path="/home/claude-user/OldDownloadActivity.java") -> 48 lines, com.example.audit
```

» **Reads the real file before judging — no fixing blind. Identifies a non-static Handler, an anonymous Thread/Runnable, a direct UI write with no liveness check, and an empty catch block.**

Reasoning: "I must read the uploaded file first to audit it" — grounds the whole review in the actual code, not assumptions.

[Stage 2] Code-Audit Report — leak paths & crash points

» **Leak #1 (Handler): non-static anonymous Handler implicitly holds the Activity. Chain = MessageQueue → Message → Handler → Activity → whole View tree; pending messages block GC.**

» **Leak #2 (Thread): anonymous Runnable holds the Activity too; the ~50 s download loop keeps it alive even after finish().**

crash points enumerated in a table:

L18 handleMessage -> setText with no liveness check -> NPE / BadTokenException

L43 empty catch {} swallows InterruptedException -> undebuggable

--- onDestroy never removes messages / stops thread -> IllegalStateException

Reasoning: names the exact leak chains and maps each crash to a concrete exception type before writing a single line of the fix.

[Stage 3] Refactor — write SecureDownloadActivity.java

```
write_file("SecureDownloadActivity.java") # 215 lines, 6 numbered fix points
static class DownloadHandler -> WeakReference<Activity>, drops msg if GC'd
static class DownloadRunnable -> holds only a WeakReference<Handler>, no Context
handleMessage(): isFinishing() || isDestroyed() guard before updateProgress()
run(): per-loop Thread.isInterrupted() check -> safe early exit (no 50 s spin)
catch (InterruptedException e){ Thread.currentThread().interrupt(); Log.e(TAG,...e);} # never swallow
onDestroy(): removeCallbacksAndMessages(null) + interrupt() + null-out View refs
```

Reasoning: converts every audited defect into a labelled fix (「修复点1..6」), preserving the original behaviour while cutting all strong-ref chains.

[Stage 4] Self-Verification & Hand-off

verification summary mapped fix-by-fix against the original defects:

Handler leak -> static + WeakReference, get() before every use

Thread leak -> static Runnable, no outer-class access, interrupt-aware

UI safety -> triple guard: isFinishing()/isDestroyed() + null-check + main Looper

exceptions -> InterruptedException restores interrupt flag; all paths Log.e(TAG,msg,e)

» **Confirms the result compiles against the Android SDK with no obvious syntax errors and can directly replace the original file — delivers the audit report and the full 215-line class, not a sketch.**

[Generated Content] — the actual refactored deliverable

SecureDownloadActivity.java · fix #1 — static Handler + WeakReference

```
private static class DownloadHandler extends Handler {
    private final WeakReference<SecureDownloadActivity> activityRef;
    DownloadHandler(Looper looper, SecureDownloadActivity a){ super(looper); activityRef = new WeakReference<>(a); }
    @Override public void handleMessage(Message msg) {
        SecureDownloadActivity act = activityRef.get();
        if (act == null) { Log.w(TAG, "Activity GC'd, drop msg"); return; }
        if (act.isFinishing() || act.isDestroyed()) { Log.w(TAG, "destroying, skip UI"); return; }
        if (msg.what == MSG_PROGRESS) act.updateProgress(msg.arg1);
    }
}
```

fix #4/#5 — interrupt-aware loop, never swallow the exception

```
for (int i = 1; i <= 100; i++) {
    if (Thread.currentThread().isInterrupted()) { Log.i(TAG, "interrupted, stop"); return; }
    Thread.sleep(DOWNLOAD_STEP_INTERVAL_MS);
    ... handler.sendMessage(handler.obtainMessage(MSG_PROGRESS, i, 0));
} catch (InterruptedException e) {
    Thread.currentThread().interrupt(); // restore the flag
    Log.e(TAG, "download interrupted during sleep", e); // full stack, no empty catch
}
```

fix #6 — onDestroy actively releases everything

```
@Override protected void onDestroy() {
    if (mHandler != null) mHandler.removeCallbacksAndMessages(null); // cut Message->Handler->Activity
    if (mDownloadThread != null && mDownloadThread.isAlive()) mDownloadThread.interrupt();
    statusText = null; mHandler = null; mDownloadThread = null; // help GC
    super.onDestroy();
}
```

Independent developer acceptance

In a separate, on-device review the developer compared the refactored code from four anonymized models and ranked this run first — rated the best implementation and recommended for adoption.

Relative standing: this run led the group; the next output was judged merely acceptable, while the weaker ones were flagged for a possible residual thread leak or failed to compile at all.

Why it won: it was the only output credited with full lifecycle awareness — background threads are interrupted safely when the Activity is destroyed.

Result A complete, compilable SecureDownloadActivity.java that closes both strong-reference leaks (static Handler + WeakReference, static Runnable), guards every UI update with liveness checks, responds to interrupts, and never swallows an exception.

Acceptance: ranked first among four models in an independent developer review — rated the best implementation and recommended for adoption.

Figure 3 The case of Seed2.1 with Code Security task.

automated slide generation using fine-grained, instance-specific rubrics [8]; OneMillion Bench evaluates expert-level agent tasks across professional domains such as law, finance, industry, healthcare, and natural science [74]; OfficeQA-Pro focuses on grounded multi-document enterprise reasoning over a large heterogeneous corpus [41]; GDPval evaluates model performance on economically valuable real-world tasks across occupational domains [43]; Finance Agent v1.1 evaluates LLM agents’ end-to-end performance on real-world financial analyst tasks across 9 categories — from basic information retrieval to complex financial modeling [3]; and APEX Agents evaluates long-horizon professional service tasks created by investment banking analysts, management consultants, and corporate lawyers [57]. Agent Startup Bench is a Seed-developed benchmark that focuses on market-validated AI workflows derived from AI-native startup products and real enterprise-user needs. Together, these evaluations test whether an agent can move beyond answering questions and complete realistic work products under practical workflow constraints.

Workspace Bench Workspace Bench evaluates whether agents can complete realistic workspace tasks involving large-scale heterogeneous file dependencies [51, Table 4]. In our evaluation, we report results under the 100-task OpenClaw setting using the metric order Total / Pass@30 / Pass@50 / Pass@70 / Pass@90 / Pass@100. The paper-reported result for 3.1-Pro is 31.6, with Pass@30 / Pass@50 / Pass@70 / Pass@90 / Pass@100 = 45.0 / 31.0 / 20.0 / 11.0 / 8.0. Unlike benchmarks that provide a small set of pre-selected files, Workspace Bench requires agents to operate within realistic workspaces, identify relevant files, reason over explicit and implicit dependencies, and produce outputs that satisfy task-specific rubrics. This makes it particularly relevant

for professional productivity scenarios where users expect agents to search across messy folders, connect information from multiple documents, update or create files, and maintain consistency between evidence and final deliverables.

Agent Startup Bench Agent Startup Bench is a Seed-developed benchmark for evaluating whether agents can complete real-world workflows derived from AI-native startup products and their users. Its core motivation is demand-driven evaluation: tasks should reflect workflows that users already choose to delegate to AI systems, rather than only researcher-designed capability tests. The benchmark is constructed through a survey- and interview-driven pipeline. We first survey AI-native startups with evidence of market validation,

MODEL CARD · CASE STUDY

Tracing official hardware files to answer an nRF54L15 DK GPIO question

Hardware Q&A / Source Retrieval · agent harness: OpenClaw

Task Overview

User: I'm building a device on the nRF54L15 DK and need pins P1.00 and P1.01 as GPIO to connect an external module, but they appear to be occupied by default. (1) Per the manual, which on-board component are these two pins assigned to? (2) To fully release P1.00 / P1.01 as plain GPIO, what physical board modifications are required (give the exact Solder-bridge numbers and the action)? (3) I'd also like to use P1.02 / P1.03 as GPIO — which resistors must be moved? Give the precise reference-designator changes.

Assistant · workflow reconstructed from the execution trace (137 tool calls: Bash 75 / Process 29 / WebSearch 19 / WebFetch 7 / Read 7)

[Stage 1] Search the manual online — blocked by Cloudflare

WebSearch("nRF54L15 DK P1.00 P1.01 default pin assignment solder bridge")
WebFetch("docs.nordicsemi.com/.../hw_cg_pin_conf.html") -> blocked by Cloudflare

» **The official online manual is gated by a human check and 2nd-hand blogs disagree — not enough to answer down to a designator.**

Reasoning: won't guess from scattered blogs — if the official page won't open, go fetch the official raw hardware files instead.

[Stage 2] Download & extract the official PCA10156 hardware files

curl / wget / python download the nRF54L15 DK hardware-files .zip from Azure blob (throttled, retried)
unzip pca10156.zip -> schematic SchDoc + Production files (BOM / Assembly / Pick-&-Place)

» **The download was rate-limited and dropped repeatedly; it kept switching tools and retrying until it pulled the schematic and production files out of a partial archive.**

Reasoning: with the official source secured, it would rather grind through a binary schematic than invent a bridge number.

[Stage 3] Parse schematic + BOM + render & vision-read the Assembly Drawing

olefile parses the Altium SchDoc (OLE compound doc) -> nets:
P1.00/XL1, P1.01/XL2 -> 32.768kHz crystal XC1 ; P1.02/NFC1, P1.03/NFC2 -> NFC antenna J7
xlrd reads the BOM -> R21 / R22 = Fitted ; R33 / R34 = Not Fitted

pypdf/fitz render Assembly Drawing PDF -> PNG, Read vision-reads the bottom-silkscreen config table (SB3-SB6)

Reasoning: when binary parsing got slow it pivoted to vision-reading the assembly drawing, cross-checking schematic nets / BOM fit-state / silkscreen table.

[Stage 4] Synthesize an actionable three-part answer

» **Maps the bridge ops (cut SB3/SB4, short SB5/SB6) and resistor moves (R21->R33, R22->R34) one-to-one onto the three questions.**

» **Adds firmware notes: with the crystal cut, LFCLK must switch to internal RC/synth; freeing NFC pins needs nfct-pins-as-gpios.**

[Generated Content] — the delivered three-part answer

Q1 · On-board component on P1.00 / P1.01 by default

These are the nRF54L15 **XL1 / XL2** pins, connected by default via shorted bridges to the **32.768 kHz crystal (X1/X2, schematic XC1)**.

Q2 · Solder-bridge ops to free P1.00 / P1.01

Cut **SB3, SB4** (remove solder — disconnect from crystal); short **SB5, SB6** (bridge with solder — connect to the GPIO header). This disconnects the crystal; if the app doesn't use an external LFCLK, switch LFCLK to internal RC/synth in firmware.

Q3 · 00hm resistors to move for P1.02 / P1.03

Move **R21 to the R33 pad** (P1.02/NFC1 -> GPIO P1.02); move **R22 to the R34 pad** (P1.03/NFC2 -> GPIO P1.03). R21/R22 are Fitted (NFC-antenna path), R33/R34 are Not Fitted (GPIO path); all 00hm / 0402.

firmware-side addition · device-tree overlay

```
&uicr { nfc-pins-as-gpios; };  
After freeing the NFC pins this must be declared, otherwise P1.02 / P1.03 still init as NFC.
```

Independent developer acceptance

In a separate developer review this run was judged the best answer in the anonymized group and recommended for adoption.

Relative standing: it pinned every bridge and resistor designator and tied each to the official files; the others stayed at the directional level or got a designator wrong.

Why it won: it was the only output that, once the official manual was blocked, still downloaded and parsed the raw hardware files so every claim is traceable — rather than answering from memory.

Result Facing a hardware question that demands exact bridge/resistor designators, and with the official online manual blocked by Cloudflare, it downloaded and parsed the official schematic, BOM and assembly drawing to deliver a complete, directly actionable three-part answer in which every claim is traceable to an official source — judged the best in an independent developer review and recommended for adoption.

Figure 4 The case of Seed2.1 with Hardware task.

such as funding, paying users, or large-scale adoption, then interview deep users to understand usage contexts, input materials, expected deliverables, and practical success criteria. Domain experts then convert validated scenarios into benchmark tasks with task descriptions, attachments, output requirements, and rubric-based evaluation criteria, followed by quality control for realism, answerability, evaluability, and discriminative difficulty.

2.2.2 Results

As shown in Table 1, Seed2.1 demonstrates strong general-agent productivity capability across workspace reasoning, professional deliverable generation, enterprise document understanding, and long-horizon service tasks. On Workspace Bench, Seed2.1-Turbo reaches 54.7 and Seed2.1-Pro reaches 53.0, approaching Claude-4.7-Opus at 55.1 and substantially outperforming Gemini-3.1-Pro at 32.8. On Agent Startup Bench, Seed2.1-Pro achieves the best listed result at 68.8, surpassing GPT-5.5 at 68.1, Claude-4.7-Opus at 62.3, and Gemini-3.1-Pro at 45.7. On Agents' Last Exam, Seed2.1-Pro reaches 19.5 / 41.4, outperforming Claude-4.7-Opus and Gemini-3.1-Pro while approaching GPT-5.5, further indicating strong performance on difficult long-horizon agent tasks.

Seed2.1-Pro is also competitive on professional deliverable and enterprise reasoning tasks. It reaches 68.8 on OneMillion Bench, close to GPT-5.5 at 69.6 and clearly above Gemini-3.1-Pro at 60.2. On OfficeQA-Pro, Seed2.1-Pro reaches 70.9, substantially outperforming GPT-5.5 at 62.9 and approaching Gemini-3.1-Pro

Benchmark	GPT-5.5	Claude-4.7 Opus	Gemini-3.1 Pro	Seed2.1 Turbo	Seed2.1 Pro
	58.7	55.1	32.8	54.7	53.0
Workspace Bench [51]	81.0 / 67.0 44.0 / 25.0 / 20.0	76.0 / 56.0 40.0 / 20.0 / 18.0	49.0 / 29.0 17.0 / 9.0 / 4.0	74.0 / 59.0 39.0 / 20.0 / 16.0	78.0 / 60.0 38.0 / 25.0 / 20.0
PresentBench [8]	68.9	61.8	52.1	48.3	54.6
Agent Startup Bench	68.1	62.3	45.7	54.0	68.8
Agents’ Last Exam [50]	24.0 / 42.8	18.4 / 40.5	15.8 / 32.0	-	19.5 / 41.4
OneMillion Bench [74]	69.6	73.0	60.2	66.6	68.8
OfficeQA-Pro [41]	62.9	76.5	72.5	62.8	70.9
GDPval [43]	84.9	82.7	67.3	82.7	87.9
Finance Agent v1.1 [3]	65.3	64.4	59.7	56.0	60.7
APEX Agents [57]	35.4	33.9	33.5	29.2	33.8

Table 1 General-agent automatic evaluation. For Workspace Bench, the first number is Total, and the following two lines report Pass@30 / Pass@50 / Pass@70 / Pass@90 / Pass@100 under the 100-task OpenClaw setting. For Agents’ Last Exam, results are reported as Pass@1 / Score. OfficeQA-Pro incorporates some internal implementations and adaptations, so its scores may show some fluctuation compared with the official leaderboard.

at 72.5, while Claude-4.7-Opus remains the strongest listed model. On GDPVal, Seed2.1-Pro achieves the best listed score at 87.9, outperforming GPT-5.5 at 84.9, Claude-4.7-Opus at 82.7, and Gemini-3.1-Pro at 67.3. On APEX Agents, Seed2.1-Pro reaches 33.8, nearly matching Claude-4.7-Opus at 33.9 and exceeding Gemini-3.1-Pro at 33.5. Together with crowdsourced agent-task evaluation, these results show that Seed2.1 has reached a strong production-ready level for general-agent productivity workflows.

2.3 Complex Daily Life Consultation

Complex daily-life consultation evaluates whether Seed2.1 can serve as a reliable advisor in realistic user scenarios. Unlike simple question-answering tasks, daily-life consultation often requires multi-step reasoning, preference understanding, constraint tracking, practical planning, and the ability to provide actionable recommendations. We combine human evaluation with automatic evaluation to capture both subjective helpfulness and scalable benchmark performance. Related public multi-turn evaluation efforts emphasize the difficulty of maintaining instruction consistency, tracking user intent, and adapting to evolving context across realistic extended conversations [14, 72].

2.3.1 Benchmark Overview

Evaluation Scope The automatic evaluation suite covers two complementary aspects of complex daily-life assistance. The first is conversation-oriented daily consultation, where the model must understand realistic user needs, maintain multi-turn context, follow user preferences, and provide practical recommendations. This part is evaluated by Seed-developed benchmarks such as xDailyBench and Doubao Multi-Turn Bench. The second is tool-augmented assistance, where the model must go beyond conversation and complete user-facing tasks through tool use, web interaction, long-horizon execution, and reusable skills. This part is evaluated by public agent benchmarks such as MCP-Atlas [2], Toolathlon [31], and SkillsBench [32], together with SeedClawBench, a Seed-developed benchmark for realistic user-facing agent tasks. Together, these benchmarks evaluate whether Seed2.1 can serve not only as a daily-life advisor, but also as a practical assistant capable of completing realistic user tasks.

xDailyBench xDailyBench is a Seed-developed benchmark for evaluating LLMs in real C-end daily scenarios. It is designed to address the gap between conventional academic benchmarks and real non-coding user experience, where tasks are often open-ended, ambiguous, personalized, and difficult to evaluate with a single exact-match metric. The benchmark is grounded in real user needs and covers more than 30 vertical scenarios across daily life, study and research, and white-collar work. It adopts rubric-based multi-dimensional

Benchmark	GPT-5.5	Claude-4.7 Opus	Gemini-3.1- Pro	Seed2.1 Turbo	Seed2.1 Pro
xDailyBench	73.0	69.0	35.2	56.4	61.0
Doubao Multi-Turn Bench	62.5	49.8	52.0	49.0	52.5
MCP-Atlas [2]	81.6	79.1	78.2	80.3	83.8
Toolathlon [31]	55.6	52.8	48.8	49.1	50.6
SeedClawBench	66.4	64.1	57.1	63.8	66.6

Table 2 Automatic evaluation for complex daily-life consultation and tool-augmented assistance in Pass@1.

evaluation to assess task completion, factual correctness, information coverage, practical usefulness, and other quality dimensions, while also supporting fine-grained capability diagnostics for model iteration.

Doubao Multi-Turn Bench Doubao Multi-Turn Bench is a Seed-developed benchmark constructed from large-scale real interactions between online users and Doubao models. Since raw online conversations contain many casual chats, single-turn questions, and simple queries, only a small subset is suitable for evaluating multi-turn dialogue quality and user preference alignment. We therefore design a filtering pipeline to identify conversations where users were dissatisfied with an AI response and where the dialogue itself reflects a realistic user need. The benchmark uses the historical conversation as context, asks the model to answer the turn where the user was dissatisfied, and evaluates the response with rubric-based criteria.

SeedClawBench SeedClawBench is a Seed-developed internal benchmark for evaluating practical agentic assistance in OpenClaw-style user-facing scenarios. The benchmark contains 100 tasks constructed from Ark online usage logs and crowdsourced OpenClaw tasks, covering realistic workflows where an agent must call tools, use skills, process intermediate artifacts, and deliver a final result. Its task categories include data processing and analysis, finance and accounting, daily office work, daily-life assistance, system and feature development, creative design, and customer-service or sales scenarios, with data processing, daily office work, and daily-life tasks accounting for the majority of the benchmark. The benchmark is built from more than 5,000 online Ark tasks and more than 600 crowdsourced tasks: tasks are first automatically tagged by scenario, then manually rewritten according to task volume and difficulty requirements, and finally reviewed with task-specific rubrics and ground truth.

SeedClawBench uses an Agent-as-Judge automatic evaluation protocol. The judge reads the agent transcript, tool trace, final deliverable, and external state snapshots, then calls read-only tools to collect evidence for each rubric item and outputs a binary judgment with an evidence chain. Since OpenClaw tasks may produce heterogeneous artifacts such as PDFs, spreadsheets, images, and videos, the evaluation system provides corresponding evaluation skills so that the judge can inspect task outputs when needed. The evaluation uses separate internal text and multimodal judging pipelines, rather than relying on a single generic judge for all artifact types. In an initial human-machine agreement study over 37 tasks and 262 comparable rubric items, the automatic judge achieved 89.69% agreement with human annotation, measured as the number of matched rubric judgments divided by the total number of rubric items.

2.3.2 Results

As shown in Table 2, Seed2.1 demonstrates strong capability in complex daily-life consultation and tool-augmented assistance. On tool-use benchmarks, Seed2.1-Pro is particularly competitive: it achieves the highest listed score on MCP-Atlas, reaching 83.8 and outperforming GPT-5.5, Claude-4.7-Opus, and Gemini-3.1-Pro. It also attains a competitive score on Toolathlon, benefiting from Seed2.1’s continual, scalable environment learning [17]. Beyond these, it leads on SeedClawBench with 66.6, slightly above GPT-5.5 at 66.4 and Claude-4.7-Opus at 64.1, showing strong performance on realistic OpenClaw-style user-facing agent tasks. Seed2.1-Turbo is already competitive on MCP-Atlas with 80.3, exceeding Claude-4.7-Opus and Gemini-3.1-Pro, while Seed2.1-Pro further strengthens this advantage.

Benchmark	Metric	GPT-5.5	Claude-4.7 Opus	Gemini-3.1 Pro	Seed2.1 Turbo	Seed2.1 Pro
Claw-Eval [Multimodal] [76]	Pass ³	43.0	44.0	27.0	46.0	51.0
Office QA Pro [Multimodal] [41]	Avg Score	69.5	76.5	72.5	71.1	72.2
WildClawBench [16]	Avg Score	65.6	67.0	61.1	62.8	61.7
Image2FloorPlan (Inhouse)	Avg Score	50.7	50.2	55.1	35.9	48.0

Table 3 Multimodal-agent evaluation. Claw-Eval reports Pass³ (all-three-pass); all other benchmarks report average score. Image2Floorplan is a Seed-developed in-house benchmark.

On conversation-oriented daily consultation, Seed2.1-Pro shows clear progress toward frontier-level user assistance. On xDailyBench, it reaches 61.0, substantially outperforming Gemini-3.1-Pro at 35.2 and narrowing the gap to Claude-4.7-Opus. On Doubao Multi-Turn Bench, Seed2.1-Pro reaches 52.5, matching or exceeding the listed Gemini-3.1-Pro and Claude-4.7-Opus results. On SkillsBench, Seed2.1-Pro reaches 60.4, close to Gemini-3.1-Pro and Claude-4.7-Opus, indicating solid skill-use and skill-composition capability in practical workflows. Overall, these results show that Seed2.1 is not only a conversational advisor for daily-life scenarios, but also a capable tool-augmented assistant that can complete realistic user tasks with strong end-to-end performance.

2.4 Multimodal Agents

Real-world agent tasks are increasingly rich-media at both the input and output ends: users provide documents, charts, screenshots, photographs, and videos, while the final deliverable may be a rendered document, poster, layout, or generated video that must be created, stored, and verified as media rather than plain text. Existing harnesses, however, are mostly built around text-and-tool loops and adapt poorly to such settings: inputs are often down-sampled or truncated, and rich-media outputs are difficult to generate and inspect. As a result, perception and media handling, rather than reasoning alone, become the bottleneck. We therefore evaluate Seed2.1 as a multimodal agent: a system that can read, reason over, and act on heterogeneous media within a long-horizon tool-using loop, instead of answering isolated visual questions.

The evaluation is organized around three complementary task types that together cover the multimodal-agent capability surface. **Visual document understanding** tasks require the agent to read and reason over heterogeneous documents, including PDFs, slides, charts, tables, and screenshots, and to locate, integrate, and act on visual evidence across multiple turns. **Open-ended real-world agent work** reflects how users delegate tasks in practice, covering productivity, coding, search, social, creative, and safety scenarios. Each task presents an open-ended goal that the agent must decompose and complete end to end, jointly testing multimodal perception, tool use, and long-horizon execution. **Spatial-structure generation** requires the agent to infer global spatial structure from many local visual views and output it in a precise, machine-checkable form, covering capabilities not captured by document or GUI tasks alone.

2.4.1 The Multimodal Agent Harness

All benchmarks are evaluated under a unified inhouse multimodal agent harness, exercising perception, tool use, code execution, and long-horizon control end to end rather than as isolated visual question answering. Two design choices are central to this setup.

High-resolution perception tools. Instead of relying on low-resolution thumbnails or sparsely selected frames, the harness provides dedicated media-perception tools that render documents and pages at high resolution and sample videos densely, keeping fine print, small chart labels, and crowded layouts legible. Our analysis shows that perception fidelity is often a key factor in multimodal-agent stability: improving the perception path measurably increases multi-sample consistency (Pass^k), even when single-pass solvability remains unchanged.

Multimodal context management. Since high-fidelity media is costly in context, the harness loads multimodal content progressively rather than all at once. The agent reads images and document pages on demand and in

batches across turns; videos are first covered coarsely, with denser frames loaded only when closer inspection is needed. This keeps each turn within a manageable context budget while preserving access to full visual detail.

2.4.2 Results

As shown in Table 3, Seed2.1 is a strong multimodal agent across document understanding, open-ended real-world tasks, and spatial generation. It shows particularly strong reliability on Claw-Eval [MM], where Seed2.1-Pro achieves the best Pass@3 result among all listed models. On Office QA Pro [MM], Seed2.1-Pro performs close to the leading model, while Seed2.1-Turbo also surpasses GPT-5.5, though grounded enterprise-document understanding remains an area for further improvement. On WildClawBench, both Seed2.1 variants are competitive with frontier models, reflecting strong end-to-end performance on open-ended real-world agent tasks. On Image2Floorplan, Seed2.1-Pro improves substantially over Seed2.1-Turbo and approaches the strongest frontier models, suggesting clear progress in spatial-structure generation while leaving room for further gains in complex spatial reconstruction.

Image2Floorplan. Image2Floorplan evaluates spatial-structure generation. It is a Seed-developed in-house benchmark in which each task provides multiple real interior photographs of an apartment, covering its rooms and their connections, and asks the agent to reconstruct the floor plan. The expected output is a structured room-connectivity graph rather than a free-form image, specifying the rooms and their adjacency, door, and passage relationships. Predictions are compared with human-annotated ground-truth graphs using a weighted graph-similarity metric that accounts for room matching, connectivity, and overall topology, and are normalized against a baseline so that the final score reflects genuine reconstruction quality.

2.5 Generalist Computer-Use Agents

In Seed2.1’s exploration of professional productivity, we aim to move beyond agents specialized for isolated tasks or fixed environments toward *Generalist Computer-Use Agents* [50]: systems capable of completing complex workflows across chat, search, browsers, code repositories, files, GUIs, and external tools. Real-world tasks require an agent to understand goals, gather information, perceive screen and interface states, manipulate GUIs, write and execute code, call tools and MCP services, and verify progress over long horizons. From this perspective, the limitations of single-paradigm agents become clear: coding or CLI agents are strong at planning, coding, and tool use but remain brittle on visual interfaces and unexpected screen states, while GUI agents can operate software visually but lack efficient access to code, files, commands, and structured tools. Seed2.1 is therefore designed as a unified digital productivity system that coordinates perception, reasoning, control, action, and execution under a single policy, learning when to see, click, search, code, or call tools, and composing these capabilities into stable, recoverable execution trajectories.

We evaluate this capability on public GUI and computer-use benchmarks together with *CreativeWork*, a Seed-developed benchmark for hybrid GUI-and-MCP productivity tasks. OSWorld evaluates desktop computer control in real operating-system environments [70], MobileWorld evaluates mobile GUI operation [29] with MCP test split excluded due to deployment issues, and GameWorld evaluates multimodal interaction in browser-game environments [42].

Importantly, we run these benchmarks under the Generalist Computer-Use Agent setting rather than a GUI-only one: beyond mouse-and-keyboard control, the agent is also granted the general, code-agent-style actions natural to each environment—for example, on OSWorld it may issue Bash commands, run scripts, and manipulate the file system alongside GUI operations. CreativeWork complements these public benchmarks by targeting real productivity scenarios that demand coordinated GUI and MCP use. It spans three representative environments—Notion, Canva, and Figma—covering document management, visual design, and interface editing, and rather than testing GUI or tool use in isolation, each task asks the agent to interpret an open-ended goal, decompose it, and decide on its own when to act through a tool call and when to fall back to the interface. By construction, the two modalities are tightly interleaved: MCP handles structured operations such as creating pages, duplicating documents, generating assets, and editing object properties, while the GUI carries what tools cannot—reading layouts, navigating property panels, resolving spatial references, inspecting visual results, and confirming the final state on screen.

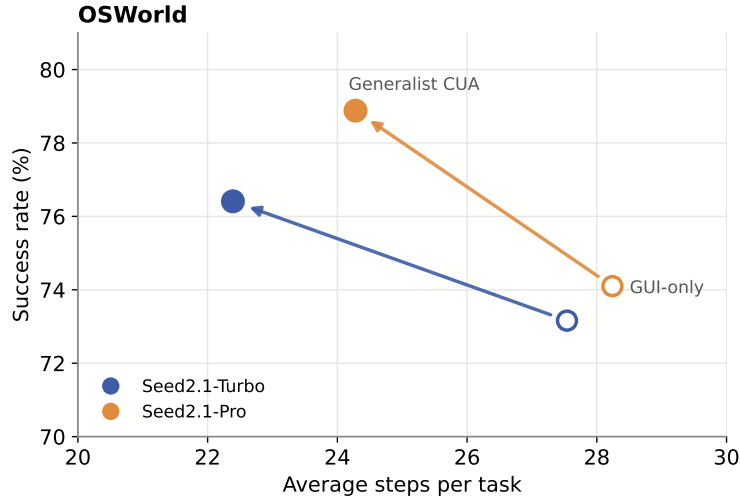


Figure 5 Efficiency of the Generalist Computer-Use Agent (CUA) setting on OSWorld. For each model, the arrow points from the GUI-only operating point (open marker) to the Generalist CUA setting (filled marker), in which the agent may also issue commands and call tools. Both models move up and to the left: higher success rate with fewer steps per task.

2.5.1 Results

As shown in Table 4, Seed2.1 demonstrates strong capability as a unified digital agent across desktop, mobile, creative-productivity, and browser-game environments. On OSWorld, Seed2.1-Pro reaches 78.2, outperforming 3.1-Pro and approaching GPT-5.5, showing strong general computer-use capability in realistic operating-system workflows. On MobileWorld GUI-only, both Seed2.1-Turbo and Seed2.1-Pro substantially outperform the listed frontier baselines, with Seed2.1-Pro reaching 73.3 compared with GPT-5.5 at 62.4, Claude-4.7-Opus at 56.4, and 3.1-Pro at 58.1. On CreativeWork, Seed2.1-Pro achieves the highest listed score at 42.5, while Seed2.1-Turbo remains above GPT-5.5, Claude-4.7-Opus, and 3.1-Pro, indicating strong capability in hybrid GUI-and-MCP productivity workflows. On GameWorld, Seed2.1-Pro outperforms Claude-4.7-Opus and 3.1-Pro, showing that Seed2.1 can also handle interactive multimodal game environments. Overall, these results suggest that Seed2.1 is not limited to text, tools, or code alone, but can coordinate GUI control, tool use, multimodal perception, and long-horizon execution in unified digital-agent workflows.

Benchmark	GPT-5.5	Claude-4.7 Opus	Gemini-3.1 Pro	Seed2.1 Turbo	Seed2.1 Pro
OSWorld [70]	78.7	82.8	76.2	76.4	78.8
MobileWorld [29]	54.7	57.1	48.4	70.0	73.1
CreativeWork	30.5	28.3	27.4	34.5	42.5
GameWorld [42]	34.1	26.5	21.2	25.9	31.2

Table 4 Generalist Computer-Use Agent evaluation. CreativeWork is a Seed-developed benchmark for hybrid GUI-and-MCP productivity tasks.

2.5.2 Why the Generalist setting matters for deployment.

Figure 5 compares two regimes for the same model on OSWorld: GUI-only, and the Generalist Computer-Use Agent (CUA) setting, where the agent may also issue commands and call tools. For both models, the Generalist setting moves the operating point up and to the left—higher success at lower cost. Success rises from 73.2% to 76.4% (Turbo) and 72.6% to 78.8% (Pro), while average steps drop from 27.5 to 22.3 (−18.7%) and 28.2 to 24.2 (−16%). The agent thus solves more tasks with roughly one-fifth fewer actions. This is decisive for real

deployment, not just leaderboards: every step on a live system adds latency, cost, and another chance for an irreversible error, so shorter trajectories are themselves more reliable. A single command often replaces a long, fragile chain of GUI clicks. That the trend holds across two model scales indicates a structural property of the paradigm: giving the agent non-GUI actions and letting it choose when to use them makes it both more capable and more efficient.

2.6 Case Study

The following case studies illustrate Seed2.1’s general-agent capability in realistic high-value workflows. To keep the main text focused, we summarize the task requirements and model behavior here, while the corresponding visual showcases are provided in Appendix D.

2.6.1 Educational Task

As shown in Figure 24, we evaluate whether Seed2.1 can complete a document-centric academic task that requires multi-document reasoning, precise calculation, and layout-preserving PDF editing. Given a GPA policy document, a transcript, and a curriculum, the model must map credits to 66 courses, handle unmatched electives, convert five-level grades into percentage scores, compute total credits and GPA, and fill only the blank fields without modifying the original transcript format.

2.6.2 Financial Task

As shown in Figure 25, we evaluate whether Seed2.1 can perform structured financial risk analysis over a constrained equity universe. The task requires selecting non-financial CSI 500 constituents as of April 24, 2026, excluding banking and non-bank financial firms, extracting 2025 annual-report indicators, combining them with latest market capitalization, and computing the Altman Z-score for each stock. This tests whether the model can enforce domain-specific screening rules, derive intermediate accounting variables, and rank companies by financial distress risk.

2.6.3 Legal Task

As shown in Figure 26, we evaluate whether Seed2.1 can support a professional bankruptcy reorganization workflow by converting fragmented commercial facts into an actionable legal memorandum. The task requires analyzing Ningbo Obsidian’s reorganization timeline, claim declaration deadline, creditor meeting schedule, and three transaction contracts involving matured debt, unmatured equipment payment, overdue interest, and a reciprocal rebate obligation. This tests whether the model can map factual events to bankruptcy-law consequences, including claim filing, interest suspension, unmatured-claim acceleration, and statutory set-off.

2.6.4 Medical Task

As shown in Figure 27, we evaluate whether Seed2.1 can support complex clinical decision-making in a neuropathological case requiring integrated diagnosis and treatment planning. The case involves a 38-year-old female with progressive language disturbance and bradyphrenia, MRI evidence of a deep left temporal mass-like lesion near the language functional area, and intraoperative pathology showing thin-walled sinus-like vascular spaces with hemorrhage of different ages, perivascular hemosiderin deposition, and gliosis, without neoplastic glial proliferation. This tests whether the model can synthesize clinical presentation, neuroimaging, lesion location, and microscopic pathology to identify the most precise disease entity rather than relying on broad descriptive labels.

2.6.5 Operations Analytic Task

As shown in Figure 28, we evaluate whether Seed2.1 can transform fragmented Q1 operational data into a board-ready, formula-driven Excel analysis system. The task requires operating on eight raw worksheets covering nurse profiles, care packages, clients, contracts, costs, inquiries, early terminations, and attendance, while preserving the original data sheets. This tests whether the model can establish a governed analytical

backend, including standardized nurse IDs, cleaned names, normalized hiring dates, tier coefficients, tenure calculations, and visual quality-control flags.

2.6.6 Airline Operations Task

As shown in Figure 34, we evaluate whether Seed2.1 can transform large, multi-source airline operations data into a traceable Q4 executive review workbook. The task requires preserving the original ten data tables while building formula-driven analysis sheets for 300 flight records, including flight-number normalization, date and time standardization, overnight duration calculation, aircraft-code cleaning, route and airport mapping, seat-capacity lookup, and Passenger Load Factor computation. This tests whether the model can convert natural-language operational requirements into maintainable Excel logic rather than producing a static summary.

2.6.7 Personal Government-Service Scheduling Task

As shown in Figure 30, we evaluate whether Seed2.1 can convert a complex personal administrative scenario into an executable government-service handling plan. The case involves residence permit application, social insurance certificate printing, second-hand housing title transfer, utility handover, household registration relocation across provinces, Shanghai-to-Hefei housing provident fund transfer, and ID-card renewal, all under a hard constraint of only three available half-day offline appointments before June 30, 2026. This tests whether the model can reason over official policy channels, jurisdictional differences, eligibility conditions, processing timelines, and dependency ordering across multiple public-service procedures.

3 Production Coding

During the iterative development of Seed 2.1, large-scale dogfooding was conducted among both internal and external developers. The model was continuously improved based on developer feedback, ultimately achieving significant gains in coding capabilities. In large-scale anonymous developer feedback, Seed 2.1 achieved a preference ranking superior to Claude Opus 4.6, reaching a level suitable for enterprise production use.

3.1 Crowdsourced Developer Evaluation

3.1.1 Claude Code

During Seed 2.1 development, we invited Coding Plan developers to participate in crowdsourced evaluation on real development tasks. Developers uploaded code repositories, submitted realistic engineering requests, and compared outputs from anonymized models under the same repository, task requirement, and execution environment. In completion-quality preference, Seed2.1-Turbo achieved 100 wins / 19 ties / 59 losses against GLM 5.1 over 178 comparisons, corresponding to a 56.2% win rate and a +23.0 percentage-point net win rate. Seed2.1-Pro achieved 136 wins / 26 ties / 68 losses against Opus 4.6 over 230 comparisons, corresponding to a 59.1% win rate and a +29.6 percentage-point net win rate. These results suggest that Seed 2.1 models are more often preferred by developers for final completion quality in realistic coding workflows.

3.1.2 Trae

Crowdsourced developer testing offers direct evidence of Seed2.1-Pro’s usefulness in real engineering workflows, as shown in Figure 6. Conducted inside Trae, the evaluation drew on authentic tasks contributed by professional developers from their own repositories, closely mirroring day-to-day production work. Each task was executed with the same repository, prompt, and runtime setup, with model identities hidden from reviewers. Seed2.1-Pro was compared head-to-head against a strong frontier reference, Claude Opus 4.7, under a controlled preference protocol with multi-dimensional human ratings, ensuring that the results reflect genuine completion quality rather than surface-level impressions.

Across 167 valid head-to-head tasks, developer preference was nearly evenly split, with Seed2.1-Pro preferred in 81 cases and Claude Opus 4.7 in 86 cases. This corresponds to a 48.5% win rate, placing Seed2.1-Pro at practical parity with Claude Opus 4.7 on production-grade tasks; the two models also achieved the same

mean score across the six rating dimensions (3.967 each). On artifact quality, Seed2.1-Pro produced fully correct, ready-to-use solutions on 29.3% of tasks, compared with 20.4% for the reference model (Pass@1). It also maintained a comparable acceptable-delivery rate (94.0% vs. 92.2%) and produced no severely broken deliverables (0.0% vs. 2.4%), resulting in a higher delivery-completeness score (3.97 vs. 3.83). These results suggest that Seed2.1-Pro is particularly strong in the quality of shipped artifacts, combining a higher rate of first-pass-correct outputs with a clean failure floor.

A separate composite usability rating, which incorporates both the process experience and the delivered output, reveals a more nuanced picture. Seed2.1-Pro reaches the top “fully usable” tier more often than Claude Opus 4.7 (58.7% vs. 54.5%), but it also falls into the “unusable” tier more frequently (11.4% vs. 7.8%), indicating a higher-variance profile. These unusable cases are primarily associated with looser instruction following and boundary adherence (3.96 vs. 4.11 and 4.16 vs. 4.38), rather than with severely broken code, which is consistent with the strong artifact-quality floor described above. By task type, Seed2.1-Pro’s advantage is concentrated in maintenance-style work, including code comprehension, refactoring, and bug fixing, where it is preferred on roughly 60–70% of tasks. This aligns with its strengths in task planning and reliable delivery, while also pointing to greenfield zero-to-one generation and tighter in-scope discipline as important directions for the next iteration.

These aggregate results are further illustrated by representative case studies from the crowdsourced evaluation and frontend showcases, as shown in Figure 6, Figure 9, and Appendix Figures 10, 11, and 12. Spanning scientific-ML library feature development and multimodal frontend artifact generation, these examples show how Seed2.1-Pro reasons over existing codebases, follows visual and product constraints, plans targeted changes, and delivers complete, runnable results.

3.2 Automatic Evaluation

The coding-agent evaluation suite combines public benchmarks and Trae-oriented internal benchmarks. The public benchmarks cover complementary coding-agent abilities, including command-line task execution, repository-level software engineering, cyber-security task solving, program synthesis, and natural-language-to-repository code modification. The Trae benchmarks are designed to reflect realistic product usage in an AI coding environment, covering web development, repository environment setup, artifact generation, bug fixing across multiple programming languages, and code generation. Together, these evaluations test whether the model can move beyond isolated code completion and operate as an end-to-end engineering assistant.

NL2Repo-Bench NL2Repo-Bench evaluates a model’s ability to convert natural language requirements into repository-level code changes [15]. Unlike single-function coding benchmarks, it requires repository understanding, file selection, dependency reasoning, and multi-file modification. This makes it closer to real software engineering workflows, where an agent must understand the existing codebase before producing coherent and maintainable changes.

SeedKernelBench SeedKernelBench is a Seed-developed benchmark for evaluating GPU kernel coding and performance optimization capability. It collects operator-level tasks mined from mainstream high-performance kernel libraries, including FlashInfer, SGLang, FlashAttention, DeepGEMM, CUTLASS, and cuDNN. Unlike conventional coding benchmarks that mainly evaluate functional correctness, SeedKernelBench focuses on whether a model can generate or optimize CUDA-style kernels that are both correct and performant. We report the average speedup ratio over reference implementations, where higher values indicate stronger kernel-level optimization capability.

Trae Agent Bench Trae Agent Bench is a Seed-developed benchmark designed to reflect real coding-agent usage in Trae. Instead of treating coding ability as isolated code completion, the benchmark focuses on end-to-end engineering workflows that users actually ask an agent to complete, including web development, bug fixing, feature implementation, repository understanding, artifact generation, and runtime environment setup. Its construction is guided primarily by product-side user intent, with programming-language coverage adjusted according to observed Trae usage.

Unlike open-source coding benchmarks that often evaluate isolated programming skills in fixed task settings, Trae Agent Bench is built around the actual user experience of an AI coding product. The benchmark emphasizes whether an agent can complete the kinds of tasks users submit in Trae, including fixing repository bugs, implementing product features, generating frontend artifacts, understanding existing codebases, and recovering runnable environments. It therefore includes repository-environment tasks, where the agent must

MODEL CARD · CASE STUDY

Adding a verified discrete-time controlled-system identifier to PySINDy

Scientific-ML library feature dev · agent harness: Trae SOLO Agent

Task Overview

User: PySINDy currently targets continuous-time ODE sparse identification. Add a capability for discrete-time controlled systems, $x[k+1] = f(x[k], u[k])$: a new `DiscreteSINDyControl` estimator that fits a sparse discrete map from (X, U) sequences, does multi-step rolling prediction, and scores R^2 on a test sequence. Constraints: (1) strictly reuse the existing `BaseEstimator / Optimizer / FeatureLibrary` three-layer pattern and the `sklearn fit/predict/score` API — don't bypass the pipeline; (2) `numpy + scipy + scikit-learn` only (no `jax / pytorch / tensorflow`); (3) follow the repo's black + isort + flake8 style with `numpydoc` docstrings; (4) support arbitrary-order polynomial + trigonometric libraries, expand multi-column `U` correctly (not as one concatenated column), and support `differentiation_method=None` to skip differentiation; (5) add a minimal example (controlled logistic map or similar) recovering the true equation from noisy data. Finish with a short design-trade-off note.

Assistant · workflow reconstructed from the execution trace (60 tool calls: `RunCommand 17 / Read 12 / Edit 12 / TodoWrite 10 / Write 4 / LS 2 / DeleteFile 2`)

[Stage 1] Read the repo, then probe what already exists

LS + Read `pysindy/_core.py, feature_library/*, utils/*` -> map the `BaseEstimator / Optimizer / FeatureLibrary` layers
`python3 -c "ps.DiscreteSINDy(...).fit(x, u=u); model.print()" -> recovers 0.8 x0 + 0.1 x1 + 0.5 u0 / 0.2 x0 + 0.7 x1 + 0.3 u1`
 » **The repo already ships a `DiscreteSINDy` that expands multi-column `U` correctly — so the real gap is a dedicated, spec-compliant estimator, not a brand-new pipeline.**
Reasoning: probe the existing class empirically before writing anything, so the new code reuses a path that already works rather than duplicating it.

[Stage 2] Pin down the discrete-control contract

» **Inherit `_BaseSINDy`; reject `differentiation_method` (a discrete map regresses directly onto $x[k+1]$, no derivative needed).**
 » **Concatenate `U` onto `X` along the feature axis so `PolynomialLibrary` naturally produces `x_i-u_j` cross terms; score defaults to open-loop multi-step R^2 .**
Reasoning: binding the target to $x[k+1]$ and feeding $[x, u]$ through the shared library is exactly what keeps the new estimator inside the existing three layers.

[Stage 3] Implement inside the existing pipeline

Edit `pysindy/_core.py` -> add class `DiscreteSINDyControl(_BaseSINDy)` (`fit / predict / score / simulate / equations / print`)
 Edit `pysindy/_init__.py` -> export `DiscreteSINDyControl`
 Write `examples/16_discrete_control/example.py` -> controlled logistic map + multi-column `U` + noise
 » **Multi-column control is handled by a single feature-axis concat reusing `fit_transform + optimizer.fit` — no bespoke `U`-expansion logic.**

[Stage 4] Verify by execution, lint, then hand off a clean diff

`python3 test_discrete_control.py` (1D logistic · 2D 2-control · poly+Fourier · rollout · multi-trajectory)
 » **The first run flushed out real bugs — a diverging logistic trajectory (`ValueError: infinity`) and an over-strict $R^2 > 0.999$ gate — which it diagnosed and fixed until all five scenarios went green, rather than declaring success on a happy path.**
 ✓ **5/5 functional scenarios pass — 1D logistic R^2 : 0.999 one-step / 0.998 rollout · 2D 2-control exact ($R^2=1.000$)**
 ✓ **Exact symbolic recovery — coefficients match the ground-truth equations to 3 decimals on every system · poly+Fourier recovers the sin term**
 ✓ **sklearn estimator-compat checks pass · simulate rollout MSE ≈ 0 · the repository's full test suite re-run with zero regressions**
`flake8 --max-line-length=88 pysindy/_core.py -> E501 @ L1479 -> fix -> 0 errors`
`DeleteFile test_discrete_control.py, test_example.py, _verify.py` (remove scaffolding)
Reasoning: every requirement — recovery, R^2 , the sklearn surface, style — is backed by a real run, and it debugged its own failures to get there, then deleted the scratch files to leave only the intended diff.

[Generated Content] — the delivered estimator, its recovery, and the design note

pyindy/_core.py · new DiscreteSINDyControl (smallest meaningful hunk)

```
@@ reuse the 3-layer pipeline; no differentiation for a discrete map @@
+ class DiscreteSINDyControl(_BaseSINDy): # x[k+1] = f(x[k], u[k])
+ def fit(self, x, u=None, x_next=None, feature_names=None, ...):
+     u = validate_control_variables(x, u)
+     x_aug = [np.concatenate((xi, ui), axis=xi.ax_coord) # multi-col U
+              for xi, ui in zip(x, u)] # -> x_i^T u_j terms
+     f_of_x = self.feature_library.fit_transform(x_aug) # reuse library
+     self.optimizer.fit(features, x_next_cat, ...) # reuse optimizer
+ def score(self, x, u=None, multi_step=True, metric=r2_score): # open-loop R2
```

Sparse recovery on held-out data — actual test-run output

```
# 1D controlled logistic map (+ noise)
(x0)[k+1] = 2.509 x0 + 0.498 u0 - 2.508 x0^2    one-step R^2=0.999 · rollout=0.998
# 2D linear system, two controls
(x)[k+1] = 0.900 x0 + 0.100 x1 + 0.600 u0
(y)[k+1] = 0.800 x1 + 0.200 u0 + 0.500 u1    R^2 = 1.000 (exact)
# polynomial + Fourier (trigonometric term)
(x0)[k+1] = 0.500 x0 + 0.400 u0 + 0.300 sin(x0) multi-step R^2 = 1.000
```

Design trade-offs

Interface: predict keeps one-step semantics through the pipeline; multi-step rollout lives in simulate; score defaults to open-loop multi-step R² (more realistic for discrete systems) with a multi_step=False fallback.

Vs. the continuous ODE path: differentiation is dropped entirely — the target is x[k+1] directly, instead of an estimated derivative.

Limitations: no separate x/u libraries (use GeneralizedLibrary if needed); open-loop R² can fall on chaotic/unstable systems as rollout error accumulates.

Independent expert review

Why it won: the reviewer credited it for building fit / predict / score / simulate / equations / print directly into the existing _core.py, a default open-loop multi-step score that can switch to one-step, and the richest example in the group — 1D logistic, a 2D dual-control system and a joint polynomial+Fourier identification with plotting — whose coefficient recovery it verified before re-running the repository's full test suite to confirm no regression.

Relative standing: the alternative submission was praised for the cleanest module boundary (a standalone new module, zero edits to _core.py) and a GeneralizedLibrary[Polynomial, Fourier] default that maps straight onto the requirement — but once its environment was missing sklearn it fell back to static review only, never running the example or lint, so for a task that explicitly asked to prove sparse recovery from noisy data it carried the lowest landing confidence.

Noted trade-off: the reviewer flagged that satisfying the differentiation_method=None contract implicitly through inheritance lightly under-honored the explicit API constraint, and that the run was on the longer side.

Result Asked to add a discrete-time controlled-system identifier to PySINDy under strict architecture, dependency and style constraints, the model first probed the existing pipeline, then added a DiscreteSINDyControl estimator that reuses the FeatureLibrary → Optimizer → BaseEstimator stack, binds the regression target to x[k+1] (no differentiation), and expands multi-column control inputs into proper cross terms. Crucially, it was the one submission that backed every claim with an actual run — debugging its own numerical overflow, recovering the exact ground-truth equations (R² ≈ 1.0 across linear, logistic and trigonometric systems), re-running the full test suite with zero regressions, and shipping a flake8-clean diff with the scaffolding removed.

Figure 6 The case of Seed2.1 with Scientific-ML library feature dev task.

reproduce dependencies, run tests, and resolve runtime issues, as well as repository question-answering tasks, where the agent must understand architecture, module behavior, repository inputs and outputs, implementation details, and potential bug locations. Task prompts are written with sufficient context to reduce ambiguity, so evaluation failures are more likely to reflect agent capability rather than unclear instructions.

The execution and scoring pipeline is containerized. Each task is solved and scored in a clean task-specific runtime, so changes made by the agent, including dependency installation, configuration updates, and environment modifications, are reflected in the final evaluation result. This makes the benchmark closer to real Trae usage, where successful completion depends not only on producing code, but also on making the repository run correctly.

Category	Benchmark	GPT-5.5	Claude-4.7 Opus	Gemini-3.1-Pro	Seed2.1-Turbo	Seed2.1-Pro
Open Benchmarks	Terminal-Bench 2.1 [38, 54]	73.8	71.7	70.7	67.6	71.0
	SWE-Pro Bench [13]	58.6	64.3	54.2	57.0	57.5
	CyberGym [65]	81.8	73.1	-	67.0	70.2
	ProgramBench [73]	0.5/5.5/65.9	0/2.5/52.1	0/1/40.7	0/0/49.4	0/1/50.3
	NL2Repo-Bench [15]	45.1	58.2	33.4	43.7	47.0
	SWE-Atlas [46]	44.7	38.7	23.6	30.6	35.2
	DeepSWE [52]	70.0	54.0	10.0	23.0	32.7
Trae	Web Bench	81.3	75.0	71.9	73.6	78.4
	SeedKernelBench	9.14x	10.70x	-	8.60x	9.21x
	Repo Env	90.0	63.3	57.1	46.7	55.0
	Artifacts	56.0	53.0	48.0	47.0	51.0
	Error Fix Python	76.0	76.0	75.5	74.0	70.7
	Error Fix JS	79.4	78.2	77.1	69.4	74.6
	Error Fix Java	65.0	73.3	62.5	66.7	66.7
	Error Fix Go	70.0	66.7	60.0	56.7	63.3
	Code Gen Python	73.3	83.3	71.7	73.3	75.6
Code Gen JS	76.0	66.4	65.3	59.7	62.4	

Table 5 Coding-agent evaluation across public and Trae-oriented benchmarks. For ProgramBench, results are reported as *resolved / almost resolved / average pass rate*, where “almost resolved” denotes tasks with at least 95% pass rate. For SeedKernelBench, results are reported as average speedup ratios, where higher is better.

Overall, Trae Agent Bench emphasizes two high-frequency capability families: fixing existing code and implementing new functionality. It also includes repository understanding and environment management as important supporting capabilities. The language and task distributions are summarized in Figures 7 and 8. Representative task examples and the overall showcase link are provided in Appendix A.

Programming-Language Distribution in Trae Agent Bench

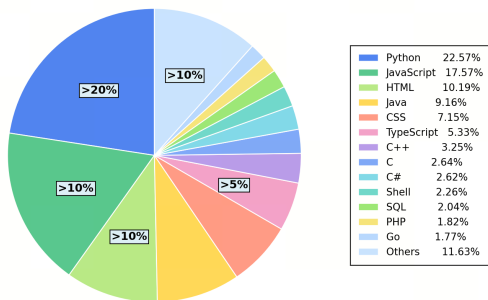


Figure 7 Programming-language distribution in Trae Agent Bench.

Task-Intent Distribution in Trae Agent Bench

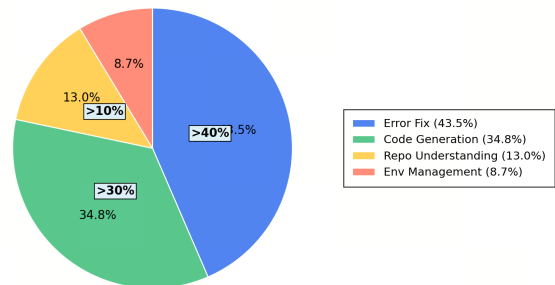


Figure 8 Task-intent distribution in Trae Agent Bench.

3.2.1 Results

As shown in Table 5, Seed2.1-Pro demonstrates competitive coding-agent capability against frontier models across both public benchmarks and Trae-oriented product evaluations. On public coding-agent benchmarks, Seed2.1-Pro is close to the frontier on Terminal Bench 2.1, achieving 71.0 compared with GPT-5.5 at 73.8 and Claude-4.7-Opus at 71.7. It also performs strongly on NL2Repo-Bench, reaching 47.0 and outperforming

GPT-5.5 at 45.1 and Gemini-3.1-Pro at 33.4, while remaining behind Claude-4.7-Opus. On SWE-Pro and SWE-Atlas, Seed2.1-Pro outperforms Gemini-3.1-Pro and approaches the stronger frontier systems, indicating solid repository-level understanding and multi-file code-editing capability. On ProgramBench, Seed2.1-Pro reaches 0/1/50.3, exceeding Gemini-3.1-Pro in average pass rate and approaching Claude-4.7-Opus, although GPT-5.5 remains clearly ahead on the most difficult program-repair tasks.

Seed2.1-Pro also shows competitive capability in performance-oriented kernel coding. On SeedKernelBench, which evaluates GPU kernel implementation and optimization tasks mined from high-performance libraries such as FlashInfer, SGLang, FlashAttention, DeepGEMM, CUTLASS, and cuDNN, Seed2.1-Pro reaches 9.21x average speedup. This slightly exceeds GPT-5.5 at 9.14x, while Claude-4.7-Opus remains the strongest listed model at 10.70x. This result suggests that Seed2.1-Pro’s coding capability extends beyond standard repository editing to low-level, performance-sensitive kernel optimization.

On Trae-oriented internal benchmarks, Seed2.1-Pro shows strong product-level engineering capability under realistic development workflows. It outperforms Claude-4.7-Opus and Gemini-3.1-Pro on Web Bench, and exceeds GPT-5.5 and Gemini-3.1-Pro on Error Fix Java. It also achieves strong results on Code Gen Python, outperforming GPT-5.5 and Gemini-3.1-Pro, while remaining behind Claude-4.7-Opus. Across Artifacts, Error Fix Go, and several frontend and debugging tasks, Seed2.1-Pro remains competitive with frontier models, showing that it can support practical Trae-style workflows involving repository setup, debugging, frontend implementation, code generation, and artifact delivery.

Beyond benchmark scores, Seed2.1-Pro also performs strongly in frontend-oriented human preference evaluation. As shown in Figure 9, it ranks #8 in Code Arena: Frontend with a score of 1539, roughly on par with Claude Opus 4.6, and reaches top-10 performance in five of seven frontend subcategories. Together with crowdsourced developer preferences, these results suggest that Seed 2.1 has reached a strong production-coding level for Trae-style development workflows, while further improvements are still needed on the most challenging open public coding-agent benchmarks.

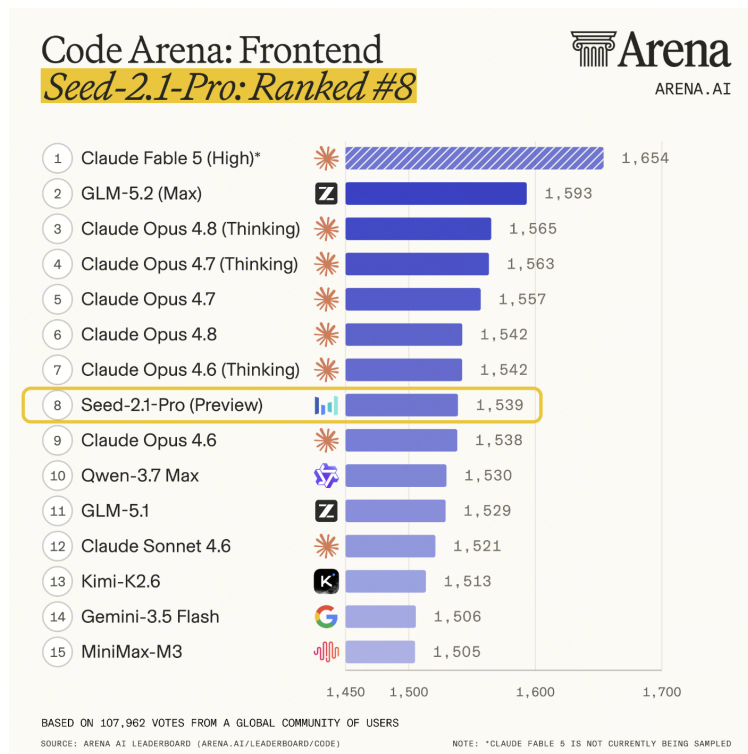


Figure 9 Code Arena frontend ranking snapshot for Seed2.1-Pro . The model ranks #8 overall with a score of 1539 and shows strong performance in React and production-facing frontend subcategories.

Case 1: Multimodal Frontend Artifact Generation

Case 1 User Input

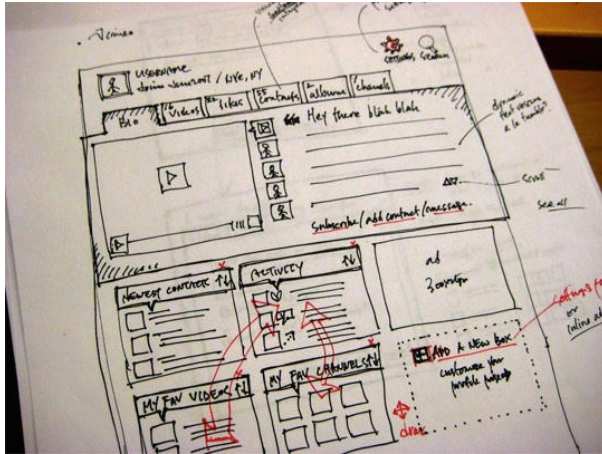
Input images. The user provides a layout sketch, a pixel-style reference, and a dashboard reference, as shown in Figures 10 and 11.

User instruction.

Replicate this social video-dashboard sketch's layout faithfully; apply the pixel look to the chrome only, not the fluid.

Fluid module, centerpiece, must match exactly: in the left video-player block, embed the WebGL fluid identical to the arena reference — vibrant multicolor dye on black, drag the mouse to paint flowing trails, click = splash, with the dat.GUI control panel. The fluid is NOT restyled; the style never touches it.

Interactions the sketch can't convey — implement these: Top tabs, Bio / Videos / Likes / Contacts / Albums / Channels, switch content. Player controls: play / pause, scrubber, volume, quality. Subscribe / Follow + Like / Comment counters that update. Search / filter. Activity feed + favourite-videos / channels grids.



Layout sketch



Pixel-art style reference

Figure 10 Primary visual inputs for the multimodal frontend artifact-generation case.

Given the user input above, the agent generates a social video-dashboard interface that follows the provided layout sketch, applies the pixel-art style to the surrounding application chrome, and keeps the central WebGL fluid module visually smooth and separate from the pixel-styled UI. The generated artifact also implements the requested interactive components, including tab switching, player controls, search and filtering, follow/subscribe actions, updating counters, activity feed, and favorite-video/channel grids.

This case is representative of production coding because the agent must follow multimodal inputs and explicit user constraints, preserve a strict style boundary between the UI chrome and the WebGL fluid module, and implement a stateful frontend artifact rather than producing isolated code snippets.

Case 2: Interactive Intro-to-AI Tutorial Website



Figure 11 Dashboard reference for the multimodal frontend artifact-generation case.

Case 2 User Input

User instruction.

Create a complete interactive introductory tutorial on artificial intelligence, with animation effects. The website should include a full zero-background course, covering concept explanations, in-class quizzes, and hands-on examples. The overall style should be minimal.



Figure 12 Generated homepage for the interactive intro-to-AI tutorial website. The artifact presents a complete beginner-oriented AI course with chapter navigation, course progress, learning statistics, and a minimal visual design.

This case evaluates whether the agent can transform a broad educational product request into a structured interactive learning website. The generated artifact organizes the course into beginner-friendly chapters, presents a clear learning path, and supports interactive learning scenarios such as concept explanation, quizzes, and hands-on examples. It is representative of production coding because the agent must jointly handle curriculum decomposition, information architecture, frontend layout, interaction design, and polished product

delivery.

Case 3: International City Homepage for Shanghai

Case 3 User Input

User instruction.

Collect image materials and generate a city homepage introducing Shanghai for an international audience. The page should be concise, modern, and visually premium.



Figure 13 Generated homepage for introducing Shanghai to an international audience. The artifact uses a premium skyline-centered hero section, modern navigation, bilingual city branding, and concise city highlights.

This case evaluates whether the agent can convert a concise creative brief into a polished city-promotion homepage. The generated artifact presents Shanghai with a modern international visual identity, balancing landmark imagery, city branding, navigation, and key city statistics. It is representative of production coding because success requires visual-material selection, audience-aware content abstraction, page composition, frontend implementation, and high-quality visual presentation rather than isolated code generation.

4 Frontier Research

Frontier research scenarios represent some of the most demanding use cases for LLM-based agents. They require deep reasoning, specialized knowledge, long-horizon problem solving, and reliable intermediate and final outputs. This section evaluates whether Seed2.1 can act as an intelligent collaborator for scientific research, computer science, advanced mathematics, and model-development workflows.

4.1 Benchmark Results

The frontier research evaluation suite focuses on tasks that go beyond routine productivity and require research-level problem solving. We evaluate on four benchmarks, each targeting a distinct capability axis: PostTrainBench [47] tests whether LLM agents can automate LLM post-training pipelines under bounded compute; FrontierScience-Research [62] requires multi-step scientific reasoning grounded in domain literature; FrontierCS [36] poses open-ended computer-science problems whose solution quality can be objectively

measured; and HorizonMath [59] targets research-level mathematical discovery on predominantly unsolved problems with automatic verification.

Benchmark	GPT-5.5	Claude-4.7 Opus	Gemini-3.1 Pro	Seed2.1 Turbo	Seed2.1 Pro
PostTrainBench [47]	25.0	27.4	-	18.3	16.5
FrontierScience-Research [62]	33.9	20.0	16.7	33.3	28.3
FrontierCS [36]	41.0 / 40.0 58.6	-	43.8 / 44.1 64.4	33.5 / 33.4 50.8	29.1 / 28.2 46.3
HorizonMath [59]	7.1	4.0	4.0	2.0	2.0

Table 6 Frontier research evaluation. FrontierCS evaluates open-ended computer-science problem solving and reports three official sub-scores for each available model.

As shown in Table 6, Seed2.1 delivers competitive performance across frontier research scenarios that demand advanced reasoning, verification, and long-horizon planning. On FrontierScience-Research, Seed2.1 Turbo scores 33.3, within 0.6 points of GPT-5.5 (33.9) and nearly double the scores of Claude-4.7 Opus (20.0) and Gemini-3.1 Pro (16.7). On FrontierCS, Seed2.1 reports results across all three official sub-scores, confirming its ability to sustain open-ended computer-science problem solving over extended reasoning horizons.

4.2 Deep Think

Seed2.1 Deep Think is an inference-time reasoning configuration designed for complex reasoning tasks in frontier research and advanced engineering. Unlike standard inference modes that directly produce a final response, Deep Think executes an automated loop of *reason* \rightarrow *verify* \rightarrow *revise* \rightarrow *select*. Within this loop the model can invoke web search to retrieve domain literature and use sandbox-based code execution to test intermediate hypotheses, check implementation details, and refine candidate solutions before committing to a final answer. The pipeline is modular: individual components—search, code execution, verifiers, external harnesses, task-specific toolchains—can be swapped or extended. This architecture makes Deep Think especially suitable for problems where correctness depends on multi-step reasoning, executable validation, and iterative refinement.

Benchmark	Seed2.1 Pro	Seed2.1 Deep Think	Gemini-3.1 Pro	Gemini-3.1 Deep Think
IMO 2025	65.2	81.0	-	81.5
IMOProof-Adv [35]	54.3	83.8	49.0	76.7
IPho 2025	79.3	89.0	76.3	87.7
FrontierScience-Research [62]	33.3	40.7	16.7	23.3

Table 7 Frontier research evaluation results of Seed2.1 Deep Think.

As shown in Table 7, Seed2.1 Deep Think substantially outperforms Seed2.1 Pro and reaches a level competitive with—or exceeding—Gemini-3.1 Deep Think: it scores 81.0 vs. 81.5 on IMO 2025, 83.8 vs. 76.7 on IMOProof-Adv, and 40.7 vs. 23.3 on FrontierScience-Research. These gains suggest that iterative reasoning with tool-assisted verification is an effective path for improving scientific-research assistance. More broadly, the results indicate that Seed2.1 is moving beyond standard coding and question-answering toward research-oriented agentic workflows, where models decompose problems, reason through domain constraints, verify intermediate results, and accelerate scientific exploration.

4.3 Scientific and Mathematical Research Assistance

Beyond benchmark-style evaluation, we further examine whether Seed2.1 can assist frontier research workflows where success requires domain understanding, long-horizon reasoning, tool use, and iterative verification. We focus on two representative settings: physics-oriented scientific computing and exploratory mathematical construction.

Physics-oriented scientific computing. In a representative physics case, Seed2.1 is asked to implement form-factor quadrature and assemble `z_core` for frozen-core quasiparticle band narrowing. Unlike ordinary programming tasks, this task requires the model to connect domain theory, numerical formulas, element-specific data files, executable code, and verifier feedback. The model must read task requirements and theoretical formulas, inspect sodium and aluminum core data, implement the required numerical pipeline, and iteratively repair the solution according to verifier feedback.

This case is representative of frontier scientific computing because success depends on far more than code generation: the model must understand the scientific objective, translate formulas into numerically stable procedures, respect environment constraints (e.g., no external dependencies beyond NumPy/SciPy), repair edge cases exposed by the verifier, and produce reproducible artifacts. As shown in Figure 14, Seed2.1 can operate within domain-specific research-engineering workflows where theory, computation, and verification are tightly coupled.

Mathematical research assistance. We also evaluate Seed2.1’s ability to assist exploratory mathematical research. Beyond solving benchmark-style math problems, research assistance requires the model to help mathematicians search for constructions, test proof ideas, and reduce the trial-and-error cost of constructive arguments. Related public evaluations include HorizonMath and other recent mathematical reasoning benchmarks [1, 4, 35, 59].

A representative case comes from research on the local geometry of oscillatory integrals on three-dimensional Riemannian manifolds [12]. In this interaction, mathematicians used Seed2.1 to search for an explicit construction related to the second item of Theorem 2.7. Seed2.1 produced a correct construction example that helped guide the complete metric construction and proof. The final theorem argument and construction example were independently verified by the researchers.

As shown in Figure 15, this case suggests that Seed2.1 can support frontier mathematical exploration by generating candidate constructions and compressing the trial-and-error cost in constructive proof development.

Together, these cases illustrate a capability profile that goes beyond single-turn question answering. In both scientific computing and mathematical exploration, Seed2.1 acts as a research-engineering partner: it ingests domain context, proposes concrete implementations or constructions, iterates under external feedback, and helps convert abstract research goals into independently verifiable artifacts.

4.4 Seed for Seed

Seed for Seed explores how Seed2.1 can support the model-development lifecycle itself, rather than only serving as a downstream user assistant. As illustrated in Figure 16, Seed for Seed organizes model self-improvement around connected loops such as the Eval Loop, Data Loop, Training Loop, Infra Loop. As the model’s coding and agentic capabilities improve, it becomes increasingly capable of participating in long-running R&D workflows such as evaluation construction, failure analysis, data cleaning, data synthesis, training iteration, and infrastructure optimization.

4.4.1 Overview

To make this capability concrete, we present representative cases spanning these loops. Across these cases a common pattern emerges: humans define goals and quality gates, while agents execute long-horizon iterative workflows—inspecting intermediate results, repairing failures, and consolidating reusable artifacts. (summarized in Table 8). Across these cases a common pattern emerges: humans define goals and quality

Task Overview

- The current directory contains the following files and folders:
- `instruction.md`: Task requirements and environment description.
 - `THEORY.md`: Theoretical background and formulas.
 - `Na/`: Development set data for Sodium.
 - `Al/`: Development set data for Aluminum.

You need to implement the form-factor quadrature and assemble `z_core`. Please carefully read the task requirements in `instruction.md` and `THEORY.md`, and write a script named `run_qp.py` for frozen-core quasiparticle band-narrowing.

Assistant:

[Stage 1: Initial Exploration & Data Insepection]

```
execute_bash(command="ls -la /workspace/")
str_replace_editor(command="view", path="/workspace/instruction.md")
str_replace_editor(command="view", path="/workspace/THEORY.md")
[Observation: Task requires calculating frozen-core quasiparticle narrowing using core data]

execute_bash(command="ls -la /workspace/Na/")
str_replace_editor(command="view", path="/workspace/Na/element_config.json")
str_replace_editor(command="view", path="/workspace/Na/core_model.json")
str_replace_editor(command="view", path="/workspace/Na/grid.csv")
execute_bash(command="head -20 /workspace/Na/atomic_core_1s.csv")
execute_bash(command="tail -10 /workspace/Na/atomic_core_1s.csv")
[Observation: Verified the CSV structure of the radial core data and the integration grid settings]
```

[Stage 2: Environment Probing & Base Setup]

```
execute_bash(command="julia -e 'using DFTK; println(DFTK.version())'") execute_bash(command="echo $JULIA_PROJECT")
execute_bash(command="ls -la /nix/store/ | grep julia")
execute_bash(command="which python3 && python3 --version")
execute_bash(command="python3 -c 'import numpy; import scipy...'"")
[Observation: Verified Python3 with numpy/scipy exists, but Pandas might be risky. Julia executable needs to be located.]

execute_bash(command="find / -name julia -type f -executable 2>/dev/null")
execute_bash(command="apt-get update && apt-get install -y julia")
execute_bash(command="mkdir -p /workspace/downloads/")
str_replace_editor(command="create", path="/workspace/downloads/run_qp.py")
[OVER 6 EDITING OPERATIONS]
str_replace_editor(command="create", path="/workspace/downloads/method.md")
```

[Stage 3: First Verifier Feedback & Deep Code Refinement]

[Observation from Verifier 1: The core logic is correct, but there are 4 flaws:

1. Julia Syntax Error (scres.p),
2. Unnecessary Pandas Dependency,
3. Numerical Fragility in K=0 Check,
4. Manual KPoint Construction.]

```
str_replace_editor(command="view", path="/workspace/downloads/run_qp.py")
str_replace_editor(command="str_replace", path="/workspace/downloads/run_qp.py", old_str="import pandas as pd...", new_str="import csv...")
str_replace_editor(command="str_replace", path="/workspace/downloads/run_qp.py", old_str="grid_df = pd.read_csv...", new_str="grid_data = np.genfromtxt...")
str_replace_editor(command="str_replace", path="/workspace/downloads/run_qp.py", old_str="if K == 0.0:", new_str="if K < 1e-12:")
str_replace_editor(command="str_replace", path="/workspace/downloads/run_qp.py", old_str="scres.p", new_str="p=scres.p")
[OVER 12 VIEW AND EDITING OPERATIONS FIXING KPOINTS_FRAC SCOPE & TYPOS]
str_replace_editor(command="view", path="/workspace/downloads/method.md")
str_replace_editor(command="str_replace", path="/workspace/downloads/method.md", old_str="pandas", new_str="...")
execute_bash(command="ls -la /workspace/downloads/")
```

[Stage 4: Second Verifier Feedback & Final Polishing]

[Observation from Verifier 2: The logic is fully sound. Minor soft spots found:

1. kpoints variable is still undefined at line 116/118 in Julia script (should be kpoints_frac),
2. Unnecessary conjugate operation on real f_K .]

```
str_replace_editor(command="view", path="/workspace/downloads/run_qp.py")
str_replace_editor(command="str_replace", path="/workspace/downloads/run_qp.py", old_str="length(kpoints)", new_str="length(kpoints_frac)") (Fixed the
remaining variable scope bug)
execute_bash(command="grep -n \"kpoints\" /workspace/downloads/run_qp.py")
[Observation: grep confirmed that the undefined 'kpoints' is completely eliminated from the codebase]

str_replace_editor(command="view", path="/workspace/downloads/method.md")
str_replace_editor(command="view", path="/workspace/downloads/run_qp.py")
execute_bash(command="ls -lh /workspace/downloads/")
[Observation: Final artifacts run_qp.py and method.md are correctly generated and polished.]
```

[Generated Content]

```
--- method.md ---
## Method for Frozen-Core Quasiparticle Band Narrowing
```

```
## Overview
This implementation computes the frozen-core quasiparticle (QP)
correction to Kohn-Sham (KS) band energies using the formula:
```

$$E_{\text{QP}}(n, k) - E_{\text{F}} = z_{\text{core}}(n, k) * (E_{\text{KS}}(n, k) - E_{\text{F}})$$

```
where `z_core` is the core renormalization factor.
```

```
...
```

```
--- run_qp.py ---
import XXX
```

```
def main():
    # Load element config
    # Load core model
    # Precompute form factors for each core channel
    # Compute J_c = integral u_c^2 V_H_c dr
    # Define f_c(K) function
    # Load grid
    # Write Julia script to run DFTK calculation
    ...
```

Task Overview

Find tensors A, B, C on \mathbb{R}^3 with $A(X, Y)$ symmetric and $B(Z, X, Y), C(W, Z, X, Y)$ symmetric in their last two variables, so that for every orthogonal basis X, Y, Z , $A(X, Y), B(Z, X, Y), C(Z, Z, X, Y)$ are not all zero.

Assistant Response (Summarized)

[Stage 1: Fix coordinates and give an explicit triple]

Fix the standard orthonormal basis e_1, e_2, e_3 of \mathbb{R}^3 , and write $U_i = \langle U, e_i \rangle$. Define

$$\begin{aligned} A(X, Y) &= X_1 Y_2 + X_2 Y_1, \\ B(Z, X, Y) &= Z_1(X_3 Y_2 + X_2 Y_3) - Z_2(X_3 Y_1 + X_1 Y_3), \\ C(W, Z, X, Y) &= \langle W, Z \rangle (X_1 Y_1 - X_2 Y_2). \end{aligned}$$

The symmetries are immediate: A is symmetric in X, Y ; every bracket in B is symmetric in X, Y ; and the final factor in C is symmetric in X, Y .

[Stage 2: Normalize the orthogonal basis]

If X, Y, Z are an arbitrary orthogonal basis, set $X' = X/\|X\|$, $Y' = Y/\|Y\|$, $Z' = Z/\|Z\|$. By multilinearity,

$$\begin{aligned} A(X, Y) &= \|X\| \|Y\| A(X', Y'), \\ B(Z, X, Y) &= \|Z\| \|X\| \|Y\| B(Z', X', Y'), \\ C(Z, Z, X, Y) &= \|Z\|^2 \|X\| \|Y\| C(Z', Z', X', Y'). \end{aligned}$$

Thus simultaneous vanishing is unchanged by normalizing. It suffices to consider orthonormal triples.

[Stage 3: Handle the axis case using B]

Suppose $(X_1^2 + X_2^2)(Y_1^2 + Y_2^2) = 0$. Then either $X = \pm e_3$ or $Y = \pm e_3$. If $X = \pm e_3$, then $Y = (Y_1, Y_2, 0)$ and $(Z_1, Z_2) = \pm(-Y_2, Y_1)$, so

$$B(Z, X, Y) = \pm(Z_1 Y_2 - Z_2 Y_1) = \pm(\mp(Y_1^2 + Y_2^2)) \neq 0.$$

The case $Y = \pm e_3$ is identical, so simultaneous vanishing is impossible.

[Stage 4: Handle the generic case using a two-square identity]

Now suppose $(X_1^2 + X_2^2)(Y_1^2 + Y_2^2) \neq 0$. Let $D(X, Y) = X_1 Y_1 - X_2 Y_2$. Then

$$A(X, Y)^2 + D(X, Y)^2 = (X_1^2 + X_2^2)(Y_1^2 + Y_2^2) \neq 0.$$

Thus A and D are not both zero. Since $C(Z, Z, X, Y) = D(X, Y)$ for $\|Z\| = 1$, either $A \neq 0$ or $C \neq 0$.

[Stage 5: Conclusion for the first construction]

For every orthogonal basis X, Y, Z , at least one of

$$A(X, Y), B(Z, X, Y), C(Z, Z, X, Y)$$

is nonzero. Hence the displayed tensors solve the problem.

Figure 15 Math research-assistance showcase. Seed2.1 proposes an explicit tensor construction and verifies the required non-vanishing condition through a staged proof. The construction provided useful guidance for the second item of Theorem 2.7 in the oscillatory-integral geometry work [12].

Seed Enters the Model Development Loop

Guided by human goals and quality gates, Seed agents plan, act, evaluate, diagnose, and iterate across AI R&D

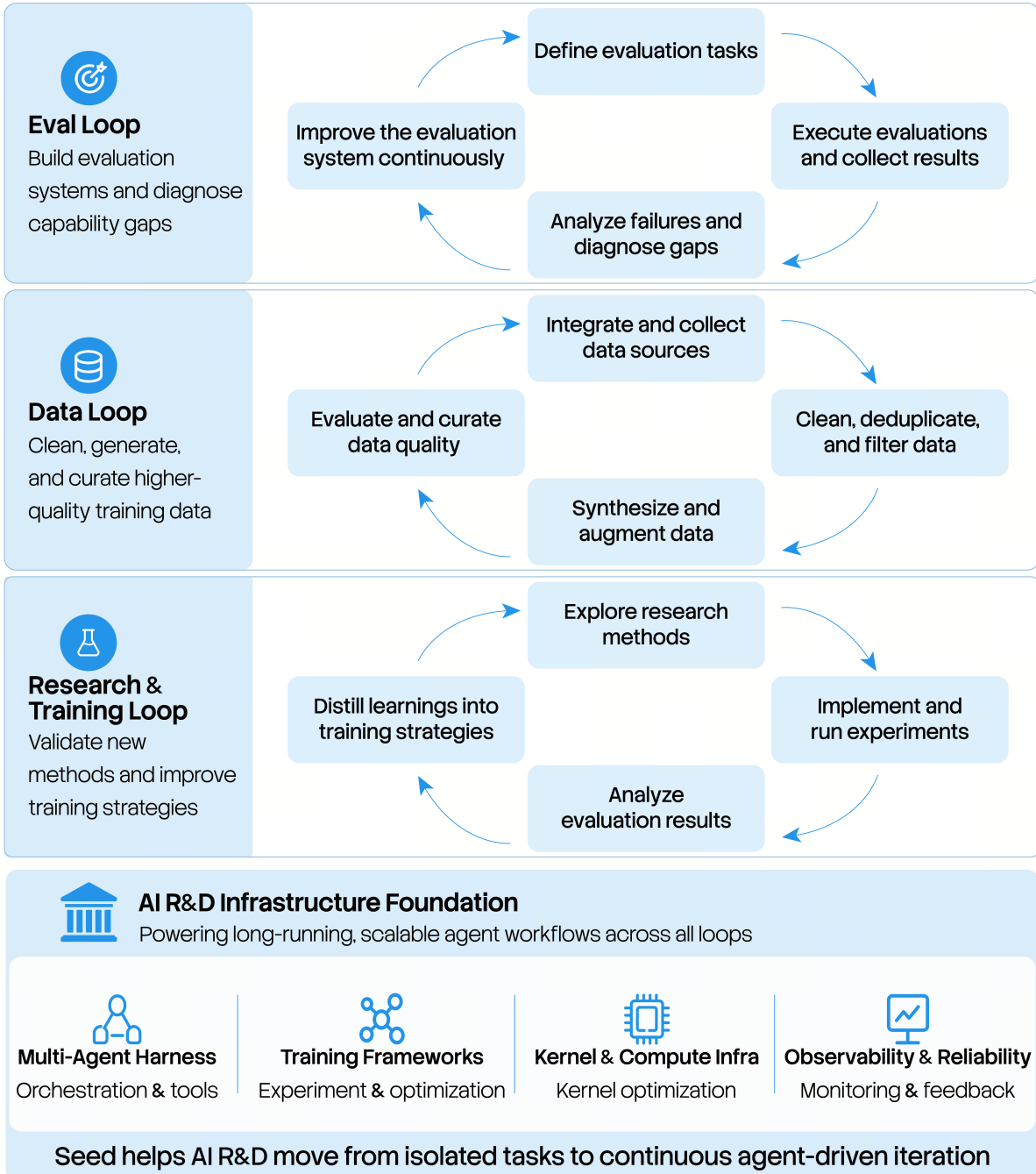


Figure 16 Seed-for-Seed development loop. Seed2.1 participates in evaluation, data, training, and infrastructure loops through long-horizon execution and continuous target-driven iteration.

gates, while agents execute long-horizon iterative workflows—inspecting intermediate results, repairing failures, and consolidating reusable artifacts.

- **Automated Benchmark Integration (Eval Loop):** agents read external benchmark papers or repositories, adapt them to internal evaluation harnesses, configure judges or verifiers, run smoke tests, repair failures, and align scoring protocols—turning what was manual engineering work into an agent-executed workflow.
- **Automated Model Diagnosis (Eval Loop):** Auto-Eval agents select or generate targeted tests, while Post-Eval agents analyze traces, tool calls, artifacts, and metric changes to identify capability gaps and root causes. This moves evaluation beyond static score reporting toward active capability analysis.
- **Iterative Rule Cleaning (Data Loop):** Optimizer and Evaluator agents iterate over regex/rule filters for noisy pretraining data, checking precision, recall, false positives, and false negatives under explicit quality constraints—converting repetitive manual filtering into a long-horizon optimization loop.
- **SFT Data Synthesis (Training Loop):** agents propose data-synthesis plans, generate training data, launch SFT jobs, trigger companion evaluations, and summarize experiment results, closing the loop from data hypothesis to measured outcome.
- **GUI Agent Co-Evolution (Training Loop):** agents compare successful and failed GUI trajectories, locate the earliest divergence step, and generate leak-safe corrective skills or training examples—allowing model failures to become direct inputs for the next training round.
- **RL Framework Optimization (Infra Loop):** agents build a dedicated multi-agent harness in which actor agents implement framework changes and critic agents evaluate code quality, experiment results, and quality-gate satisfaction, enabling engineering experience to accumulate across iterations.

4.4.2 Showcase Evidence

We highlight seven of the above cases with quantitative or qualitative evidence, spanning the Eval, Data, Training, Infra, and Kernel loops.

Eval Loop: Automated External Benchmark Integration. Automated benchmark integration pipeline (Figure 17). Given only a repository URL, the agent autonomously executes an 11-step pipeline—from cloning and analyzing the external benchmark, through code generation and infrastructure deployment, to iterative score alignment—producing a production-ready evaluation that faithfully reproduces published baselines. The baselines table shows correctness, latency, tool usage, and cost for the internally aligned configuration (Claude Code CLI + Opus 4.6) across four corpus × format settings (N = 133 questions; exact-match tolerance = 0%), reproduced from the OfficeQA Pro paper §3.2. The entire integration, which previously required approximately three engineer-days per benchmark, completes in roughly six hours with zero human intervention after the initial URL input.

Eval Loop: From Scores to Root Causes. ProEval capability audit of Seed 2.0 Pro on puzzle_zero_shot (250 questions, 25 types; Figure 18). Through 5 iterative rounds with 241 targeted probe questions (95+55+30+31+30), the dual-agent system (Post-Eval + Auto-Eval) discovered that 18 percentage points of failures (across 8 representative puzzle types) were false negatives caused by evaluator design flaws (format matching, single-answer acceptance). After correction, true capability is substantially higher than surface scores suggest. The audit precisely identified three hard capacity limits — search depth (Sudoku cliff at 27 givens), global constraint integration (Skyscrapers cliff at n=6), and generative vocabulary — and demonstrated that expert-level prompt engineering has zero effect on capacity-bounded tasks. The complete audit, which would require weeks of manual expert analysis, was completed autonomously by the agent system.

Data Loop: Autonomous Rule Iteration for Pre-training Data Cleaning. As shown in Figure 19, Evaluator and Optimizer agents alternate per round under a precision ≥ 0.95 gate; iteration runs until the 300M token budget is exhausted. The recall progression curve (top-right) shows Seed 2.1 Pro tracking GPT-5.5 throughout the 300M-token budget, reaching 43.3% vs. 44.5% validation recall. Claude Opus 4.8 leads at 51.6%. All runs fully autonomous, no human intervention.

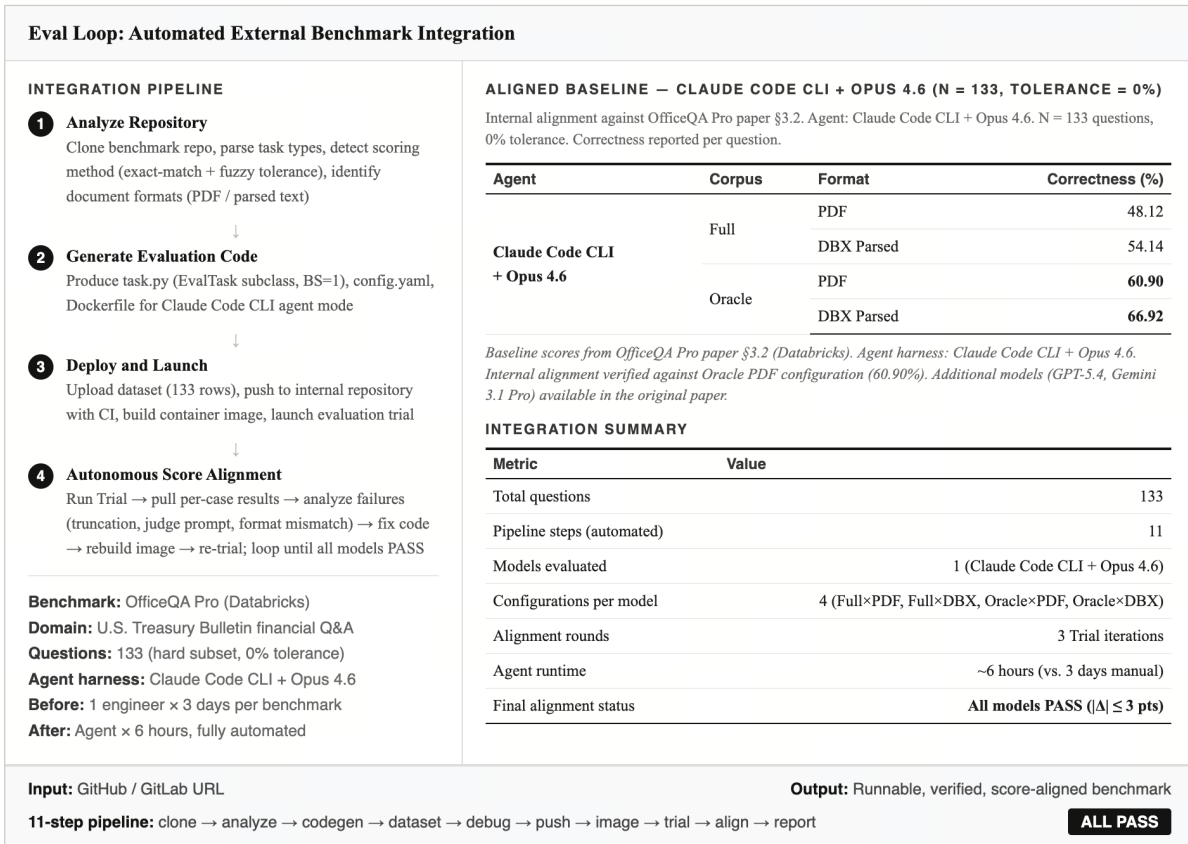


Figure 17 Eval Loop: automated external benchmark integration.

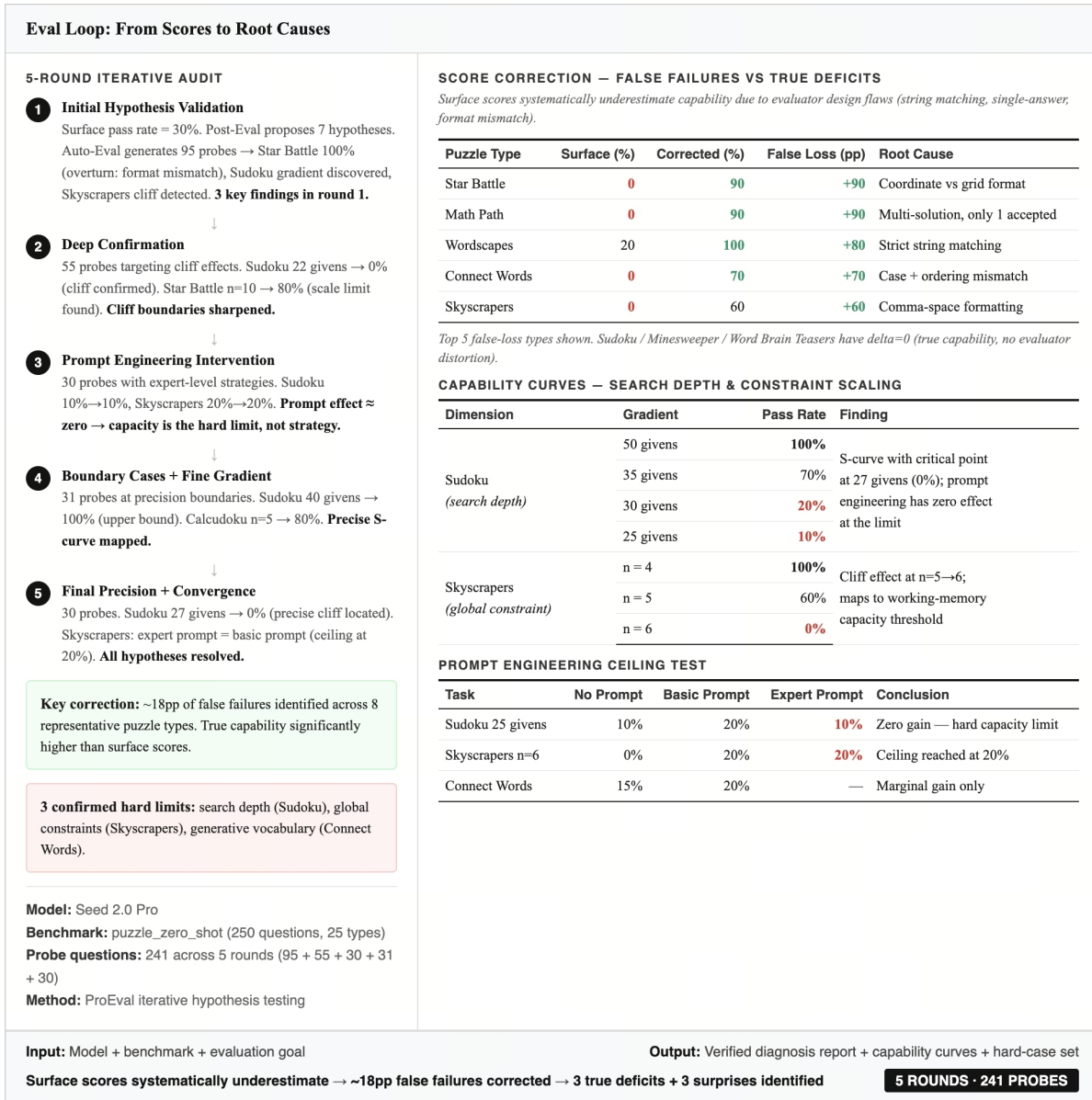


Figure 18 Eval Loop: from scores to root causes.

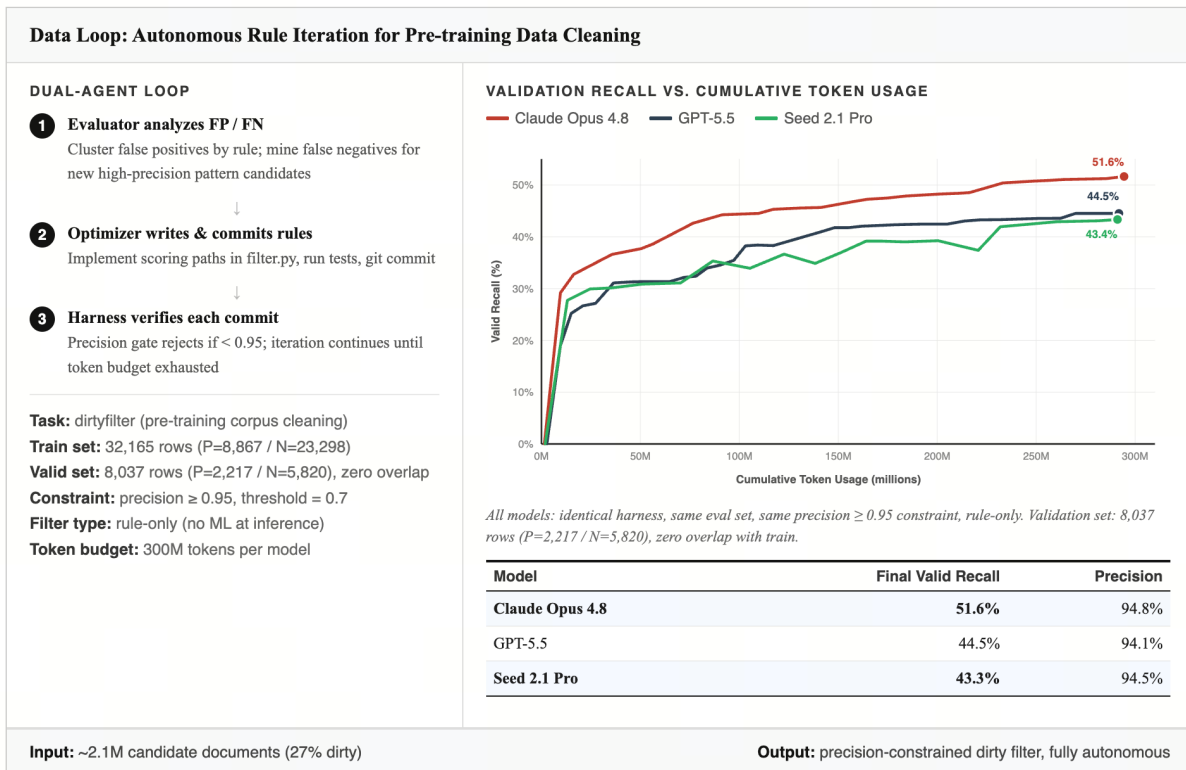


Figure 19 Data Loop: autonomous rule iteration for pre-training data cleaning.

Training Loop: Data-Synthesis-Driven SFT Improvement. A Master Agent orchestrator dispatches an Actor Agent (data synthesis, fork, train, deploy) and a Critic Agent (independent evaluation) in a closed loop (Figure 20). Each iter round launches up to 5 parallel SFT variants spanning a learning-rate \times data-mixture sweep, runs the full 5-stage pipeline (data \rightarrow train \rightarrow deploy \rightarrow eval \rightarrow stop), and ends with the Master Agent selecting a champion and deriving the next-iter strategy. Across 3 iter rounds and 13 experiments over 4.9 days, the champion’s score rose from 0.40 (baseline) to 0.70 (+75)

Training Loop: GUI Trajectory Diagnosis and Auto-Correction. As illustrated in Figure 21, from a single Rollout batch, the system captures a failed trajectory and a success trajectory for a real-world GUI task: purchasing the cheapest high-speed train ticket on Ctrip from Beijing to Shanghai on June 22, 2026. Action-level diff localizes the first divergence to step 8: the failed agent skipped the price-sort tab and selected the cheapest train visible under the default order, G531 at ¥576, while the success trajectory first sorted by price and selected the true lowest-fare option, D11 at ¥296. The failed agent’s own reasoning at the divergence step, “picked the visibly-cheapest train,” quoted verbatim from `assistant_message`, reveals a partial-observability bias caused by relying on the default list order. The system then distills a compact inline correction skill, about 40 tokens, using only locally visible UI state. After injecting the skill, a re-trial succeeds and reduces the fare by 48.6%. The failed/success trajectory pair is then banked as training data, closing both the model-improvement loop, where the skill and trajectory pair become next-round training signal, and the environment-improvement loop, where the same pair informs future question-bank curation. The full diagnosis-and-correction cycle, which would otherwise require hours of manual root-cause analysis, completes in under 30 seconds without human intervention.

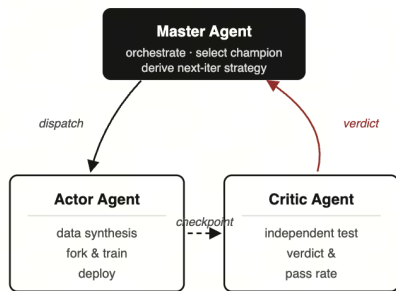
Infra Loop: Multi-Agent Harness for RL Framework Co-Evolution. Given a long-horizon framework re-implementation task, the agent does not begin by editing the target (Figure 22). It first constructs a task-specific multi-agent harness with three specialized roles. The **Actor** decomposes the task goal, designs an implementation plan, and drives development and testing. It autonomously launches RL experiments to validate its work, and iterates between implementation and verification under a per-round work plan,

Training Loop: Autonomous SFT Optimization

ITERATION LOOP

- 1 Plan & Data Synthesis**
 Actor Agent reads the optimization goal and generates the SFT data recipe and training plan
- 2 Parallel Model Training**
 Search data recipe, learning rate, training config
- 3 Auto-Deploy & Serve**
 Each trained checkpoint is automatically deployed to the inference cluster
- 4 Independent Evaluation**
 Critic Agent runs independent test on each endpoint in parallel
- 5 Verdict, Release & Next Iter**
 Master Agent selects the champion, releases GPUs, derives the next-iter strategy from the observed sweep, and re-enters step 1

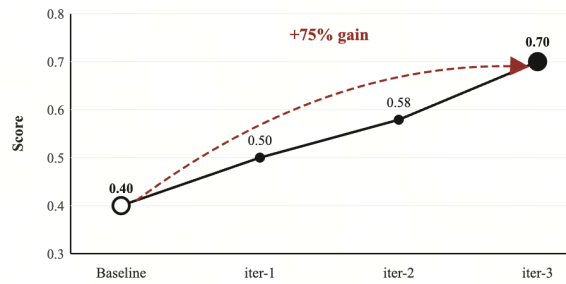
ROLES



PERFORMANCE ACROSS 3 ITER ROUNDS

Gain reported relative to the baseline checkpoint.

Iter	Variants	Score	Gain vs Base
Baseline	1	0.40	—
iter-1	3	0.50	+25%
iter-2	1	0.58	+45%
iter-3	5	0.70	+75%



PIPELINE & COMPLIANCE SUMMARY

Metric	Value
Total iter rounds	3
Total experiments	13
Concurrent forks (max)	5
Pipeline stages per experiment	5 (data → train → deploy → eval → stop)
Best score	0.70 (+75% vs baseline)
Loop wall-clock	4.9 days (autonomous)
Human-in-the-loop	No

Input: optimization goal + priorities **Output:** champion checkpoint + Critic verdict + next-iter strategy **Loop:** plan → fork → deploy → eval → verdict → next iter

5 / 5 VERIFIED · +75%

Figure 20 Training Loop: data synthesis driving SFT improvement.

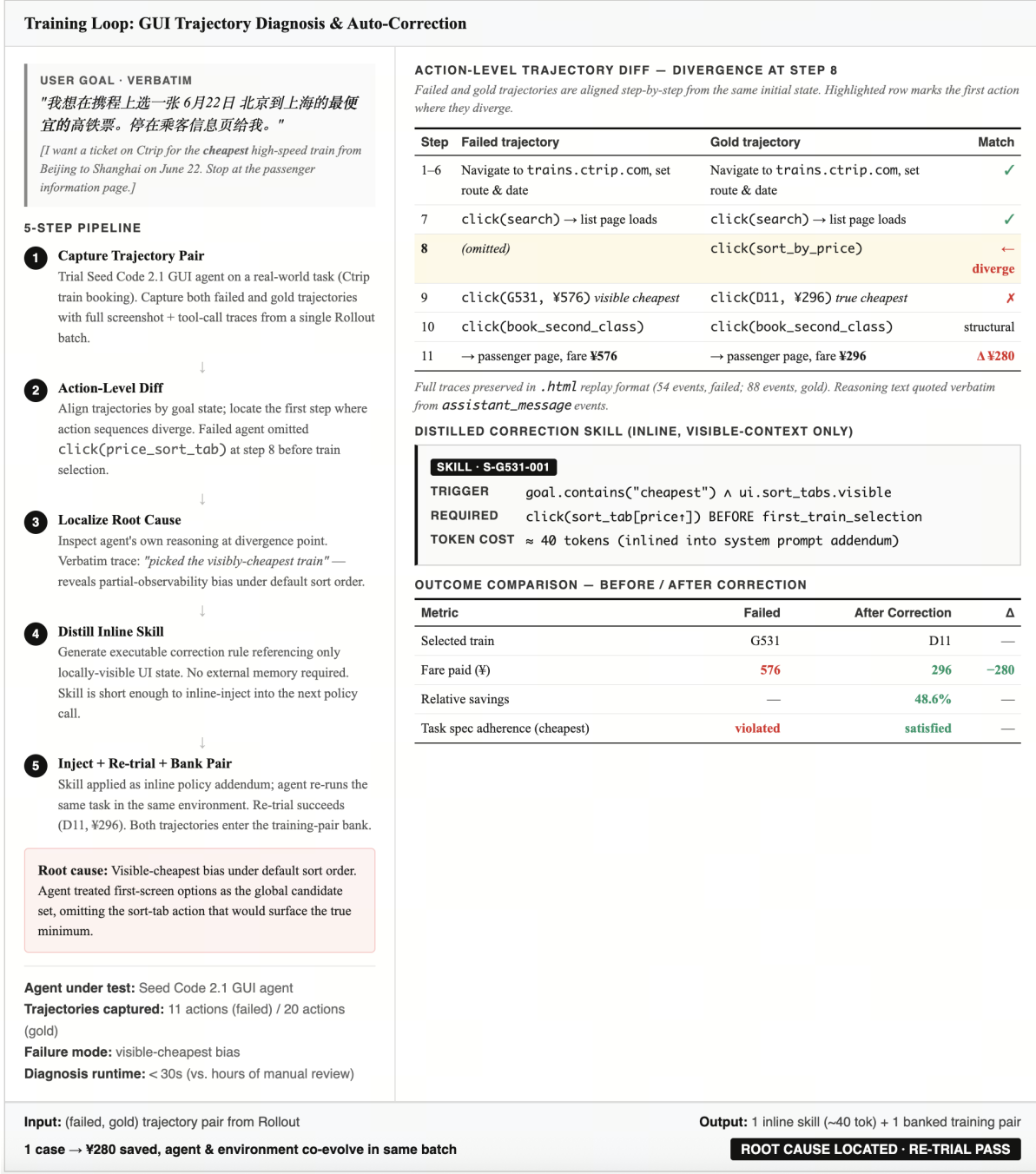


Figure 21 Training Loop: GUI trajectory diagnosis and auto-correction.

dispatching round-scoped sub-agents — branch, fix, and investigation workers — as needed. The **Critic** holds a production-readiness bar strictly stronger than spec compliance. Each round, it dispatches Analyst sub-agents along four axes — code audit, experimental-result interpretation, spec compliance, and production-readiness — against a single source-of-truth requirements file it maintains. Every verdict is anchored in concrete evidence; unverifiable is a first-class outcome. Its deliverable to the Actor is not a score but a remediation packet: what blocks the round, what is rework versus refinement, the minimum next step, and the open questions to resolve before completion. The **Monitor** sits above the inner Actor–Critic loop. It spawns evidence and review sub-agents to ground-truth claims, distills generalized rules (sediment) into NOTES.md so lessons from round N reduce mistakes in round N+1, and — over longer horizons — evolves the Actor–Critic harness itself toward iteratively better performance. At this snapshot the loop has spanned over one month and 600+ rounds of continuous operation. We deliberately do not report a terminal benchmark: by design, the loop has not terminated.

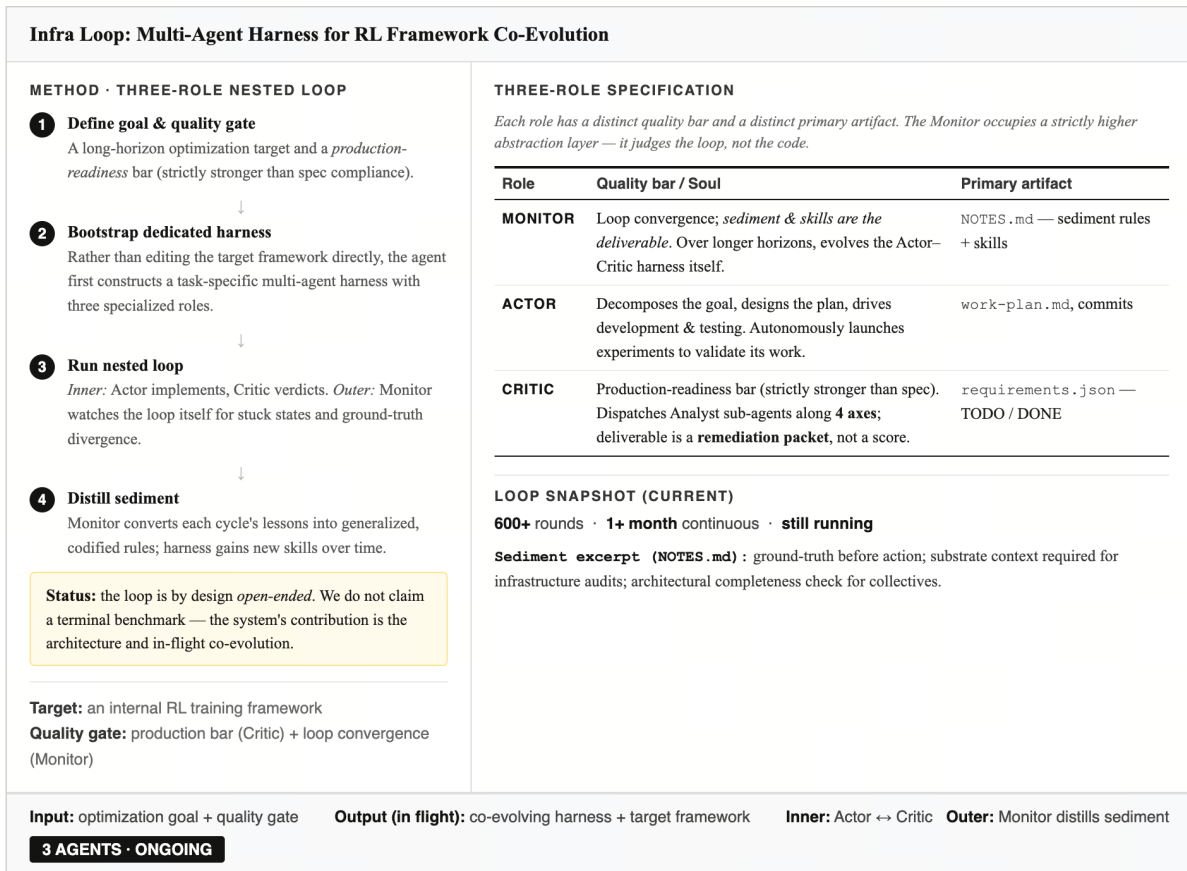


Figure 22 Infra Loop: multi-agent harness for RL framework co-evolution.

Kernel Loop: Agent-Driven GPU Kernel Optimization. Our model excels at agentic GPU kernel generation (Figure 23). Given a PyTorch reference and a target device, it autonomously inspects the hardware, fuses multi-stage operations into a single kernel, and iteratively benchmarks, profiles, and autotunes memory loads, compute precision, warps, and pipeline stages—keeping only what genuinely helps. We find it writes clean, correct Triton from the first attempt and reasons soundly about the memory hierarchy: it recognizes that attention decode is memory-bound, knows when fusion eliminates intermediate HBM traffic, and, just as importantly, it recognizes when fashionable tricks like split-KV or persistent grid-stride kernels actually hurt and confidently reverts them. On a representative paged-attention decode workload for the NVIDIA H800, its first fused Triton kernel—collapsing paged K/V gather, attention, and online softmax into one—already beats the PyTorch baseline by 4.54×, and after autotuning (Hopper TMA loads, BF16 compute, and

warp/pipeline-stage sweeps) it reaches 1.36 ms at 94% MBU—5.90× faster than eager PyTorch and matching the FlashAttention-3 reference. We believe this disciplined, measurement-driven loop is what makes the model a genuinely useful kernel engineer rather than just a code generator.

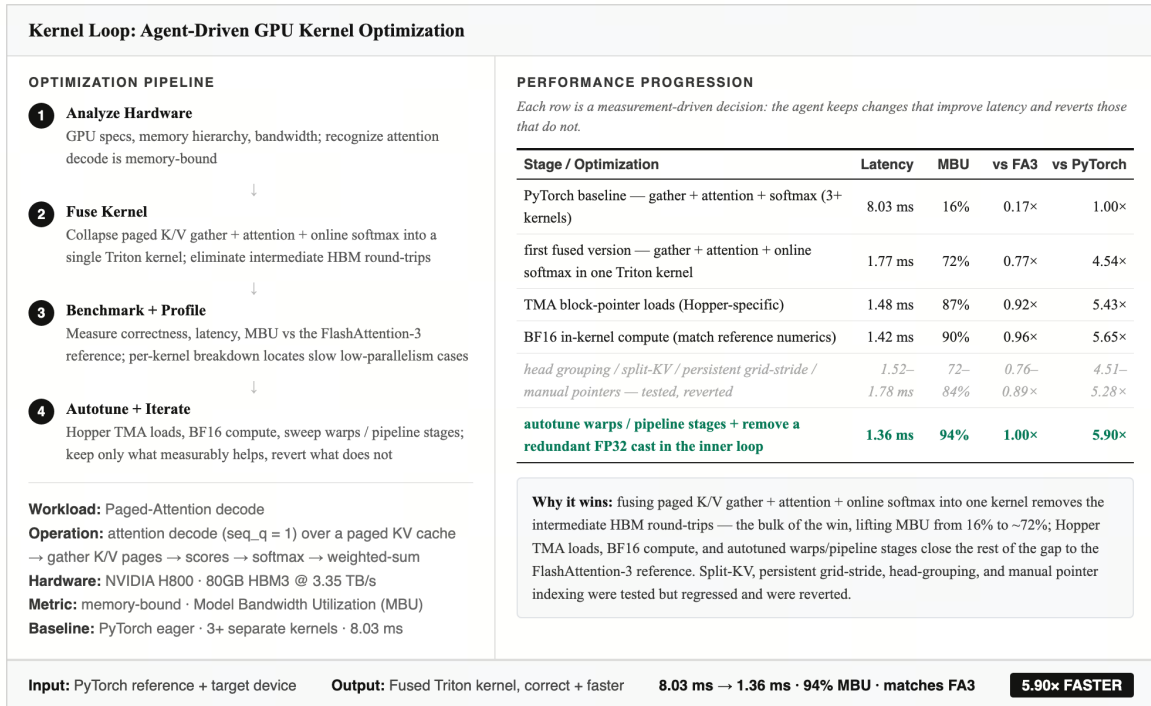


Figure 23 Kernel Loop: agent-driven GPU kernel optimization.

4.4.3 Summary

Overall, Seed for Seed points toward a self-reinforcing development pattern: better models help build better evaluations, cleaner data, stronger training pipelines, and more reliable infrastructure—which in turn produce more capable future models. The showcases above demonstrate that this loop is already operational: Seed2.1 can autonomously execute multi-hour optimization runs, coordinate multi-agent actor-critic workflows, and convert model failures into actionable training signals, all with minimal human intervention beyond goal specification and quality-gate design.

References

- [1] Mislav Balunović, Jasper Dekoninck, Ivo Petrov, Nikola Jovanović, and Martin Vechev. Matharena: Evaluating llms on uncontaminated math competitions. *arXiv preprint arXiv:2505.23281*, 2025.
- [2] Chaithanya Bandi, Ben Hertzberg, Geobio Boo, Tejas Polakam, Jeff Da, Sami Hassaan, Manasi Sharma, Andrew Park, Ernesto Hernandez, Dan Rambado, Ivan Salazar, Rafael Cruz, Chetan Rane, Ben Levin, Brad Kenstler, and Bing Liu. MCP-Atlas: A large-scale benchmark for tool-use competency with real mcp servers, 2026. URL <https://arxiv.org/abs/2602.00933>.
- [3] Antoine Bigeard, Langston Nashold, Rayan Krishnan, and Shirley Wu. Finance agent benchmark: Benchmarking llms on real-world financial research tasks. *arXiv preprint arXiv:2508.00828*, 2025.
- [4] ByteDance-Seed. Beyondaime: Advancing math reasoning evaluation beyond high school olympiads. <https://huggingface.co/datasets/ByteDance-Seed/BeyondAIME>, 2025.

- [5] Meng Cao, Pengfei Hu, Yingyao Wang, Jihao Gu, Haoran Tang, Haoze Zhao, Jiahua Dong, Wangbo Yu, Ge Zhang, Ian Reid, and Xiaodan Liang. Video simpleqa: Towards factuality evaluation in large video language models. *CoRR*, abs/2503.18923, 2025.
- [6] Joya Chen, Ziyun Zeng, Yiqi Lin, Wei Li, Zejun Ma, and Mike Zheng Shou. Livecc: Learning video LLM with streaming speech transcription at scale. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2025, Nashville, TN, USA, June 11-15, 2025*, pages 29083–29095, 2025.
- [7] Liang Chen, Weichu Xie, Yiyan Liang, Hongfeng He, Hans Zhao, Zhibo Yang, Zhiqi Huang, Haoning Wu, Haoyu Lu, Yiping Bao, et al. Babyvision: Visual reasoning beyond language. *arXiv preprint arXiv:2601.06521*, 2026.
- [8] Xin-Sheng Chen, Jiayu Zhu, Pei-lin Li, Hanzheng Wang, Shuojin Yang, and Meng-Hao Guo. PresentBench: A fine-grained rubric-based benchmark for slide generation, 2026. URL <https://arxiv.org/abs/2603.07244>.
- [9] Junhao Cheng, Yuying Ge, Teng Wang, Yixiao Ge, Jing Liao, and Ying Shan. Video-holmes: Can MLLM think like holmes for complex video reasoning? *CoRR*, abs/2505.21374, 2025.
- [10] Xianfu Cheng, Wei Zhang, Shiwei Zhang, Jian Yang, Xiangyuan Guan, Xianjie Wu, Xiang Li, Ge Zhang, Jiaheng Liu, Yuying Mai, et al. Simplevqa: Multimodal factuality evaluation for multimodal large language models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4637–4646, 2025.
- [11] Daniel Cores, Michael Dorckenwald, Manuel Mucientes, Cees G. M. Snoek, and Yuki M. Asano. Tvbench: Redesigning video-language evaluation. *CoRR*, abs/2410.07752, 2024.
- [12] Song Dai, Liuwei Gong, and Shaoming Guo. The local geometry of oscillatory integrals on manifolds: Dimension three, 2026. URL <https://arxiv.org/abs/2606.12927>.
- [13] Xiang Deng, Jeff Da, Edwin Pan, Yannis Yiming He, Charles Ide, Kanak Garg, Niklas Lauffer, Andrew Park, Nitin Pasari, Chetan Rane, et al. Swe-bench pro: Can ai agents solve long-horizon software engineering tasks? *arXiv preprint arXiv:2509.16941*, 2025.
- [14] Kaustubh Deshpande, Ved Sirdeshmukh, Johannes Baptist Mols, Lifeng Jin, Ed-Yeremai Hernandez-Cardona, Dean Lee, Jeremy Kritz, Willow E Primack, Summer Yue, and Chen Xing. Multichallenge: A realistic multi-turn conversation evaluation benchmark challenging to frontier llms. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 18632–18702, 2025.
- [15] Jingzhe Ding, Shengda Long, Changxin Pu, Huan Zhou, Hongwan Gao, Xiang Gao, Chao He, Yue Hou, Fei Hu, Zhaojian Li, et al. Nl2repo-bench: Towards long-horizon repository generation evaluation of coding agents. *arXiv preprint arXiv:2512.12730*, 2025.
- [16] Shuangrui Ding, Xuanlang Dai, Long Xing, Shengyuan Ding, Ziyu Liu, Yang JingYi, Penghui Yang, Zhixiong Zhang, Xilin Wei, Xinyu Fang, et al. Wildclawbench: A benchmark for real-world, long-horizon agent evaluation. *arXiv preprint arXiv:2605.10912*, 2026.
- [17] Guanting Dong, Junting Lu, Junjie Huang, Wanjun Zhong, Longxiang Liu, Shijue Huang, Zhenyu Li, Yang Zhao, Xiaoshuai Song, Xiaoxi Li, Jiajie Jin, Yutao Zhu, Hanbin Wang, Fangyu Lei, Qinyu Luo, Mingyang Chen, Zehui Chen, Jiazhan Feng, Ji-Rong Wen, and Zhicheng Dou. Agent-world: Scaling real-world environment synthesis for evolving general agent intelligence, 2026. URL <https://arxiv.org/abs/2604.18292>.
- [18] Mengfei Du, Binhao Wu, Zejun Li, Xuan-Jing Huang, and Zhongyu Wei. Embspatial-bench: Benchmarking spatial understanding for embodied tasks with large vision-language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 346–355, 2024.
- [19] Xinrun Du, Yifan Yao, Kaijing Ma, Bingli Wang, Tianyu Zheng, King Zhu, Minghao Liu, Yiming Liang, Xiaolong Jin, Zhenlin Wei, et al. Supergpqa: Scaling llm evaluation across 285 graduate disciplines. *arXiv preprint arXiv:2502.14739*, 2025.
- [20] Chaoyou Fu, Yuhan Dai, Yongdong Luo, Lei Li, Shuhuai Ren, Renrui Zhang, Zihan Wang, Chenyu Zhou, Yunhang Shen, Mengdan Zhang, Peixian Chen, Yanwei Li, Shaohui Lin, Sirui Zhao, Ke Li, Tong Xu, Xiawu Zheng, Enhong Chen, Caifeng Shan, Ran He, and Xing Sun. Video-mme: The first-ever comprehensive evaluation benchmark of multi-modal llms in video analysis. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2025, Nashville, TN, USA, June 11-15, 2025*, pages 24108–24118, 2025.
- [21] Ling Fu, Zhebin Kuang, Jiajun Song, Mingxin Huang, Biao Yang, Yuzhe Li, Linghao Zhu, Qidi Luo, Xinyu Wang, Hao Lu, et al. Ocrbench v2: An improved benchmark for evaluating large multimodal models on visual text localization and reasoning. *arXiv preprint arXiv:2501.00321*, 2024.

- [22] Xingyu Fu, Yushi Hu, Bangzheng Li, Yu Feng, Haoyu Wang, Xudong Lin, Dan Roth, Noah A Smith, Wei-Chiu Ma, and Ranjay Krishna. Blink: Multimodal large language models can see but not perceive. In *European Conference on Computer Vision*, pages 148–166. Springer, 2024.
- [23] Yunzhuo Hao, Jiawei Gu, Huichen Will Wang, Linjie Li, Zhengyuan Yang, Lijuan Wang, and Yu Cheng. Can mllms reason in multimodality? emma: An enhanced multimodal reasoning benchmark. *arXiv preprint arXiv:2501.05444*, 2025.
- [24] Linyang He, Qiyao Yu, Hanze Dong, Baohao Liao, Xinxing Xu, Micah Goldblum, Jiang Bian, and Nima Mesgarani. LiveMathematicianBench: A Live Benchmark for Mathematician-Level Reasoning with Proof Sketches. *arXiv e-prints*, art. arXiv:2604.01754, April 2026. doi: 10.48550/arXiv.2604.01754.
- [25] Wenyi Hong, Yean Cheng, Zhuoyi Yang, Weihang Wang, Lefan Wang, Xiaotao Gu, Shiyu Huang, Yuxiao Dong, and Jie Tang. Motionbench: Benchmarking and improving fine-grained video motion understanding for vision language models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2025, Nashville, TN, USA, June 11-15, 2025*, pages 8450–8460, 2025.
- [26] Jen-Tse Huang, Dasen Dai, Jen-Yuan Huang, Youliang Yuan, Xiaoyuan Liu, Wenxuan Wang, Wenxiang Jiao, Pinjia He, and Zhaopeng Tu. Visfactor: Benchmarking fundamental visual cognition in multimodal large language models. *arXiv preprint arXiv:2502.16435*, 2025.
- [27] Zhenpeng Huang, Xinhao Li, Jiaqi Li, Jing Wang, Xiangyu Zeng, Cheng Liang, Tao Wu, Xi Chen, Liang Li, and Limin Wang. Online video understanding: Ovbench and videochat-online. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2025, Nashville, TN, USA, June 11-15, 2025*, pages 3328–3338, 2025.
- [28] Sheng Jin, Minghao Liu, Yunze Xiao, Zeqi Zhou, Heli Qi, Yifan Yao, Meishu Song, Kaijing Ma, Xuan Zhang, Sicong Jiang, et al. Knowledge index of noah’s ark, 2026. URL <https://arxiv.org/abs/2606.05104>.
- [29] Quyu Kong, Xu Zhang, Zhenyu Yang, Nolan Gao, Chen Liu, Panrong Tong, Chenglin Cai, Hanzhang Zhou, Jianan Zhang, Liangyu Chen, Zhidan Liu, Steven Hoi, and Yue Wang. MobileWorld: Benchmarking autonomous mobile agents in agent-user interactive, and mcp-augmented environments, 2025. URL <https://arxiv.org/abs/2512.19432>.
- [30] Jingyao Li, Jingyun Wang, Molin Tan, Haochen Wang, Cilin Yan, Likun Shi, Jiayin Cai, Xiaolong Jiang, and Yao Hu. Crossvid: A comprehensive benchmark for evaluating cross-video reasoning in multimodal large language models. *CoRR*, abs/2511.12263, 2025.
- [31] Junlong Li, Wenshuo Zhao, Jian Zhao, Weihao Zeng, Haoze Wu, Xiaochen Wang, Rui Ge, Yuxuan Cao, Yuzhen Huang, Wei Liu, Junteng Liu, Zhaochen Su, Yiyang Guo, Fan Zhou, Lueyang Zhang, Juan Michelini, Xingyao Wang, Xiang Yue, Shuyan Zhou, Graham Neubig, and Junxian He. The tool decathlon: Benchmarking language agents for diverse, realistic, and long-horizon task execution, 2025. URL <https://arxiv.org/abs/2510.25726>.
- [32] Xiangyi Li, Yimin Liu, Wenbo Chen, Bingran You, Zonglin Di, Yifeng He, Shenghan Zheng, Kyoung Whan Choe, Jiankai Sun, Shuyi Wang, Chujuan Tao, Binxu Li, Xuandong Zhao, Hejia Geng, Xiaojun Wu, Junwei Zhou, Xiaokun Chen, Hanwen Xing, Yubo Li, Qunhong Zeng, Di Wang, Yuanli Wang, Roey Ben Chaim, et al. SkillsBench: Benchmarking how well agent skills work across diverse tasks, 2026. URL <https://arxiv.org/abs/2602.12670>.
- [33] Fenfen Lin, Yesheng Liu, Haiyu Xu, Yue Chen, Zheqi He, Mingxuan Zhao, Miguel Hu Chen, Jin-Ge Yao, and Xi Yang. Do vision-language models measure up? benchmarking visual measurement reading with measurebench. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 38544–38553, 2026.
- [34] Pan Lu, Hritik Bansal, Tony Xia, Jiacheng Liu, Chunyuan Li, Hannaneh Hajishirzi, Hao Cheng, Kai-Wei Chang, Michel Galley, and Jianfeng Gao. Mathvista: Evaluating mathematical reasoning of foundation models in visual contexts. *arXiv preprint arXiv:2310.02255*, 2023.
- [35] Minh-Thang Luong, Dawsen Hwang, Hoang H Nguyen, Golnaz Ghiasi, Yuri Chervonyi, Insuk Seo, Junsu Kim, Garrett Bingham, Jonathan Lee, Swaroop Mishra, et al. Towards robust mathematical reasoning. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 35406–35430, 2025.
- [36] Qiuyang Mang, Wenhao Chai, Zhifei Li, Huanzhi Mao, Shang Zhou, Alexander Du, Hanchen Li, Shu Liu, Edwin Chen, Yichuan Wang, Xieting Chu, Zerui Cheng, Yuan Xu, Tian Xia, Zirui Wang, Tianneng Shi, Jianzhu Yao, Yilong Zhao, Qizheng Zhang, Charlie Ruan, Zeyu Shen, Kaiyuan Liu, Runyuan He, Dong Xing, Zerui Li, Zirong Zeng, Yige Jiang, Lufeng Cheng, Ziyi Zhao, Youran Sun, Wesley Zheng, Meiyuwang Zhang, Ruyi Ji, Xuechang

- Tu, Zihan Zheng, Zexing Chen, Kangyang Zhou, Zhaozi Wang, Jingbang Chen, Aleksandra Korolova, Peter Henderson, Pramod Viswanath, Vijay Ganesh, Saining Xie, Zhuang Liu, Dawn Song, Sewon Min, Ion Stoica, Joseph E. Gonzalez, Jingbo Shang, and Alvin Cheung. FrontierCS: Evolving challenges for evolving intelligence, 2025. URL <https://arxiv.org/abs/2512.15699>.
- [37] Ahmed Masry, Mohammed Saidul Islam, Mahir Ahmed, Aayush Bajaj, Firoz Kabir, Aaryaman Kartha, Md Tahmid Rahman Laskar, Mizanur Rahman, Shadikur Rahman, Mehrad Shahmohammadi, et al. Chartqapro: A more diverse and challenging benchmark for chart question answering. *arXiv preprint arXiv:2504.05506*, 2025.
- [38] Mike A. Merrill, Alexander G. Shaw, Nicholas Carlini, Boxuan Li, Harsh Raj, Ivan Bercovich, Lin Shi, Jeong Yeon Shin, Thomas Walshe, E. Kelly Buchanan, Junhong Shen, Guanghao Ye, Haowei Lin, Jason Poulos, Maoyu Wang, Marianna Nezhurina, Jenia Jitsev, Di Lu, Orfeas Menis Mastromichalakis, Zhiwei Xu, Zizhao Chen, Yue Liu, Robert Zhang, Leon Liangyu Chen, Anurag Kashyap, Jan-Lucas Uslu, Jeffrey Li, Jianbo Wu, Minghao Yan, Song Bian, Vedang Sharma, Ke Sun, Steven Dillmann, Akshay Anand, Andrew Lanpouthakoun, Bardia Koopah, Changran Hu, Etash Guha, Gabriel H. S. Dreiman, Jiacheng Zhu, Karl Krauth, Li Zhong, Niklas Muennighoff, Robert Amanfu, Shangyin Tan, Shreyas Pimpalgaonkar, Tushar Aggarwal, Xiangning Lin, Xin Lan, Xuandong Zhao, Yiqing Liang, Yuanli Wang, Zilong Wang, Changzhi Zhou, David Heineman, Hange Liu, Harsh Trivedi, John Yang, Junhong Lin, Manish Shetty, Michael Yang, Nabil Omi, Negin Raoof, Shanda Li, Terry Yue Zhuo, Wuwei Lin, Yiwei Dai, Yuxin Wang, Wenhao Chai, Shang Zhou, Dariush Wahdany, Ziyu She, Jiaming Hu, Zhikang Dong, Yuxuan Zhu, Sasha Cui, Ahson Saiyed, Arinbjörn Kolbeinsson, Jesse Hu, Christopher Michael Rytting, Ryan Marten, Yixin Wang, Alex Dimakis, Andy Konwinski, and Ludwig Schmidt. Terminal-bench: Benchmarking agents on hard, realistic tasks in command line interfaces, 2026. URL <https://arxiv.org/abs/2601.11868>.
- [39] Arsha Nagrani, Sachit Menon, Ahmet Iscen, Shyamal Buch, Ramin Mehran, Nilpa Jha, Anja Hauth, Yukun Zhu, Carl Vondrick, Mikhail Sirotenko, Cordelia Schmid, and Tobias Weyand. MINERVA: evaluating complex video reasoning. *CoRR*, abs/2505.00681, 2025.
- [40] Junbo Niu, Yifei Li, Ziyang Miao, Chunjiang Ge, Yuanhang Zhou, Qihao He, Xiaoyi Dong, Haodong Duan, Shuangrui Ding, Rui Qian, Pan Zhang, Yuhang Zang, Yuhang Cao, Conghui He, and Jiaqi Wang. Ovo-bench: How far is your video-llms from real-world online video understanding? In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2025, Nashville, TN, USA, June 11-15, 2025*, pages 18902–18913, 2025.
- [41] Krista Opsahl-Ong, Arnav Singhvi, Jasmine Collins, Ivan Zhou, Cindy Wang, Ashutosh Baheti, Owen Oertell, Jacob Portes, Sam Havens, Erich Elsen, Michael Bendersky, Matei Zaharia, and Xing Chen. OfficeQA Pro: An enterprise benchmark for end-to-end grounded reasoning, 2026. URL <https://arxiv.org/abs/2603.08655>.
- [42] Mingyu Ouyang, Siyuan Hu, Kevin Qinghong Lin, Hwee Tou Ng, and Mike Zheng Shou. GameWorld: Towards standardized and verifiable evaluation of multimodal game agents, 2026. URL <https://arxiv.org/abs/2604.07429>.
- [43] Tejal Patwardhan, Rachel Dias, Elizabeth Proehl, Grace Kim, Michele Wang, Olivia Watkins, Simón Posada Fishman, Marwan Aljubei, Phoebe Thacker, Laurance Fauconnet, Natalie S. Kim, Patrick Chao, Samuel Misereindino, Gildas Chabot, David Li, Michael Sharman, Alexandra Barr, Amelia Glaese, and Jerry Tworek. GDPval: Evaluating ai model performance on real-world economically valuable tasks, 2025. URL <https://arxiv.org/abs/2510.04374>.
- [44] Long Phan, Alice Gatti, Ziwen Han, Nathaniel Li, Josephina Hu, Hugh Zhang, Chen Bo Calvin Zhang, Mohamed Shaaban, John Ling, Sean Shi, et al. Humanity’s last exam. *arXiv preprint arXiv:2501.14249*, 2025.
- [45] ARC Prize. Arc agi: The \$1 million artificial general intelligence prize. <https://arcprize.org/arc-agi/1/>, 2024.
- [46] Mohit Raghavendra, Soham Dan, Miguel Romero Calvo, Yannis Yiming He, Johannes Baptist Mols, Gautam Anand, Cole McCollum, Edgar Arakelyan, Vijay Bharadwaj, Andrew Park, Jeff Da, MohammadHossein Rezaei, Bing Liu, Brad Kenstler, and Yunzhong He. SWE Atlas: Benchmarking Coding Agents Beyond Issue Resolution. *arXiv e-prints*, art. arXiv:2605.08366, May 2026. doi: 10.48550/arXiv.2605.08366.
- [47] Ben Rank, Hardik Bhatnagar, Ameya Prabhu, Shira Eisenberg, Karina Nguyen, Matthias Bethge, and Maksym Andriushchenko. PostTrainBench: Can llm agents automate llm post-training?, 2026. URL <https://arxiv.org/abs/2603.08640>.
- [48] Jonathan Roberts, Mohammad Reza Taesiri, Ansh Sharma, Akash Gupta, Samuel Roberts, Ioana Croitoru, Simion-Vlad Bogolin, Jialu Tang, Florian Langer, Vyas Raina, et al. Zerobench: An impossible visual benchmark for contemporary large multimodal models. *arXiv preprint arXiv:2502.09696*, 2025.

- [49] Ziyao Shangguan, Chuhan Li, Yuxuan Ding, Yanan Zheng, Yilun Zhao, Tesca Fitzgerald, and Arman Cohan. TOMATO: assessing visual temporal reasoning capabilities in multimodal foundation models. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*, 2025.
- [50] Yiyu Sun, Xinyang Han, Weichen Zhang, Yuanbo Pang, Tianyu Wang, Yuhan Cao, Yixiao Huang, Chris Duroiu, Haoyun Zhang, Jeffrey Lin, et al. Agents’ last exam, 2026. URL <https://arxiv.org/abs/2606.05405>.
- [51] Zirui Tang, Xuanhe Zhou, Yumou Liu, Linchun Li, Weizheng Wang, Hongzhang Huang, Jun Zhou, Jiachen Song, Shaoli Yu, Jinqi Wang, Zihang Zhou, Hongyi Zhou, Yuting Lv, Jinyang Li, Jiashuo Liu, Ruoyu Chen, Chunwei Liu, Guoliang Li, Jihua Kang, and Fan Wu. Workspace-Bench 1.0: Benchmarking ai agents on workspace tasks with large-scale file dependencies, 2026. URL <https://arxiv.org/abs/2605.03596>.
- [52] DeepSWE Team. DeepSWE: Measuring frontier coding agents on original, long-horizon engineering tasks. <https://github.com/datacurve-ai/deep-swe>, 2026. GitHub repository.
- [53] Gemini Robotics Team, Saminda Abeyruwan, Joshua Ainslie, Jean-Baptiste Alayrac, Montserrat Gonzalez Arenas, Travis Armstrong, Ashwin Balakrishna, Robert Baruch, Maria Bauza, Michiel Blokzijl, et al. Gemini robotics: Bringing ai into the physical world. *arXiv preprint arXiv:2503.20020*, 2025.
- [54] Terminal-Bench Team. Introducing terminal-bench 2.0 and harbor. <https://www.tbench.ai/news/announcement-2-0>, nov 2025. Accessed: 2025-12-10.
- [55] Minyang Tian, Luyu Gao, Shizhuo Dylan Zhang, Xinan Chen, Cunwei Fan, Xuefei Guo, Roland Haas, Pan Ji, Kittithat Krongchon, Yao Li, Shengyan Liu, Di Luo, Yutao Ma, Hao Tong, Kha Trinh, Chenyu Tian, Zihan Wang, Bohao Wu, Yanyu Xiong, Shengzhu Yin, Minhui Zhu, Kilian Lieret, Yanxin Lu, Genglin Liu, Yufeng Du, Tianhua Tao, Ofir Press, Jamie Callan, Eliu Huerta, and Hao Peng. SciCode: A Research Coding Benchmark Curated by Scientists. *arXiv e-prints*, art. arXiv:2407.13168, July 2024. doi: 10.48550/arXiv.2407.13168.
- [56] Jordy Van Landeghem, Rubèn Tito, Łukasz Borchmann, Michał Pietruszka, Paweł Joziak, Rafał Powalski, Dawid Jurkiewicz, Mickaël Coustaty, Bertrand Anckaert, Ernest Valveny, et al. Document understanding dataset and evaluation (dude). In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 19528–19540, 2023.
- [57] Bertie Vidgen, Austin Mann, Abby Fennelly, John Wright Stanly, Lucas Rothman, Marco Burstein, Julien Benček, David Ostrofsky, Anirudh Ravichandran, Debnil Sur, Neel Venugopal, Alannah Hsia, Isaac Robinson, Calix Huang, Olivia Varones, Daniyal Khan, Michael Haines, Zach Richards, Chirag Mahapatra, Brendan Foody, and Osvald Nitski. APEX-Agents, 2026. URL <https://arxiv.org/abs/2601.14242>.
- [58] An Vo, Khai-Nguyen Nguyen, Mohammad Reza Taesiri, Vy Tuong Dang, Anh Totti Nguyen, and Daeyoung Kim. Vision language models are biased. *arXiv preprint arXiv:2505.23941*, 2025.
- [59] Erik Y. Wang, Sumeet Motwani, James V. Roggeveen, Eliot Hodges, Dulhan Jayalath, Charles London, Kalyan Ramakrishnan, Flaviu Cipcigan, Philip Torr, and Alessandro Abate. HorizonMath: Measuring ai progress toward mathematical discovery with automatic verification, 2026. URL <https://arxiv.org/abs/2603.15617>.
- [60] Haochen Wang, Xiangtai Li, Zilong Huang, Anran Wang, Jiacong Wang, Tao Zhang, Jiani Zheng, Sule Bai, Zijian Kang, Jiashi Feng, et al. Traceable evidence enhanced visual grounded reasoning: Evaluation and methodology. *arXiv preprint arXiv:2507.07999*, 2025.
- [61] Ke Wang, Junting Pan, Weikang Shi, Zimu Lu, Houxing Ren, Aojun Zhou, Mingjie Zhan, and Hongsheng Li. Measuring multimodal mathematical reasoning with math-vision dataset. *Advances in Neural Information Processing Systems*, 37:95095–95169, 2024.
- [62] Miles Wang, Robi Lin, Kat Hu, Joy Jiao, Neil Chowdhury, Ethan Chang, and Tejal Patwardhan. Frontierscience: Evaluating ai’s ability to perform expert-level scientific tasks, 2026. URL <https://arxiv.org/abs/2601.21165>.
- [63] Weihang Wang, Zehai He, Wenyi Hong, Yean Cheng, Xiaohan Zhang, Ji Qi, Shiyu Huang, Bin Xu, Yuxiao Dong, Ming Ding, and Jie Tang. Lvbench: An extreme long video understanding benchmark. *CoRR*, abs/2406.08035, 2024.
- [64] Zhaowei Wang, Wenhao Yu, Xiyu Ren, Jipeng Zhang, Yu Zhao, Rohit Saxena, Liang Cheng, Ginny Wong, Simon See, Pasquale Minervini, et al. Mmlongbench: Benchmarking long-context vision-language models effectively and thoroughly. *arXiv preprint arXiv:2505.10610*, 2025.

- [65] Zhun Wang, Tianneng Shi, Jingxuan He, Matthew Cai, Jialin Zhang, and Dawn Song. CyberGym: Evaluating ai agents’ cybersecurity capabilities with real-world vulnerabilities at scale, 2025. URL <https://arxiv.org/abs/2506.02548>.
- [66] Zirui Wang, Mengzhou Xia, Luxi He, Howard Chen, Yitao Liu, Richard Zhu, Kaiqu Liang, Xindi Wu, Haotian Liu, Sadhika Malladi, et al. Charxiv: Charting gaps in realistic chart understanding in multimodal llms. *Advances in Neural Information Processing Systems*, 37:113569–113697, 2024.
- [67] Jason Wei, Zhiqing Sun, Spencer Papay, Scott McKinney, Jeffrey Han, Isa Fulford, Hyung Won Chung, Alex Tachard Passos, William Fedus, and Amelia Glaese. Browsecomp: A simple yet challenging benchmark for browsing agents. *arXiv preprint arXiv:2504.12516*, 2025.
- [68] Haoning Wu, Dongxu Li, Bei Chen, and Junnan Li. Longvideobench: A benchmark for long-context interleaved video-language understanding. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024.
- [69] XAI. Realworldqa. URL <https://huggingface.co/datasets/xai-org/RealworldQA>.
- [70] Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh Jing Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, Yitao Liu, Yiheng Xu, Shuyan Zhou, Silvio Savarese, Caiming Xiong, Victor Zhong, and Tao Yu. OSWorld: Benchmarking multimodal agents for open-ended tasks in real computer environments, 2024. URL <https://arxiv.org/abs/2404.07972>.
- [71] Weiye Xu, Jiahao Wang, Weiyun Wang, Zhe Chen, Wengang Zhou, Aijun Yang, Lewei Lu, Houqiang Li, Xiaohua Wang, Xizhou Zhu, et al. Visulogic: A benchmark for evaluating visual reasoning in multi-modal large language models. *arXiv preprint arXiv:2504.15279*, 2025.
- [72] Chenghao Yang, Yinbo Luo, Zhoufutu Wen, Qi Chu, Tao Gong, Longxiang Liu, Kaiyuan Zhang, Jianpeng Jiao, Ge Zhang, Wenhao Huang, et al. Mars-bench: A multi-turn athletic real-world scenario benchmark for dialogue evaluation. *arXiv preprint arXiv:2505.23810*, 2025.
- [73] John Yang, Kilian Lieret, Jeffrey Ma, Parth Thakkar, Dmitrii Pedchenko, Sten Sootla, Emily McMilin, Pengcheng Yin, Rui Hou, Gabriel Synnaeve, Diyi Yang, and Ofir Press. ProgramBench: Can language models rebuild programs from scratch?, 2026. URL <https://arxiv.org/abs/2605.03546>.
- [74] Qianyu Yang, Yang Liu, Jiaqi Li, Jun Bai, Hao Chen, Kaiyuan Chen, Tiliang Duan, Jiayun Dong, Xiaobo Hu, Zixia Jia, Yang Liu, Tao Peng, Yixin Ren, Ran Tian, Zaiyuan Wang, Yanglihong Xiao, Gang Yao, Lingyue Yin, Ge Zhang, Chun Zhang, Jianpeng Jiao, Zilong Zheng, and Yuan Gong. OneMillion-Bench: How far are language agents from human experts?, 2026. URL <https://arxiv.org/abs/2603.07980>.
- [75] Sihan Yang, Runsen Xu, Yiman Xie, Sizhe Yang, Mo Li, Jingli Lin, Chenming Zhu, Xiaochen Chen, Haodong Duan, Xiangyu Yue, et al. Mmsi-bench: A benchmark for multi-image spatial intelligence. *arXiv preprint arXiv:2505.23764*, 2025.
- [76] Bowen Ye, Rang Li, Qibin Yang, Yuanxin Liu, Linli Yao, Hanglong Lv, Zhihui Xie, Chenxin An, Lei Li, Lingpeng Kong, Qi Liu, Zhifang Sui, and Tong Yang. Claw-Eval: Towards trustworthy evaluation of autonomous agents, 2026. URL <https://arxiv.org/abs/2604.06132>.
- [77] Yida Yin, Harish Krishnakumar, Chung Peng Lee, Boya Zeng, Wenhao Chai, Shengbang Tong, Wenhao Chen, Hu Xu, Xingyu Fu, Gabriel Sarch, et al. Worldbench: A challenging and visually diverse multimodal reasoning benchmark. *arXiv preprint arXiv:2606.06538*, 2026.
- [78] Xiang Yue, Tianyu Zheng, Yuansheng Ni, Yubo Wang, Kai Zhang, Shengbang Tong, Yuxuan Sun, Botao Yu, Ge Zhang, Huan Sun, et al. Mmmu-pro: A more robust multi-discipline multimodal understanding benchmark. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15134–15186, 2025.
- [79] Renrui Zhang, Dongzhi Jiang, Yichi Zhang, Haokun Lin, Ziyu Guo, Pengshuo Qiu, Aojun Zhou, Pan Lu, Kai-Wei Chang, Yu Qiao, et al. Mathverse: Does your multi-modal llm truly see the diagrams in visual math problems? In *European Conference on Computer Vision*, pages 169–186. Springer, 2024.
- [80] Zehua Zhao, Zhixian Huang, Junren Li, Siyu Lin, Junting Zhou, Fengqi Cao, Kun Zhou, Rui Ge, Tingting Long, Yuexiang Zhu, Yan Liu, Jie Zheng, Junnian Wei, Rong Zhu, Peng Zou, Wenyu Li, Zekai Cheng, Tian Ding, Yaxuan Wang, Yizhao Yan, Tingru Wei, Haowei Ming, Weijie Mao, Chen Sun, Yiming Liu, Zichen Wang, Zuo

- Zhang, Tong Yang, Hao Ma, Zhen Gao, and Jian Pei. SUPERChem: A Multimodal Reasoning Benchmark in Chemistry. *arXiv e-prints*, art. arXiv:2512.01274, December 2025. doi: 10.48550/arXiv.2512.01274.
- [81] Zhicheng Zheng, Xin Yan, Zhenfang Chen, Jingzhou Wang, Qin Zhi Eddie Lim, Joshua B. Tenenbaum, and Chuang Gan. Contphy: Continuum physical concept learning and reasoning from videos. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*, volume 235 of *Proceedings of Machine Learning Research*, pages 61526–61558, 2024.
- [82] Runjie Zhou, Youbo Shao, Haoyu Lu, Bowei Xing, Tongtong Bai, Yujie Chen, Jie Zhao, Lin Sui, Haotian Yao, Zijia Zhao, et al. Worldvqa: Measuring atomic world knowledge in multimodal large language models. *arXiv preprint arXiv:2602.02537*, 2026.
- [83] Chiwei Zhu, Benfeng Xu, Mingxuan Du, Shaohan Wang, Xiaorui Wang, Zhendong Mao, and Yongdong Zhang. FS-Researcher: Test-Time Scaling for Long-Horizon Research Tasks with File-System-Based Agents. *arXiv e-prints*, art. arXiv:2602.01566, February 2026. doi: 10.48550/arXiv.2602.01566.
- [84] Chengke Zou, Xingang Guo, Rui Yang, Junyu Zhang, Bin Hu, and Huan Zhang. Dynamath: A dynamic visual benchmark for evaluating mathematical reasoning robustness of vision language models. *arXiv preprint arXiv:2411.00836*, 2024.

A Representative Trae Agent Bench Examples

This appendix presents representative examples from Trae Agent Bench. Long prompts, code snippets, and stack traces are shortened with ellipses for readability, while preserving the original task structure, comments, and implementation hints.

Example 1: JavaScript Error Fixing

Subset: `error_fix_js_v1_3`. **Total cases:** 170.

Prompt. Please fix the bug in the project according to the information below.

Issue description. The custom time input requires two clicks before it receives focus.

Bug description. When the user clicks the time input for the first time, it does not receive focus. The input only receives focus after a second click.

Reproduction steps.

- Go to the custom time-input example in the date picker demo.
- Click the date-picker input to open the picker.
- Click the time input.
- The time input does not receive focus.
- Click the time input again.
- The time input receives focus.

Expected behavior. The time input should receive focus on the first click.

Environment. Windows 11; Firefox Developer Edition 128.0b7 and Chrome 126.0.6478.127.

Additional context. The issue appeared after upgrading from `react-datepicker` 4.17.0 to 7.2.0. The previous version worked correctly.

Comments.

- A contributor volunteered to take the issue.
- A follow-up comment indicates that a pull request had been opened.
- The proposed fix makes the custom time input focus correctly on the first click.

Example 2: Python Error Fixing

Subset: `error_fix_python_v1_3`. **Total cases:** 100.

Prompt. Please fix the bug in the project according to the information below.

Issue description. `pt.max` is not differentiable inside a PyMC model.

Description. The following call to `model.dlogp()` raises a `NotImplementedError`:

```
with pm.Model() as model:
    x = pm.Normal("x", shape=(2,))
    mu = x.max()
    pm.Normal("obs", mu, observed=np.random.uniform())

model.dlogp()
```

This should be differentiable, because `pt.max` already implements gradients. The issue may be related to graph rewrites, log-probability rewrites, or `MaxAndArgmax`.

Comments.

- `min` triggers the same `NotImplementedError`.
- This is expected because `min` can be implemented as the negative of a maximum over negated val-

ues.

- A commenter identifies the issue as related to gradient handling during graph rewrites.
- The maintainers note that this issue is probably not suitable for a first-time contributor because it requires deeper understanding of rewrite and gradient logic.

Example 3: Java Error Fixing

Subset: error_fix_java_v1_2. **Total cases:** 60.

Prompt. Please fix the bug in the project according to the information below.

Issue description. When parsing a generic method with an array type parameter, such as `List<T[]>`, the parser raises `java.lang.UnsupportedOperationException: T[]`.

Versions. javaparser-core: 3.25.0. javaparser-symbol-solver-core: 3.25.0.

Test case.

```
import com.github.javaparser.StaticJavaParser;
import com.github.javaparser.ast.CompilationUnit;
import com.github.javaparser.ast.expr.MethodCallExpr;
import com.github.javaparser.symbolsolver.JavaSymbolSolver;
...

public class TypeParamIsArray {
    public static void main(String[] args) {
        CombinedTypeSolver combinedTypeSolver = new CombinedTypeSolver();
        combinedTypeSolver.add(new ReflectionTypeSolver());

        JavaSymbolSolver symbolSolver =
            new JavaSymbolSolver(combinedTypeSolver);
        StaticJavaParser.getConfiguration()
            .setSymbolResolver(symbolSolver);

        CompilationUnit cu = StaticJavaParser.parse(
            "import java.util.List;\n" +
            "import java.util.ArrayList;\n" +
            "class D {\n" +
            "    void main() {\n" +
            "        Integer[] a = new Integer[]{1, 2};\n" +
            "        List<Integer[]> l = new ArrayList<>();\n" +
            "        l.add(a);\n" +
            "        m(1);\n" +
            "    }\n" +
            "    <T extends Number> void m(List<T[]> a) {}\n" +
            "}\n");

        MethodCallExpr method = cu.findFirst(
            MethodCallExpr.class,
            c -> c.getNameAsString().equals("m")
        ).get();

        System.out.println(method.calculateResolvedType());
    }
}
```

Stack trace excerpt.

```
Exception in thread "main" java.lang.UnsupportedOperationException: T[]
    at MethodResolutionLogic
        .isAssignableMatchTypeParametersMatchingQName(...)
    at MethodResolutionLogic
        .isAssignableMatchTypeParameters(...)
    at MethodResolutionLogic.isApplicable(...)
    ...
    at Expression.calculateResolvedType(Expression.java:552)
    at TypeParamIsArray.main(TypeParamIsArray.java:29)
```

Comments.

- The reporter confirms Java 11.0.18.

- The reproduction is later updated because the original parsed compilation unit did not compile.
- The suspected cause is incorrect type inference for `Arrays.asList(int[])`.
- The relevant method may be `MethodCallExprContext.resolveMethodTypeParameters(...)`.
- The key distinction is that `Arrays.asList(new Integer[] {1, 2})` should return `List<Integer>`, while `Arrays.asList(new int[] {1, 2})` should return `List<int[]>`.

Additional code from discussion.

```
public static<T> List<T> asList(T... a) {
    return new ArrayList<>(a);
}
```

Example 4: Go Error Fixing

Subset: `error_fix_go_v1_3`. **Total cases:** 30.

Prompt. Please fix the bug in the project according to the information below.

Issue description. The CORS middleware sends empty and/or unnecessary headers.

Problem description.

- It sends an empty `Access-Control-Allow-Origin` header when the request has no `Origin` header or when the origin is not allowed.
- It sends CORS headers even when the request has no `Origin` header or when the provided origin is not allowed.

Expected behavior. The CORS middleware should not send an empty `Access-Control-Allow-Origin` header. It should only send `Access-Control-*` headers when a valid `Origin` header is provided.

Reproduction commands.

```
# Origin is not allowed
curl -X OPTIONS -H "Origin: http://bar.com" -I http://localhost:1323/

HTTP/1.1 204 No Content
Access-Control-Allow-Methods: GET,HEAD,PUT,PATCH,POST,DELETE
Access-Control-Allow-Origin:
Vary: Origin
...

# Origin is missing
curl -X OPTIONS -I http://localhost:1323/

HTTP/1.1 204 No Content
Access-Control-Allow-Methods: GET,HEAD,PUT,PATCH,POST,DELETE
Access-Control-Allow-Origin:
Vary: Origin
...
```

Debugging code.

```
package main

import (
    "net/http"

    "github.com/labstack/echo/v4"
    "github.com/labstack/echo/v4/middleware"
)

func main() {
    e := echo.New()
    e.Use(middleware.CORSWithConfig(middleware.CORSConfig{
        AllowOrigins: []string{"http://example.com"},
    }))
}
```

```

e.GET("/", func(c echo.Context) error {
    return c.String(http.StatusOK, "Hello, World!")
})

e.Logger.Fatal(e.Start(":1323"))
}

```

Version. v4.1.17

Example 5: JavaScript Code Generation

Subset: code_gen_js_v1_3. **Total cases:** 150.

Prompt. Please implement the following feature based on the provided information.

Feature description. Expose a `resetScale` method from the `ImagePreview` component.

Motivation. The user needs a button-controlled way to reset image zoom.

Proposed API. Expose `resetScale` as part of the component API.

Comments.

- The maintainers agree with the requested API change.
- They invite a pull request against the `main` branch.
- They ask for changelog entries, TypeScript definitions, documentation, and tests if needed.
- They also ask the contributor to make sure CI passes before review.

Tips.

```

ImagePreview component:
  expose resetScale through useExpose
  resetScale signature: () => void

ImagePreviewItem component:
  expose resetScale
  export ImagePreviewItemProps

types.ts:
  add ImagePreviewItemExpose
  add ImagePreviewItemInstance
  add resetScale to ImagePreviewExpose

```

Example 6: Python Code Generation

Subset: code_gen_python_v1_3. **Total cases:** 90.

Prompt. Please implement the following feature based on the provided information.

Feature description. Implement a Syntetos–Boylan ADI/CV feature extractor for classifying different types of demand.

Task. The implementation should follow the Syntetos–Boylan expert classification of time series from *The accuracy of intermittent demand estimates*.

Parameters.

```

adi_threshold: default 1.32
cv_threshold: default 0.49
features: optional list of {"adi", "cv2", "class"}

```

Behavior.

```

adi:
  average demand interval.
  last index minus first index,

```

```

divided by number of non-zero values minus one.

cv2:
    variance / mean squared,
    computed on the sample of non-zero values.
    The reference uses the biased variance estimator.

class:
    adi <= threshold and cv <= threshold: "smooth"
    adi <= threshold and cv > threshold: "erratic"
    adi > threshold and cv <= threshold: "intermittent"
    adi > threshold and cv > threshold: "lumpy"

```

Comments.

- A contributor expresses interest in implementing the feature to better understand time-series analysis.
- The maintainer confirms that the task is suitable as a good first issue.
- The maintainer offers help with the new-estimator guide or partial pull requests.
- The contributor later explains a delay due to midterms, and the maintainer confirms that there is no rush.

Tips.

```

Create class ADICVTransformer in:
    sktime/transformations/series/adi_cv.py

Class:
    ADICVTransformer(BaseTransformer)

Constructor:
    __init__(self, features=None,
              adi_threshold=1.32,
              cv_threshold=0.49)

_transform:
    return pd.DataFrame with columns:
        "adi", "cv2", "class"

Class labels:
    "smooth", "erratic", "intermittent", "lumpy"

```

Example 7: Frontend Artifact Generation

Subset: artifacts_v3_1. **Total cases:** 100.

Prompt excerpt A. Implement the following React requirement in `pages/index.tsx`. Prefer Tailwind for styling. All required IDs, classes, function names, and implementation constraints must be followed exactly.

Example task: Restaurant inventory statistics page.

```

Page title:
<title>Restaurant Inventory Statistics</title>
<h1>Restaurant Inventory Statistics</h1>

Sales form:
#sales-form
#gongbao Kung Pao Chicken sold today
#yuxiang Yu-Shiang Shredded Pork sold today
#qingzhu Steamed Sea Bass sold today
#edit Update inventory

Error area:
#error-messages
show "Please enter a numeric value" if input is invalid
show "Insufficient inventory" if stock is insufficient
hide after 2 seconds

```

```
Stock table:
#stock-table
columns: Dish Name / Current Stock / Sold Today
```

Initialization.

```
let dishes = [
  { id: "gongbao", name: "Kung Pao Chicken", stock: 100, threshold: 0 },
  { id: "yuxiang", name: "Yu-Shiang Shredded Pork", stock: 80, threshold: 0 },
  { id: "qingzhu", name: "Steamed Sea Bass", stock: 60, threshold: 0 }
];
```

Prompt excerpt B. Implement a product-review management system with two modules: product management and review management.

Required elements:

```
#container, #title, #nav
#nav-product, #nav-review
```

Product module:

```
#product-module, #add-commodity
#add-product-form
#product-name, #product-price
#product-description, #product-stock
#product-status, #product-category
#add-product-btn, #cancel
#product-list
.edit-commodity, .delete-commodity
```

Review module:

```
#review-module, #add-comment
#review-form, #review-user
#rating-stars, .star
#review-content, #add-review-btn
#review-list
.edit-comment, .delete-comment
```

Modal:

```
#modal, #modal-content
#modal-message, #modal-textarea
#modal-yes, #modal-no
```

Original code file.

```
import React from 'react';

const Demo = () => {
  return (
    <main>
      <h1>Rsbuild with React</h1>
    </main>
  );
};

export default Demo;
```

Example 8: Web-Bench Code Task

Subset: web_bench_code_v2_2. **Total cases:** 80.

Prompt. Add a square-root button with the text “ $\sqrt{\quad}$ ” to the right of the clear button. Clicking it should calculate the square root using the current display content directly.

Original code.

```
<!doctype html>
<html lang="en">
  <head>
    ...
    <title>Simple Calculator</title>
    <style>
      .calculator { width: 300px; margin: 50px auto; ... }
```

```

    .display { width: 100%; height: 40px; ... }
    .buttons {
      display: grid;
      grid-template-columns: repeat(4, 1fr);
      gap: 5px;
    }
    button { padding: 10px; font-size: 18px; ... }
  </style>
</head>
<body>
  <div class="calculator">
    <input type="text" class="display" id="display" readonly />
    <div class="buttons">
      <button onclick="calculate('7')">7</button>
      ...
      <button onclick="calculate('=')">=</button>
      <button onclick="calculate('+')">+</button>
      <button onclick="calculate('C')" style="grid-column: span 4">
        Clear
      </button>
    </div>
  </div>
</body>
</html>
<script>
  let displayValue = '';

  function calculate(value) {
    if (value === 'C') {
      displayValue = '';
    } else if (value === '=') {
      try {
        displayValue = eval(displayValue).toString();
      } catch (error) {
        displayValue = 'Error';
      }
    } else {
      displayValue += value;
    }
    document.getElementById('display').value = displayValue;
  }
</script>
</body>
</html>

```

Example 9: Repository Question Answering

Subset: repo_qa_agent_v1_2. **Total cases:** 100.

Prompt format. Repo QA tasks evaluate whether the agent understands the whole repository, including architecture, repository inputs and outputs, module behavior, implementation details, and bug-localization reasoning.

Task categories.

1. Architecture summarization
2. Repository input/output understanding
3. Module understanding
4. Implementation Q&A
5. Bug localization

Example 10: Repository Environment Management

Subset: repo_env_v1_1. **Total cases:** 60.

Prompt format. Repo Env tasks evaluate whether the agent can reconstruct and debug a runnable repository environment.

Typical requirements.

1. Inspect the repository structure.

2. Install or repair dependencies.
3. Configure the runtime environment.
4. Run tests or project commands.
5. Diagnose build, dependency, or configuration failures.
6. Verify the repository in a clean execution environment.

B Seed-for-Seed Showcase

This appendix presents representative Seed-for-Seed showcase cases. Instead of treating these cases as isolated demos, we organize them as a model-development loop in which Seed2.1 participates in evaluation, data, training, and infrastructure workflows. The cases illustrate four recurring capabilities: long-horizon execution, target-driven iteration, multi-agent collaboration, and reusable workflow consolidation.

Overall Showcase A complete internal showcase, including demo videos, execution traces, intermediate artifacts, and final reports, is available at: [Seed-for-Seed Overall Showcase](#).

Eval Loop	Eval Loop	Data Loop
<p>Case 1: Automated Benchmark Integration</p> <p><i>Problem:</i> external benchmarks require manual adaptation before internal evaluation.</p> <p><i>Agent workflow:</i> read paper/repo → adapt harness → configure verifier → run smoke tests → align scores.</p> <p><i>Output:</i> runnable internal benchmark and evaluation report.</p>	<p>Case 2: Automated Model Diagnosis</p> <p><i>Problem:</i> score tables do not explain capability gaps or next-step improvements.</p> <p><i>Agent workflow:</i> Auto-Eval selects or generates tests → Post-Eval analyzes traces → hypotheses are verified through new tests.</p> <p><i>Output:</i> model-diagnosis report and reusable hard-case set.</p>	<p>Case 3: Iterative Rule Cleaning</p> <p><i>Problem:</i> noisy pretraining data requires repeated regex/rule cleaning.</p> <p><i>Agent workflow:</i> Optimizer proposes rules → Evaluator checks precision/recall → rules are revised.</p> <p><i>Output:</i> cleaned data, rule history, and failure cases.</p>
Training Loop	Training Loop	Infra Loop
<p>Case 4: SFT Data Synthesis</p> <p><i>Problem:</i> improving an SFT metric requires a long algorithm-engineering loop.</p> <p><i>Agent workflow:</i> propose data plan → synthesize data → launch SFT → run companion evaluation.</p> <p><i>Output:</i> synthetic data, training job, evaluation results, and experiment report.</p>	<p>Case 5: GUI Agent Co-Evolution</p> <p><i>Problem:</i> failed GUI rollouts are hard to convert into precise corrective training data.</p> <p><i>Agent workflow:</i> compare success/-failure trajectories → locate earliest divergence → generate leak-safe corrective skills.</p> <p><i>Output:</i> updated task set, corrective skills, and training data.</p>	<p>Case 6: RL Framework Optimization</p> <p><i>Problem:</i> RL infrastructure changes often require heavy engineering effort.</p> <p><i>Agent workflow:</i> build multi-agent harness → actor implements changes → critic evaluates experiments → iterate to quality gate.</p> <p><i>Output:</i> reusable multi-agent harness and optimized RL training framework.</p>

Table 8 Seed-for-Seed showcase matrix. The six cases cover evaluation, data, training, and infrastructure loops in the model-development lifecycle.

C Fundamental Capability

Seed 2.1 maintains frontier-level capability in world knowledge, reasoning, and visual understanding. In addition, Seed 2.1 further strengthens multilingual capability, reflecting the Seed model family’s commitment to broader international deployment.

Capability	Benchmark	GPT-5.5	Gemini-3.1 Pro	Claude-4.7 Opus	Seed2.1 Turbo	Seed2.1 Pro
Infographics	ChartQAPro [37]	69.4	<u>70.2</u>	65.5	70.9	70.9
	OCRBenchv2 [21]	61.1	<u>62.8</u>	56.9	<u>62.8</u>	63.2
	CharXiv-DQ [66]	<u>95.0</u>	94.9	93.1	94.6	95.5
	CharXiv-RQ [66]	<u>83.2</u>	<u>83.5</u>	82.1	82.5 / 83.6*	85.4 / 86.4*
Math	MathVista [34]	84.2	90.2	84.4	<u>90.5</u>	90.7
	MathVision [61]	92.2	89.2	83.1	90.1 / 92.7*	92.6 / 94.5*
	DynaMath [84]	75.9	72.1	63.1	68.1	<u>73.1</u>
	MathVerse (Vision-Only) [79]	84.6	87.7	77.4	<u>89.2</u>	89.7
STEM	MMMU-Pro [78]	<u>81.2</u>	80.5	74.0	80.1 / 82.2*	81.6 / 82.7*
	EMMA [23]	<u>79.0</u>	72.1	68.7	78.4	79.3
Visual Puzzles	ZeroBench [48] (main)	<u>13.0</u>	12.0	8.0	11.0 / 20.0*	18.0 / 22.0*
	ZeroBench (sub)	41.0	41.9	37.1	<u>49.1 / 57.2*</u>	49.4 / 56.3*
	VisuLogic [71]	43.5	44.8	32.6	<u>52.9</u>	54.3
Perception & Recognition	VLMsAreBiased [58]	49.8	<u>73.5</u>	30.2	68.3	83.6
	VisFactor [26]	56.2	<u>54.0</u>	30.5	43.9	51.4
	RealWorldQA [69]	82.2	85.4	75.6	<u>86.3</u>	86.7
	BabyVision [7]	55.9	54.4	22.2	<u>62.9</u>	73.7
	MeasureBench [33]	49.9	44.4	29.7	<u>58.9</u>	62.9
Knowledge	SimpleVQA [10]	58.6	69.9	56.5	<u>71.1</u>	74.5
	WorldVQA [82]	34.6	44.3	35.9	<u>48.6</u>	53.0
	WorldBench [77]	58.4	62.9	52.9	<u>63.7</u>	67.6
Spatial Reasoning	BLINK [22]	78.3	79.1	70.4	<u>79.4</u>	81.4
	MMSIBench (circular) [75]	36.0	27.4	17.4	31.4	<u>35.9</u>
	TreeBench [60]	61.5	60.5	36.8	71.1	71.1
	ERQA [53]	<u>64.5</u>	70.8	52.5	<u>71.3</u>	72.0
	EmbSpatialBench [18]	81.9	84.2	77.2	<u>82.5</u>	<u>83.4</u>
LongContext Understanding	DUDE [56]	81.7	82.1	84.5	<u>83.1</u>	82.8
	MMLongBench-128K [64]	-	70.7	-	<u>76.9</u>	78.3

Table 9 Fundamental vision capability evaluation. We report Pass@1 in these benchmarks. The best score (no-tool result) for each benchmark is marked in **bold**, and the second best is underlined. Seed2.1 results marked with an * are obtained with tools.

C.1 Vision Capabilities

We evaluate Seed2.1 Pro on a broad suite of public vision-language benchmarks, covering infographics, mathematical and STEM reasoning, visual puzzles, perception and recognition, knowledge-intensive VQA, spatial reasoning, and long-context understanding. As shown in Table 9, Seed2.1 Pro delivers consistently strong results against leading frontier models, achieving the best performance on a wide range of benchmarks. In particular, it obtains top scores on CharXiv-RQ (85.4, or 86.4 with tools), MathVision (92.6, or 94.5 with tools), MMMU-Pro (81.6, or 82.7 with tools), ZeroBench, and several perception-oriented benchmarks, demonstrating a substantial improvement in fundamental visual understanding.

Math, STEM, and Knowledge. Seed2.1 Pro demonstrates comprehensive strength in mathematical reasoning, achieving the best no-tool performance on MathVista (90.7), MathVision (92.6), and MathVerse (89.7), while remaining highly competitive on DynaMath. With tool use, its advantage becomes more pronounced, reaching 94.5 on MathVision and further widening the gap over other frontier models. In STEM reasoning, Seed2.1 Pro also shows strong progress, achieving the best result on MMMU-Pro (81.6, or 82.7 with tools) and EMMA (79.3). For knowledge-intensive VQA, a traditionally strong area for the Seed series, Seed2.1 Pro further consolidates its advantage, setting new best results on SimpleVQA (74.5), WorldVQA (53.0), and WorldBench (67.6), indicating more reliable visual knowledge grounding and real-world recognition.

Perception-Centric Visual Understanding. Beyond reasoning-heavy tasks, Seed2.1 Pro makes broad improvements in perception-centered capabilities, including infographic understanding, visual puzzles, perception and

Capability	Benchmark	Human	Gemini-3.1-Pro	Gemini-3.5-Flash	Seed2.1 Turbo	Seed2.1 Pro
Knowledge & Reasoning	VideoSimpleQA [5]	-	70.0	<u>76.0</u>	71.4	76.4
	VideoHolmes [‡] [9]	-	65.9	<u>67.1</u>	<u>67.6</u>	68.2
	Minerva [‡] [39]	-	63.5	<u>68.6</u>	65.9	70.7
Motion & Perception	TVBench [11]	94.8	71.0	76.4	<u>77.2</u>	80.5
	TOMATO [49]	95.2	60.4	<u>71.9</u>	56.8	79.5
	MotionBench [25]	-	69.9	70.6	<u>74.8</u>	74.9
	ContPhy [81]	-	64.5	<u>63.8</u>	61.1	63.6
Long Video	VideoMME [‡] [20]	-	86.7	87.2	<u>89.0</u>	89.2
	LVBench [63]	-	66.2	76.3	<u>76.8</u>	78.0
	LongVideoBench [68]	-	76.5	77.6	80.6	80.6
Multi Video	CrossVid [30]	89.2	48.8	58.6	<u>63.2</u>	65.0
Streaming	LiveSports-3K [6]	-	89.0	<u>86.5</u>	77.1	76.8
	OVOBench [40]	92.8	64.1	64.5	<u>79.2</u>	80.7
	OVBench [27]	-	58.8	56.5	<u>69.7</u>	70.0

Table 10 Fundamental video understanding capability evaluation.

recognition, and spatial reasoning. It achieves leading results on OCRBenchv2 (63.2), CharXiv-DQ (95.5), and CharXiv-RQ (85.4, or 86.4 with tools), reflecting stronger document understanding, chart reading, and fine-grained visual detail extraction. On visual puzzle benchmarks, Seed2.1 Pro reaches the best performance on ZeroBench main (18.0, or 22.0 with tools) and VisuLogic (54.3), showing that improved perception also benefits visually grounded logic solving. The model further achieves top results on VLMsAreBiased (83.6), RealWorldQA (86.7), BabyVision (73.7), and MeasureBench (62.9), and leads or matches the best performance on spatial benchmarks such as BLINK (81.4), TreeBench (71.1), and ERQA (72.0), suggesting stronger robustness in fine-grained recognition and real-world spatial understanding.

Long-Context Understanding. Seed2.1 Pro also performs strongly on long-context multimodal understanding, achieving 78.3 on MMLongBench-128K. This indicates its ability to process long documents, multi-page materials, and extended task contexts, supporting more stable performance in complex agentic workflows.

Qualitative Results. Beyond the quantitative evaluation results reported above, we also include qualitative examples that demonstrate Seed2.1’s capabilities in tool-augmented vision and visual coding. Further details can be found in Section E and Section F, respectively.

C.2 Video Understanding Capabilities

We conduct a comprehensive evaluation of the Seed2.1 family, including Seed2.1-Turbo and Seed2.1-Pro, across a broad spectrum of video understanding capabilities, spanning knowledge and reasoning, motion perception, long-video comprehension, multi-video understanding, and streaming video analysis. As shown in Table 10, Seed2.1 consistently advances the performance frontier across these dimensions, with Seed2.1-Pro serving as the strongest model in the family and achieving leading results on most reported benchmarks. These results demonstrate the effectiveness of Seed2.1 in strengthening core video intelligence, particularly in video reasoning, fine-grained motion understanding, long video understanding, and real-time streaming scenarios.

Specifically, Seed2.1-Pro achieves strong performance on knowledge and reasoning benchmarks, reaching 75.5 on VideoSimpleQA, 68.2 on VideoHolmes, and 70.7 on Minerva. Its gains are especially pronounced in motion perception, where it obtains 80.5 on TVBench, 79.5 on TOMATO, and 74.9 on MotionBench, substantially outperforming Gemini-3.1-Pro and Gemini-3.5-Flash on these benchmarks and narrowing the gap to human-level motion understanding. For long-video understanding, Seed2.1-Pro further reaches 89.2 on VideoMME, 78.0 on LVBench, and 80.6 on LongVideoBench, showing robust capability in long-range temporal

modeling and evidence aggregation. In streaming video understanding, Seed2.1-Pro also achieves 80.7 on OVOBench, highlighting its improved ability to process evolving visual inputs and support interactive video reasoning. Despite these advances, gaps remain on several challenging settings such as human-level motion perception and certain streaming scenarios, indicating that fine-grained temporal state tracking, physical dynamics modeling, and real-time cross-context reasoning remain important directions for future work.

C.3 Language Capabilities

This section evaluates the model’s fundamental language capability, including knowledge, reasoning, and multilingual performance. The knowledge evaluation covers both broad graduate-level knowledge and culturally grounded factual knowledge, using benchmarks such as SuperGPQA [19], KINA [28], and HLE-Verified [44]. The reasoning evaluation further includes scientific and research-code reasoning through SciCode [55], as well as olympiad-style frontier science reasoning through FrontierScience-Olympiad [62]. In addition, we include MSQA to evaluate multilingual and multicultural knowledge coverage, and use HLE-textonly and BrowseComp with search tools to assess search capabilities.

Category	Benchmark	GPT-5.5	Claude-4.7 Opus	Gemini-3.1 Pro	Seed2.1 Turbo	Seed2.1 Pro
Knowledge	SuperGPQA [19]	72.7	68.5	76.6	67.4	70.8
	KINA [28]	52.6	46.7	53.2	46.6	48.3
	HLE-Verified (no tool) [44]	50.4	46.9	48.2	42.4	42.9
	SuperChem [80]	61.1	55.0	71.4	56.6	59.8
	ArcAGI2 [45]	85.0	75.8	77.1	61.3	62.5
Reasoning	SciCode [55]	58.4	56.4	62.3	57.8	59.8
	FrontierScience-Olympiad [62]	69.0	69.0	79.0	76.0	75.0
	FS - Research [83]	33.9	20.0	16.7	23.3	28.3
	LiveMathematicianBench [24]	50.8	36.7	43.5	27.7	20.9
	MathArena Apex	69.8	-	58.9	35.4	31.3
	AetherCode	81.6	52.0	74.7	67.9	65.8
BeyondAIME [4]	91.0	79.0	90.0	88.0	87.0	
Multilingual	MSQA	57.4	42.8	69.7	42.0	50.2
Search	HLE-textonly(with Search) [44]	52.2	54.7	51.4	54.6	55.7
	BrowseComp(with Search) [67]	84.4	79.3	85.9	84.9	86.2

Table 11 Fundamental language capability evaluation.

MSQA MSQA, short for Multicultural SimpleQA, is an internal multilingual benchmark designed to evaluate culturally specific factual knowledge across 11 major languages. It contains 1,086 human-annotated and reviewed questions covering history and collective memory, beliefs and value systems, social norms and customs, linguistic expression, and cultural products. Each question is designed to have a single objective answer, remain stable over time, and focus on knowledge that is culturally grounded rather than globally generic.

Results As shown in Table 11, Seed 2.1 shows balanced capability in knowledge, reasoning, and multilingual understanding. Seed2.1-Pro consistently improves over Seed2.1-Turbo on knowledge benchmarks such as SuperGPQA and KINA, as well as on MSQA, indicating stronger factual coverage and better multilingual cultural knowledge with scale. On reasoning-heavy benchmarks, Seed 2.1 performs strongly on SciCode and FrontierScience-Olympiad, with both Seed2.1-Turbo and Seed2.1-Pro remaining competitive against frontier baselines. These results suggest that Seed 2.1 maintains strong core language capability while expanding toward more multilingual and internationally relevant use cases.

D General Agent Showcase Figures

This appendix collects detailed visual showcases for the General Agent section. The main text summarizes the task requirements and model behavior, while the figures here provide supporting visual evidence for crowdsourced evaluation examples, Agent Startup Bench cases, and daily-life task cases.

The input file contained garbled characters, and the OCR recognition also resulted in garbled characters. The task was then completed using visual processing

Reasoning

The input file contained garbled characters, and the OCR recognition also resulted in garbled characters. The task was then completed using visual processing.

Reasoning

The extracted content is highly disorganized, with many duplicated columns. Since the markdown table is corrupted, I will rely on the PDF images to extract the actual course data.

Reasoning

The China-S font uses full-width CJK digits, causing excessive spacing in values such as “2.5”. I will switch back to Helvetica, which uses half-width proportional digits and renders numbers more accurately.

Result

学生中文成绩单											
姓名	小明	学号	201913561208	性别	男	身份证号	200100007	班级	政经教数	出生日期	20020705
院系	国(英)-2	入学日期	20170902	院系简称	机电学院	专业名称	24004机械设计与制造及其自动化(4+2)	培养方案	24004机械设计与制造及其自动化(4+2) 机电类		
课程教学信息											
课程名称	学分	成绩	评价	课程名称	学分	成绩	评价	课程名称	学分	成绩	评价
程序设计基础	94	及格	必修	大学英语(4-1)	96	及格	必修	大学英语(4-1)	96	及格	必修
英语强化1	91	及格	必修	大学英语(4-1)	98	及格	必修	大学英语(4-1)	98	及格	必修
机械制图与计算机绘图(1-1)	84	及格	必修	高等数学A(1-1)	79	不及格	必修	高等数学A(1-1)	79	不及格	必修
体育(1-1)	100	及格	必修	机械制图与计算机绘图(高)	79	不及格	必修	机械制图与计算机绘图(高)	79	不及格	必修
大学英语(4-2)	97	及格	必修	形势与政策(上)-1	94	及格	必修	形势与政策(上)-1	94	及格	必修
英语强化2	93	及格	必修	大学生心理健康教育	99	及格	必修	大学生心理健康教育	99	及格	必修
公共英语听力会话训练	85	及格	必修	高等数学A(2-2)	84	及格	必修	高等数学A(2-2)	84	及格	必修
机械制图与计算机绘图(1-2)	85	及格	必修	机械制图与计算机绘图	94	及格	必修	机械制图与计算机绘图	94	及格	必修
物理学	90	及格	必修	军事理论	94	及格	必修	军事理论	94	及格	必修
数学(1-2)	96	及格	必修	数字图像处理设计(Photoshop)	100	及格	选修	数字图像处理设计(Photoshop)	100	及格	选修
英语强化3	73	不及格	必修	中国近现代史纲要	78	不及格	必修	中国近现代史纲要	78	不及格	必修
大学英语C	97	及格	必修	材料力学	90	及格	选修	材料力学	90	及格	选修
工程制图(1-1)	83	及格	必修	德语强化3	85	及格	选修	德语强化3	85	及格	选修
机械制图与计算机绘图(1-1)	89	及格	必修	机械工业出版社	88	及格	必修	机械工业出版社	88	及格	必修
体育(1-3)	90	及格	必修	线性代数	94	及格	选修	线性代数	94	及格	选修
形势与政策(上)-2	及格	及格	必修	创新创业	95	及格	选修	创新创业	95	及格	选修
大学英语D	100	及格	必修	德语强化4	83	不及格	选修	德语强化4	83	不及格	选修
物理学	81	不及格	必修	机械工业出版社	76	不及格	必修	机械工业出版社	76	不及格	必修
机械制图与计算机绘图	94	及格	必修	机械制图与计算机绘图(1-2)	94	及格	必修	机械制图与计算机绘图(1-2)	94	及格	必修
机械制图与计算机绘图	94	及格	必修	机械工业出版社	77	不及格	必修	机械工业出版社	77	不及格	必修
英语强化4	96	及格	必修	物理学	83	不及格	必修	物理学	83	不及格	必修
数学(1-4)	75	不及格	必修	德语强化1	84	及格	必修	德语强化1	84	及格	必修
英语强化5	97	及格	必修	工程实践(1-2)	82	不及格	必修	工程实践(1-2)	82	不及格	必修
科技英语1	91	及格	必修	跨文化与商务技术	95	及格	必修	跨文化与商务技术	95	及格	必修
公共英语听力会话训练	87	及格	必修	www.mhhe.com/china/9787030499999.html	84	及格	必修	www.mhhe.com/china/9787030499999.html	84	及格	必修
形势与政策(上)-3	及格	及格	必修	英语强化5	75	不及格	必修	英语强化5	75	不及格	必修
单片机原理与应用	69	不及格	必修	单片机原理与应用课程设计	75	不及格	必修	单片机原理与应用课程设计	75	不及格	必修
英语强化6	73	不及格	必修	英语强化6	95	及格	选修	英语强化6	95	及格	选修
机械工业出版社	74	不及格	必修	机械工业出版社课程设计	及格	及格	必修	机械工业出版社课程设计	及格	及格	必修
机械工业出版社	87	及格	必修	科技英语2	93	及格	必修	科技英语2	93	及格	必修
可编程控制器	90	及格	选修	数控加工工艺与编程	86	不及格	选修	数控加工工艺与编程	86	不及格	选修
机械工业出版社	74	不及格	必修	机械工业出版社及英语	88	及格	必修	机械工业出版社及英语	88	及格	必修
生产实习	及格	及格	必修	形势与政策(上)-4	8	不及格	必修	形势与政策(上)-4	8	不及格	必修

Their performance is particularly remarkable in quantitative and foundational engineering courses.

This student has demonstrated a robust academic foundation in the Mechanical Design, Manufacturing, and Automation (Sino-German) program.

Figure 24 Agent Startup Bench educational task. Seed 2.1 performs multi-document reasoning, GPA calculation, and layout-preserving PDF backfilling.

Task

- Using the CSI 500 constituent list as of the close on April 24, 2026, calculate the Altman Z-score for all non-financial constituents, excluding stocks classified under the SW Level-1 banking and non-bank financial sectors. Use the latest disclosed annual report data, i.e., the 2025 annual reports, and the latest trading-day closing market capitalization.
- The Altman Z-score is calculated as:
- $Z = 1.2 \times (\text{Working Capital} / \text{Total Assets}) + 1.4 \times (\text{Retained Earnings} / \text{Total Assets}) + 3.3 \times (\text{EBIT} / \text{Total Assets}) + 0.6 \times (\text{Market Capitalization} / \text{Total Liabilities}) + 1.0 \times (\text{Revenue} / \text{Total Assets})$
- where:
- Working Capital = Current Assets – Current Liabilities
- Retained Earnings = Surplus Reserve + Undistributed Profit
- EBIT = Total Profit + Financial Expenses
- Rank all stocks by Z-score in ascending order and list the 15 stocks with the highest distress risk. Output a table including: stock code, stock name, Working Capital / Total Assets, Retained Earnings / Total Assets, EBIT / Total Assets, Market Capitalization / Total Liabilities, Revenue / Total Assets, and final Z-score. Round all metrics to four decimal places.

Input Attachments

股票代码	股票简称	营运资本/总资产	留存收益/总资产	息税前利润/总资产	总市值/总负债	营业收入/总资产	Z值
600606	绿地控股	0.0136	0.0148	-0.0197	0.0191	0.1682	0.1516
000709	河钢股份	-0.2670	0.0716	0.0261	0.1149	0.4349	0.3697
688538	和辉光电	-0.0211	-0.4298	-0.0538	1.6363	0.2053	0.3826
601106	中国一重	0.0444	-0.3031	0.0133	0.8630	0.2511	0.4418
300070	新水源	-0.0436	0.1597	0.0048	0.2917	0.0958	0.4581
000537	绿发电力	-0.0028	0.1182	0.0203	0.3076	0.0958	0.4609
000539	粤电力A	-0.0815	0.0620	0.0199	0.2382	0.2771	0.4746
601866	中远海发	-0.1888	0.1282	0.0426	0.3524	0.1915	0.4843
601666	平煤股份	-0.2353	0.1839	0.0178	0.3778	0.2536	0.5140
600021	上海电力	-0.0774	0.0463	0.0447	0.3455	0.2016	0.5322
601991	大唐发电	-0.1813	0.0277	0.0517	0.3220	0.3636	0.5486
600663	陆家嘴	0.0127	0.1110	0.0221	0.3312	0.1085	0.5507
600157	永泰能源	-0.2360	0.1275	0.0222	0.6441	0.2143	0.5692
600820	隧道股份	-0.0583	0.1179	0.0231	0.1367	0.3186	0.5721
601155	新城控股	-0.0755	0.2132	0.0200	0.1691	0.1996	0.5750

Agent reasoning + tool call

Planning and Task Decomposition. The model first decomposes the open-ended financial analysis task into trackable subtasks, including dependency setup, constituent filtering, financial data retrieval, market-cap collection, Z-score calculation, and final ranking.

Robust Data Pipeline Construction. Starting from a zero-data environment, the model installs required packages, handles network failures via PyPI mirror fallback, and progressively switches across unstable APIs to build a hybrid pipeline using Eastmoney, Sina, and value_em.

Financial Reasoning and Validation. The model distinguishes key accounting metrics, cross-validates financial fields across data sources, filters out financial-sector constituents, and computes Altman Z-scores for stocks with complete data to identify the 15 highest-risk firms.

Result

股票代码	股票简称	营运资本/总资产	留存收益/总资产	息税前利润/总资产	总市值/总负债	营业收入/总资产	Z值
600606	绿地控股	0.0136	0.0148	-0.0197	0.0191	0.1682	0.1516
000709	河钢股份	-0.2670	0.0716	0.0261	0.1149	0.4349	0.3697
688538	和辉光电	-0.0211	-0.4298	-0.0538	1.6363	0.2053	0.3826
601106	中国一重	0.0444	-0.3031	0.0133	0.8630	0.2511	0.4418
300070	新水源	-0.0436	0.1597	0.0048	0.2917	0.0958	0.4581
000537	绿发电力	-0.0028	0.1182	0.0203	0.3076	0.0958	0.4609
000539	粤电力A	-0.0815	0.0620	0.0199	0.2382	0.2771	0.4746
601866	中远海发	-0.1888	0.1282	0.0426	0.3524	0.1915	0.4843
601666	平煤股份	-0.2353	0.1839	0.0178	0.3778	0.2536	0.5140
600021	上海电力	-0.0774	0.0463	0.0447	0.3455	0.2016	0.5322
601991	大唐发电	-0.1813	0.0277	0.0517	0.3220	0.3636	0.5486
600663	陆家嘴	0.0127	0.1110	0.0221	0.3312	0.1085	0.5507
600157	永泰能源	-0.2360	0.1275	0.0222	0.6441	0.2143	0.5692
600820	隧道股份	-0.0583	0.1179	0.0231	0.1367	0.3186	0.5721
601155	新城控股	-0.0755	0.2132	0.0200	0.1691	0.1996	0.5750

Figure 25 Agent Startup Bench financial task. Seed 2.1 computes Altman Z-scores over a constrained equity universe and produces an audit-friendly risk table.

Task

1. Case Study: Airline Q4 Operations Review in Excel.
2. Given a 2024 Q4 operational workbook for Blue Sky Airlines, covering 300 flights, 6 aircraft types, 15 routes, 4 hubs, around 50 crew members, and delay-compensation rules, the model is required to directly augment the original .xlsx file without modifying the 10 raw sheets. It must create native Excel-based analyses using formulas, named ranges, conditional formatting, charts, and KPI cards, rather than writing static results.
3. The task includes standardizing flight data, calculating cabin revenue, delay compensation, airline losses, crew hours and overtime pay, fuel costs, airport fees, hub cost allocation, route profitability, and aircraft-type efficiency. The final workbook must include a structured operations dashboard with KPI cards, summary tables, and native Excel charts, while preserving sheet order, formatting rules, number formats, frozen panes, filters, and named ranges used in cross-sheet formulas.

Input Attachments

	A	B	C	D	E	F
1	机型代码	机型名称	总座位数	小时油耗(吨)	座位单位成本(元)	机型数量
2	B737	波音737-800	160	0.3	380	1
3	A320	空客A320	180	0.32	360	1
4	A321	空客A321	200	0.35	350	1
5	B738	波音737MAX-8	190	0.28	360	1
6	B787	波音787-9梦想	280	0.62	320	1.6
7	A350	空客A350-900	320	0.7	310	1.6

Agent reasoning + tool call

1. Data Exploration and Task Planning. The model first inspects all 10 raw sheets to understand table structures and reference data, then decomposes the complex Excel task into nine trackable subtasks before execution.
2. Error Diagnosis and Data Standardization. After an initial recalculation produced hundreds of #VALUE! errors, the model traced them to formula-level issues such as flight-number parsing and unquoted Chinese sheet names. It then standardized flight numbers, dates, times, aircraft codes, and overnight flights, reducing formula errors to zero.
3. Formula-Native Execution with Iterative Validation. The model strictly avoids static result values and builds each analysis sheet using native Excel formulas. After every major layer – cleaning, cabin revenue, delay compensation, fuel cost, crew cost, and route allocation – it recalculates the workbook and verifies zero formula errors, eventually producing over 37k valid formulas.
4. Crew Overtime Reasoning. For the most complex crew-cost module, the model expands data to flight-by-crew records, sorts by crew member, month, date, and flight, and implements monthly cumulative-hour logic. It correctly resets hours by month, applies the 100-hour overtime threshold, handles zero/blank durations, and triggers 1.3x overtime pay where applicable.

Result

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	机型代码	机型名称	总座位数	小时油耗(吨)	座位单位成本(元)	机型数量	机型效率	机型油耗	机型成本	机型收入	机型利润	机型贡献	机型排名
2	B737	波音737-800	160	0.3	380	1	0.0008	192.00	7280.00	14560.00	12640.00	12640.00	1
3	A320	空客A320	180	0.32	360	1	0.0018	648.00	2592.00	5184.00	4536.00	4536.00	2
4	A321	空客A321	200	0.35	350	1	0.0028	1120.00	4480.00	8960.00	7840.00	7840.00	3
5	B738	波音737MAX-8	190	0.28	360	1	0.0014	504.00	1960.00	3920.00	3416.00	3416.00	4
6	B787	波音787-9梦想	280	0.62	320	1.6	0.0112	3916.80	15667.20	31334.40	27417.60	27417.60	5
7	A350	空客A350-900	320	0.7	310	1.6	0.0140	4480.00	17920.00	35840.00	31360.00	31360.00	6
8	合计		1330	2.57	1330	6.6	0.0202	7280.00	28960.00	57920.00	50400.00	50400.00	

The dataset already exposes large twin-aisle wide-body aircraft like the A350 and B787 alongside single-aisle workhorses like the B738 and A320.

High-frequency, mixed-fleet operations imply that seat capacity ceilings and Revenue per Available Seat Kilometer (RASK) vary dramatically between flights. The data already spans multiple route codes from R01 to R15, which provides the raw analytical foundation for executing Step 6: "allocating quarterly shared pool expenses proportionally to flights operating out of the four major hub airports based on their aircraft weightings (wide-body: 1.6, single-aisle: 1.0)."

Figure 26 Agent Startup Bench legal task. Seed 2.1 converts bankruptcy-reorganization facts into a structured legal memorandum and recovery strategy.

Task

Case Study: Bankruptcy Reorganization Claim Filing and Recovery Strategy

Role: Bankruptcy restructuring and commercial disputes counsel.

Client: Huadong Intelligent Equipment Co., Ltd., a long-term supplier of lithium-battery production-line vision inspection equipment to Ningbo Yaoshi New Energy Co., Ltd.

Background: Ningbo Yaoshi entered court-supervised bankruptcy reorganization on April 10, 2026. The administrator set May 20, 2026 as the claim filing deadline and May 28, 2026 as the first creditors' meeting.

Task: Prepare a practical memorandum for the client's finance, business, legal, and external counsel teams, covering:

- Calculate ordinary bankruptcy claims under Contracts A–C, separately listing principal, pre-acceptance interest, unmaturred claims deemed due, and set-off items. Explain how set-off affects claim amount, voting base, and distribution base.
 - Assess recovery paths for title-retained equipment and goods in transit under Contracts D–E, distinguishing reclaim rights, secured priority, subrogation to resale proceeds, and ordinary claim conversion.
 - Advise whether the client may refuse further supply under Contract F when the administrator requests continued performance without providing security, and classify any new supply claims.
 - Determine whether the client may continue pursuing the parent company's joint guarantee after filing bankruptcy claims, and explain how double recovery should be avoided.
 - Provide an executable action checklist before the first creditors' meeting, including claim filing, set-off notice, reclaim notice, contract-performance reply, guarantor demand letter, and supporting materials.
- Output Requirements: Deliver a structured, slide-ready legal memorandum with exact claim amounts, statutory references, no tables, no generic advice, and clear action items.

Agent reasoning + tool call

Case Study Highlights

The model accurately computes and files bankruptcy claims on a gross basis, reporting RMB 15.1341M before setoff and treating the RMB 1.20M rebate as a separate setoff notice rather than netting it directly.

It classifies title-retained equipment unit by unit, distinguishing reclaimable units, resold units, proceeds-based secured recovery, and fallback ordinary claims.

For goods in transit, it correctly applies the seller's reclamation right based on shipment, non-receipt, and non-payment, rather than being misled by EXW risk or title transfer.

It also confirms that bankruptcy filing does not bar parallel recovery against the guarantor, while requiring claim and voting adjustments to avoid double recovery.

Result

The figure displays a grid of 12 panels, each representing a section of a legal memorandum. The panels are organized into three main parts:

- Part 1: General Information and Overview**
 - Panel 1: Title page: 宁波曙石破产重整债权申报与回款路径备忘录
 - Panel 2: Table of Contents (目录)
 - Panel 3: Part 1: General Information (第一部分)
- Part 2: Contractual Analysis**
 - Panel 4: Part 2: Contractual Analysis (第二部分)
 - Panel 5: Sub-section 2.1: Contract (I) (所有保留设备的品牌及规格)
 - Panel 6: Sub-section 2.2: Contract (II) (所有保留设备的品牌及规格)
 - Panel 7: Sub-section 2.3: Contract (III) (所有保留设备的品牌及规格)
 - Panel 8: Sub-section 2.4: Contract (IV) (所有保留设备的品牌及规格)
- Part 3: Legal and Procedural Matters**
 - Panel 9: Part 3: Legal and Procedural Matters (第三部分)
 - Panel 10: Sub-section 3.1: Legal (所有保留设备的品牌及规格)
 - Panel 11: Sub-section 3.2: Legal (所有保留设备的品牌及规格)
 - Panel 12: Sub-section 3.3: Legal (所有保留设备的品牌及规格)

Figure 28 Agent Startup Bench operations-analytics task. Seed 2.1 converts fragmented operational worksheets into a board-ready, formula-driven Excel analysis system.

Task

Case Study: Airline Q4 Operations Review in Excel.
 Given a 2024 Q4 operational workbook for Blue Sky Airlines, covering 300 flights, 6 aircraft types, 15 routes, 4 hubs, around 50 crew members, and delay-compensation rules, the model is required to directly augment the original .xlsx file without modifying the 10 raw sheets. It must create native Excel-based analyses using formulas, named ranges, conditional formatting, charts, and KPI cards, rather than writing static results.
 The task includes standardizing flight data, calculating cabin revenue, delay compensation, airline losses, crew hours and overtime pay, fuel costs, airport fees, hub cost allocation, route profitability, and aircraft-type efficiency. The final workbook must include a structured operations dashboard with KPI cards, summary tables, and native Excel charts, while preserving sheet order, formatting rules, number formats, frozen panes, filters, and named ranges used in cross-sheet formulas.

Input Attachments

	A	B	C	D	E	F
1	机型代码	机型名称	总座位数	小时利用率	座位单位成本(元)	机队规模
2	B737	波音737-800	160	0.3	380	1
3	A320	空客A320	180	0.32	360	1
4	A321	空客A321	200	0.35	350	1
5	B738	波音737MAX-8	190	0.28	360	1
6	B787	波音787-9梦想	290	0.62	320	1.6
7	A350	空客A350-900	320	0.7	310	1.6

Agent reasoning + tool call

Data Exploration and Task Planning. The model first inspects all 10 raw sheets to understand table structures and reference data, then decomposes the complex Excel task into nine trackable subtasks before execution.
Error Diagnosis and Data Standardization. After an initial recalculation produced hundreds of #VALUE! errors, the model traced them to formula-level issues such as flight-number parsing and unquoted Chinese sheet names. It then standardized flight numbers, dates, times, aircraft codes, and overnight flights, reducing formula errors to zero.
Formula-Native Execution with Iterative Validation. The model strictly avoids static result values and builds each analysis sheet using native Excel formulas. After every major layer—cleaning, cabin revenue, delay compensation, fuel cost, crew cost, and route allocation—it recalculates the workbook and verifies zero formula errors, eventually producing over 37k valid formulas.
Crew Overtime Reasoning. For the most complex crew-cost module, the model expands data to flight-by-crew records, sorts by crew member, month, date, and flight, and implements monthly cumulative-hour logic. It correctly resets hours by month, applies the 100-hour overtime threshold, handles zero/blank durations, and triggers 1.3× overtime pay where applicable.

Result

	A	B	C	D	E	F	G	H	I	J	K	L
1	航线代码	航线名称	城市	航班数	运营频率	运营时段	运营时段	运营时段	运营时段	运营时段	运营时段	运营时段
2	R01	拉萨-上海	拉萨	34	3376	422,750.00	Err:502	Err:502	286,000.00	80,750.00	41,200.00	Err:502
3	R02	拉萨-西安	拉萨	28	4576	720,480.00	Err:502	Err:502	360,000.00	90,000.00	45,760.00	Err:502
4	R03	上海-成都	上海	26	3800	638,376.00	174,874.00	524,655.00	335,000.00	91,504.35	1,023.33	-49,790.43
5	R04	拉萨-成都	拉萨	22	2872	543,360.00	Err:502	Err:502	242,000.00	74,053.30	48,914.67	Err:502
6	R05	上海-拉萨	上海	22	3024	621,712.00	Err:502	Err:502	291,000.00	86,543.40	24,246.33	Err:502
7	R06	拉萨-上海	拉萨	29	3748	621,600.00	Err:502	Err:502	290,000.00	120,000.00	32,243.33	Err:502
8	R07	拉萨-成都	拉萨	26	2928	489,972.00	190,991.27	79,844.10	180,000.00	50,803.74	40,409.47	-99,188.77
9	R08	拉萨-成都	拉萨	26	3196	492,544.00	Err:502	Err:502	230,000.00	62,321.96	64,464.47	Err:502
10	R09	上海-拉萨	上海	19	4271	814,446.00	Err:502	Err:502	530,000.00	87,373.91	98,000.00	Err:502
11	R10	拉萨-成都	拉萨	12	2928	287,520.00	Err:502	Err:502	210,000.00	60,589.96	2,858.33	Err:502
12	R11	拉萨-成都	拉萨	13	2460	1,050,807.00	288,876.34	17,233.34	360,000.00	98,242.64	40,269.33	197,886.33
13	R12	上海-拉萨-成都	上海	22	4828	4,190,376.00	Err:502	Err:502	1,056,000.00	190,869.37	41,988.33	Err:502
14	R13	拉萨-成都	拉萨	13	2714	2,261,791.00	Err:502	Err:502	780,000.00	197,766.64	17,668.00	Err:502
15	R14	上海-成都	上海	12	1000	108,360.00	58,166.87	44,765.00	120,000.00	32,668.77	19,000.00	-17,312.70
16	R15	拉萨-成都	拉萨	11	1432	382,432.00	85,346.71	98,887.30	120,000.00	37,236.65	21,389.00	-70,414.60
17	总计			304	48,425.00	14,718,876.00	Err:502	Err:502	6,481,000.00	1,290,000.00	616,861.00	Err:502

The dataset already exposes large twin-aisle wide-body aircraft like the A350 and B787 alongside single-aisle workhorses like the B738 and A320.

High-frequency, mixed-fleet operations imply that seat capacity ceilings and Revenue per Available Seat Kilometer (RASK) vary dramatically between flights. The data already spans multiple route codes from R01 to R15, which provides the raw analytical foundation for executing Step 6: "allocating quarterly shared pool expenses proportionally to flights operating out of the four major hub airports based on their aircraft weightings (wide-body: 1.6, single-aisle: 1.0)."

Figure 29 Agent Startup Bench airline-operations task. Seed 2.1 builds a traceable Q4 executive review workbook from large, multi-source airline operations data.

Task

Today is May 8, 2026. Zhang Chen, male, holds household registration in Fuyang, Anhui Province. His resident identity card is valid until June 28, 2026. His wife Li Shan is registered under the household administration of Xinyang, Henan Province. Their six-year-old daughter has her household registration under Li Shan. The three family members currently rent a residence in Tong'an Subdistrict, Baohe District, Hefei City. The landlord completed the residential registration for Zhang Chen on March 1, 2026.

Zhang Chen's new employer...

Zhang Chen intends to sign a second-hand housing sales contract for a property located in Baohe District, Hefei City on May 18, 2026...

Zhang Chen hopes to complete or schedule the following matters no later than June 30, 2026:

1. Apply for a residence permit;
2. Print the personal social insurance participation certificate;
3. Complete the title transfer registration of the second-hand house as well as the handover of water, electricity and gas accounts;
4. Transfer the household registrations of Zhang Chen, his wife Li Shan and their daughter to Hefei City;
5. Process the cross-city transfer of Shanghai housing provident fund to Hefei;
6. Apply for a new resident identity card as Zhang Chen's current ID card is due to expire.

Constraints: Zhang Chen can only arrange three half-day offline service appointments, which are specified as follows:

- Afternoon of May 22, 2026;
- Morning of June 19, 2026;
- Afternoon of June 23, 2026.

Please formulate an executable handling plan in accordance with official public policies and channels issued by Anhui Province and Hefei Municipal Government.

Output Requirements:

1.

Agent reasoning

1. **Accurate policy understanding:** Distinguished correctly the difference between the intra-provincial one-stop household registration transfer for Zhang Chen with Fuyang, Anhui household registration and the inter-provincial household registration transfer for Li Shan and their daughter with Xinyang, Henan household registration.
2. **Precise judgment on mandatory constraints:** Clearly identified the two preconditions for the cross-regional transfer of the Shanghai housing provident fund (the Shanghai provident fund account has been sealed for more than six months and the Hefei provident fund account has been normally contributed for more than six months). It is specified that the earliest application can only be submitted in November 2026 and the transfer cannot be completed before June 30 to prevent misleading the applicant.
3. **Optimized logical time scheduling:** The offline arrangements for the three half-day time slots strictly follow the dependency sequence of "signing the housing sales contract → approval of mortgaged housing transfer → collection of real estate ownership certificate → household registration relocation". ID card renewal is prioritized on May 22 to avoid invalidation upon expiry in June, with no scheduling conflicts in all arrangements.
4. **Rigorous statement of expected outcomes:** Clarified that Li Shan and their daughter can only submit inter-provincial household transfer application materials on June 23, and the approval for household migration permit will take 10 to 20 working days. The whole formalities can only be finalized from late July to August, without promising a processing time limit beyond official policy provisions.
5. **Comprehensive risk reminders:** More than ten risk points are covered in the precautionary prompts, including clauses on household registration relocation agreed in the housing contract, confirmation of service outlets for mortgaged housing transfer, verification of residence permit policies, and peak periods for ID card renewal. Two items that cannot be finished before June 30 are explicitly listed to avoid unrealistic user expectations in advance.
6. **Directly implementable plan:** Official service channels, core required documents and processing time limits are specified for all procedures. Key information including task completion status, the earliest available processing date and parallel operability is presented in tabular form, allowing the applicant to carry out formalities directly in accordance with the plan.

Result

Zhang Chen's Service Handling Plan for Formalities in Hefei (to Be Completed by June 30, 2026)

The screenshot displays a detailed service handling plan. On the left, there is a list of tasks in Chinese, including 'Apply for a residence permit', 'Print the personal social insurance participation certificate', and 'Transfer household registration'. On the right, a table provides a structured overview of these tasks. The table includes columns for task ID, name, status, and dependencies. A dashed green line highlights a specific path through the tasks, indicating the sequence of actions to be taken.

Task ID	Task Name	Status	Dependencies
1	Apply for a residence permit	Not Started	None
2	Print the personal social insurance participation certificate	Not Started	None
3	Complete the title transfer registration of the second-hand house	Not Started	1, 2
4	Transfer the household registrations of Zhang Chen, his wife Li Shan and their daughter to Hefei City	Not Started	3
5	Process the cross-city transfer of Shanghai housing provident fund to Hefei	Not Started	3, 4
6	Apply for a new resident identity card	Not Started	1, 2, 3, 4, 5


Figure 30 xDailyBench personal government-service scheduling task. Seed 2.1 converts a complex administrative scenario into an executable appointment and document-handling plan.

E Visual Tool Showcase Figures

This appendix collects detailed visual showcases for the Visual Tool section.

USER

The image shows a total of 49 tiger patterns arranged in 7 rows and 7 columns. One of them is different from the others. Which row and column is it in? The answer format is (x,y). (For example, the answer for the 2nd row and 3rd column is (2,3)). Give your final answer in `\boxed{Answer}` format.



1 Segment every tiger

ASSISTANT

The tigers are solid black silhouettes that look almost identical, and a hard grid split would be thrown off by the uneven white margins. I will binarize the image and use **connected components** to cut out each tiger, then sort them by centroid (**row first, then column**) into a 7x7 layout.

```
# binarize, then one connected component = one tiger
_, th = cv2.threshold(img,127,255,cv2.THRESH_BINARY_INV)
n,_,stats,cen = cv2.connectedComponentsWithStats(th,8)
tigers=[(*stats[i][:4],cen[i]) for i in range(1,n) if stats[i][4]>1000]
tigers.sort(key=lambda t:(t[4][1],t[4][0])) # row(y) -> col(x)
rows=[tigers[i*7:(i+1)*7] for i in range(7)]
```

CODE TOOL

```
STDOUT: Number of connected components: 50
Found 49 tigers
-> grouped into 7 rows x 7 cols by centroid
```

2 Tight-crop & translation-align

ASSISTANT

Segmentation gave a clean 7x7 of all 49 tigers. But each one sits at a slightly different offset that would fool a raw pixel diff, so I will tightly crop and center every tiger to a common size, **search ±10px shifts** to align each to a reference, and average the aligned tigers into a **consensus silhouette**.

```
# align each tiger to ref by the shift with minimal SAD
for i in range(7):
    for j in range(7):
        # ...scan dx,dy in [-10,10], keep argmin
        SAD(shift(tig,ref))...
        best_dx[i,j],best_dy[i,j]=argmin_shift(tig[i,j],ref)
overlay = mean(shift(tig[i,j]) for all i,j) # consensus
```

CODE TOOL



DISPLAY — average overlay of all 49 aligned tigers (edges stay sharp => silhouettes agree)

3 Diff against consensus to locate the outlier

ASSISTANT

The averaged overlay comes out with crisp edges, so the silhouettes line up almost perfectly — any real difference must be tiny and local. I will **XOR** each aligned tiger against the consensus, **erode** away anti-aliasing edge noise, and count the residual pixels per cell to pinpoint the outlier.

```
consensus=(overlay>127).astype(np.uint8)*255
for i in range(7):
    for j in range(7):
        xor=cv2.bitwise_xor(shift(tig[i,j]),consensus)
        xor=cv2.erode(xor,np.ones((3,3))) # drop edge noise
        diff[i,j]=(xor>127).sum()
```

CODE TOOL

```
STDOUT: Eroded difference pixel counts
(ignore edge noise):
[[ 0.  0.  0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.  0. 10.]
 [ 0.  0.  0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.]]
Outlier: 1-indexed (4, 7), diff pixels = 10.0
Second largest: 0.0
```



difference map — only (4,7) lights up, at the ear

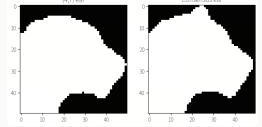
4 Zoom the ear to confirm

ASSISTANT

The eroded diff is zero everywhere except (4,7), which keeps ~10 pixels right at the top of the head. To check this is a genuine shape difference and not an alignment artifact, I will zoom into the top-right 50x50 **ear region** of (4,7) and compare it side-by-side with the consensus ear.

```
# compare top-right 50x50 (ear) of (4,7) vs consensus
plt.subplot(121); plt.imshow(shift_47[:50,-50:]) # (4,7)
plt.subplot(122); plt.imshow(consensus[:50,-50:]) # normal
```

CODE TOOL




DISPLAY — ear zoom. left: (4,7) flat top | right: consensus pointed tip (reaches y=0)

ASSISTANT The zoom confirms it: every normal tiger's ear ends in a **pointed tip** reaching the top edge, but (4,7)'s ear is **flattened** with the tip missing. So the different tiger is in row 4, column 7 → `\boxed{(4,7)}`

Figure 31 Seed2.1 leverages the code tool to better interpret images and tackle the problems.

USER

Based on the image, entering from the upper right and upper left entrances, which one can exit the maze from the lower left exit?
 Choices: (A) Left entrance (B) Right entrance (C) Both entrances can be exited
 Give your final answer in `\boxed{Answer}` format.



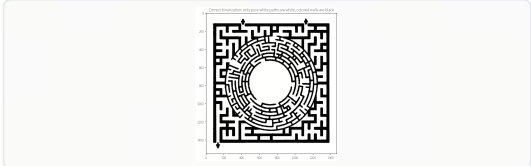
1 Binarize: paths vs. walls

ASSISTANT

This is a reachability problem, better solved with code than by eye. The trap: the inner circular walls are a very light cyan that a plain grayscale threshold would mistake for path. I will keep a pixel as path only if it is **nearly pure white (R,G,B all > 240)**, so every colored wall — red frame, purple, faint cyan — becomes a wall.

```
# pure-white test keeps even the faint cyan rings as walls
mask = (rgb[:, :, 0]>240)&(rgb[:, :, 1]>240)&(rgb[:, :, 2]>240)
binary = np.where(mask, 255, 0).astype(np.uint8) # 255=path, 0=wall
```

CODE TOOL



DISPLAY — pure-white binarization; all walls incl. the faint cyan rings are now solid black

2 Seal the outside background

ASSISTANT

With the walls binarized, the area **outside** the red frame is still white, so a flood-fill could slip out through an entrance gap, walk around the outside, and reach the exit illegally. I will take the **bounding box** of all wall pixels and fill everything outside it with wall, confining the search to inside the maze.

```
# bounding box of all wall pixels, then block everything outside
ys,xs = np.where(binary==0)
min_x,max_x,min_y,max_y = xs.min(),xs.max(),ys.min(),ys.max()
binary[:min_y]=0; binary[max_y+1:]=0
binary[:, :min_x]=0; binary[:, max_x+1:]=0
```

CODE TOOL

```
STDOUT: Outer frame bbox now: min_x=73, max_x=1385,
min_y=61, max_y=1493
```

3 Locate the two entrances and the exit

ASSISTANT

Now I need start/exit points. I will scan the top wall for its two gaps (the entrance arrows) and the bottom wall for the lower-left exit gap, discard the spurious gaps at the very frame edge, and step one point just inside each real opening (the first white pixel past the wall).

```
top_gaps = find_gaps(binary, y=120) # two upper entrances
bottom_gaps = find_gaps(binary, y=1420) # lower-left exit
# keep the real gaps (200<x<1200), step inside each opening
start_left, start_right = inside(top_gaps)
exit_pt = inside(bottom_gaps)
```

CODE TOOL

```
STDOUT: Real start points (upper left and right entrances):
left=(415, 129), right=(1120, 128)
Real exit point (lower left): (134, 1420)
```

4 BFS flood-fill from each entrance

ASSISTANT

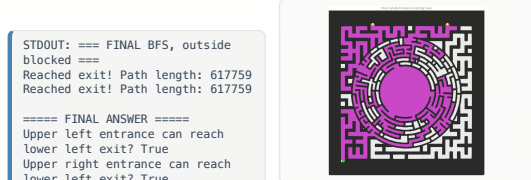
With the three points fixed, I will flood-fill (4-neighbour **BFS**) the white paths from each entrance and check whether the lower-left exit pixel ends up in each entrance's reachable set.

```
def reachable(binary, start): # 4-neighbour BFS over white
    ...
    visited_left = reachable(binary, start_left)
    visited_right = reachable(binary, start_right)
    left_ok = exit_pt in visited_left
    right_ok = exit_pt in visited_right
```

CODE TOOL

```
STDOUT: === FINAL BFS, outside blocked ===
Reached exit! Path length: 617759
Reached exit! Path length: 617759

===== FINAL ANSWER =====
Upper left entrance can reach lower left exit? True
Upper right entrance can reach lower left exit? True
```



DISPLAY — magenta = reachable from both entrances · green = exit

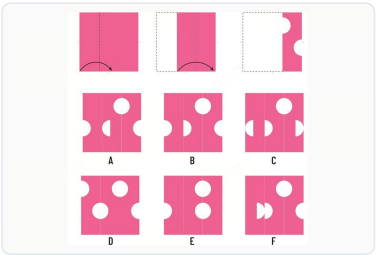
ASSISTANT Both BFS runs reach the exit, and their reachable sets are **identical** (617,759 px) — the open central room links the two top entrances into one region that includes the lower-left exit. So both entrances can exit → `\boxed{C}`

Figure 32 Seed2.1 leverages the code tool to better interpret images and tackle the problems.

65

USER

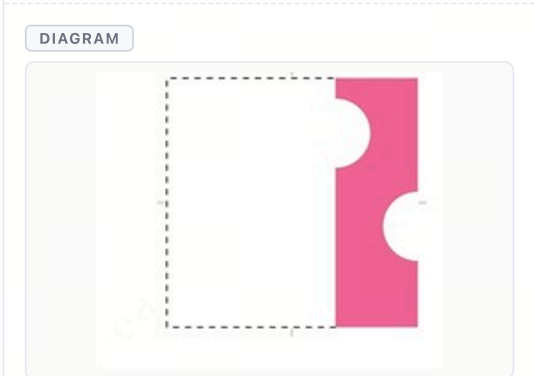
According to the method shown in the diagram, after folding and cutting the card, what is the shape of the entire card?
 Choices: (A) (B) (C) (D) (E) (F)
 Give your final answer in `\boxed{Answer}` format.



1 Read the diagram — a Z (accordion) fold

ASSISTANT

The square is split into **three equal vertical strips**. The first arrow folds the left strip right onto the middle; the second folds that pair right again onto the right strip. Two left-over-right folds make a **Z / accordion fold**, stacking all three layers on the rightmost strip. On that stack the diagram cuts **one semicircle into the left edge near the top** and **one into the right edge lower down**. I will sketch the stack to track which edge is a crease and which is a free edge.



folded Z-stack: left edge = crease $x=2$ / free $x=0$; right edge = crease $x=1$ / free $x=3$

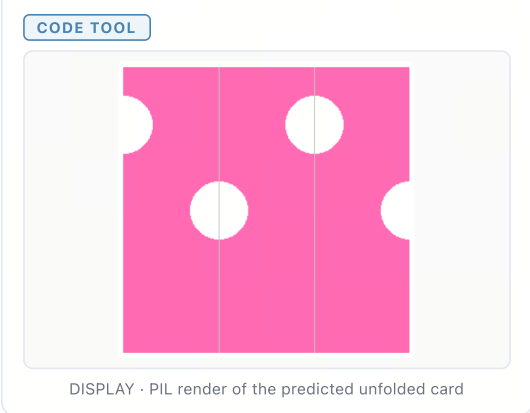
2 Predict the unfolding & render it

ASSISTANT

A notch on a **crease** unfolds into a **full circle** on that fold line; a notch on a **free edge** stays a **half-notch** at the rim. So the top cut \rightarrow full circle on $x=2$ (high) + half-notch far-left; the lower cut \rightarrow full circle on $x=1$ (low) + half-notch far-right. I will draw this with PIL to verify before matching the options.

```

draw.ellipse(circle(200, 60, r)) # full circle on
x=2 crease
draw.pieslice(notch(0, 60, r)) # half-notch far-
left
draw.ellipse(circle(100, 150, r)) # full circle on
x=1 crease
draw.pieslice(notch(300, 150, r)) # half-notch far-
right
  
```



ASSISTANT

The render has two full circles on *different* fold lines with edge half-notches level with them — only one option matches (E's two circles are stacked on the same line) \rightarrow `\boxed{D}`

Figure 33 Seed2.1 leverages the code tool to better interpret images and tackle the problems.

USER

Find the hidden words or phrases (in English) in the sequence of images from left to right. Note, that they might have undergone transforms. Give your answer in lowercase as a comma-separated list.



1 Split the triptych & spot the trick

ASSISTANT

Each panel hides a word inside a fine, repeating texture — grass strands, scattered cows, city lights. Staring at the strokes never resolves it; the word only emerges when I squint, i.e. blur away the texture so local averages fuse into letters. Each panel may also be reoriented. To work them one at a time, I will cut the image into three equal panels and check each one's orientation.

```

w, h = img.size
left = img.crop((0, 0, w//3, h)) # grass strands
mid = img.crop((w//3, 0, 2*w//3, h)) # cows in a field
right = img.crop((2*w//3, 0, w, h)) # coastal city at dusk

```

ASSISTANT

The cows hang from the top edge with their feet up, so the middle panel is flipped top-to-bottom. After undoing that, I will apply a heavy **Gaussian blur** so the individual cows melt into solid strokes — the white cows become the letters and the dark trees fill the holes of the o's.

```

mid_up = mid.transpose(Image.FLIP_TOP_BOTTOM)
blur = cv2.GaussianBlur(np.array(mid_up), (45,45), 0) # "squint"
plt.imshow(blur)

```

CODE TOOL



left panel · grass

CODE TOOL



middle panel · cows

CODE TOOL



right panel · city

2 Middle panel — flip upright, then blur

ASSISTANT

The middle blur clearly reads **hello world**. For the left panel, the brown bank sits at the top with the water reflection above it, so it is rotated a full 180° (not just flipped). I will rotate it back and blur the same way; the lighter green patches between the grass strands should resolve into a word.

```

left_up = left.rotate(180)
blur = cv2.GaussianBlur(np.array(left_up), (75,75), 0)
plt.imshow(blur)

```

ASSISTANT

The left blur reads **athens**. The right panel is already upright — pink sky on top, water below — so no flip or rotation is needed. I will simply blur it and let the street and building lights smear into glowing letters.

```

blur = cv2.GaussianBlur(np.array(right), (75,75), 0) # no reorientation
plt.imshow(blur)

```

CODE TOOL



heavy Gaussian blur of the up-righted middle panel

CODE TOOL



heavy Gaussian blur of the rotated left panel

CODE TOOL



heavy Gaussian blur of the upright right panel

ASSISTANT

The right blur reads **hippopotamus**. Reading the three panels left → right → **athens, hello world, hippopotamus**

Figure 34 Seed2.1 leverages the code tool to better interpret images and tackle the problems.

F Visual Coding Showcases

The visual coding showcases illustrate Seed 2.1’s ability to transform multimodal visual inputs, such as real webpage screenshots, design drafts, sketches, and flowcharts, into directly runnable single-file webpages or frontend applications. These cases highlight the model’s integrated frontend production skills: reconstructing page layout, visual style, content hierarchy, and responsive behavior; implementing interactive components, dynamic effects, and complete user flows; and preserving visual fidelity while producing executable artifacts.

Case 1: Webpage Recreation

User Input

Input image. The user provides a reference webpage screenshot, as shown in Figure 35.

User instruction. Please precisely recreate this website based on the reference image.

Case 2: Sketch-to-Webpage Generation 1

User Input

Input image. The user provides a sketch of the landing page, as shown in Figure 36.

User instruction.

Create an official landing page for the NOVA Esports club based on this draft sketch. Use an 8-bit pixel arcade retro style with the specified high-contrast retro color palette. Add a CRT scan-line effect across the full page. All elements should use square, non-rounded pixel design, with pixel borders and offset shadows. Implement interactions including a boot-up startup animation, player-selection switching, animated match-record bars, a live countdown, and fan barrage message sending. Alternate the corresponding dark backgrounds across different sections as required. For copy details that are not explicit in the sketch, you may supplement or adjust them appropriately.

Unless I explicitly ask to change the layout in the instructions above, or to add content that would change the overall layout, strictly follow the sketch’s layout, module order, element count, spatial relationships, and information hierarchy when generating the webpage. You may add necessary visual details and UI polish only if they do not change the layout, do not break the information hierarchy, and do not conflict with the previous instructions.

Unless I explicitly ask to preserve the draft, hand-drawn, or wireframe style, do not directly copy the sketch’s rough lines, red annotations, arrow explanations, or temporary marks. Treat them as design intent and translate them into a formal webpage UI.

The final page should look like a normal, complete, usable, and visually polished website, not a traced copy of the sketch.

Case 3: Sketch-to-Webpage Generation 2

User Input

Input image. The user provides a sketch of the homepage, as shown in Figure 37.

User instruction.

Create a homepage for the Morandi Atelier home-furnishing brand based on this draft sketch. The overall page should feel premium, quiet, and natural, using a Morandi-inspired low-saturation palette with off-white, gray-brown, and wood-texture tones. Use generous whitespace, soft shadows, rounded cards, and subtle frosted-glass or translucent overlay effects. Headings should have an artistic feel, while body copy should remain concise and restrained. Use high-quality furniture and interior-scene imagery where possible, emphasizing materials, craftsmanship, and lifestyle expression. The hero image should feel immersive, with an elegant gradient overlay. Buttons should

have clear primary and secondary hierarchy. Navigation should be clean on desktop and expandable on mobile. Series cards, space-inspiration filter tabs, service entries, and store information should all include refined hover feedback and transition animations. When switching inspiration categories, the content should update naturally. The newsletter email form should provide clear input states and submission feedback. The brand-story and store-service sections should strengthen messages around craftsmanship, heritage, environmental commitment, appointment experience, and free consultation, making the page feel more like a boutique home-furnishing showroom than an e-commerce promotion page. Keep the copy restrained and elegant, avoiding an overly promotional tone. For copy details that are not fully written in the sketch, you may supplement or adjust them according to the page semantics.

Unless I explicitly ask to change the layout in the instructions above, or to add content that would change the overall layout, strictly follow the sketch's layout, module order, element count, spatial relationships, and information hierarchy when generating the webpage. You may add necessary visual details and UI polish only if they do not change the layout, do not break the information hierarchy, and do not conflict with the previous instructions.

Unless I explicitly ask to preserve the draft, hand-drawn, or wireframe style, do not directly copy the sketch's rough lines, red annotations, arrow explanations, or temporary marks. Treat them as design intent and translate them into a formal webpage UI.

The final page should look like a normal, complete, usable, and visually polished website, not a traced copy of the sketch.

Case 4: Flowchart-to-Webpage Generation

User Input

Input image. The user provides a flowchart-style process sketch, as shown in Figure 38.

User instruction.

Based on this flowchart/process-sketch image, understand the business flow and page logic it expresses, and generate a directly runnable single-file webpage application. Do not redraw the flowchart itself as nodes, arrows, swimlanes, or diagrams; the goal is not to build a flowchart viewer. Instead, implement the real user-facing product experience behind the flow: pages, forms, dashboards, wizards, tools, or applications.

The complete interaction chain in the flowchart must be implemented in the page, including user actions, page states, step transitions, validation, loading states, success paths, failure paths, and conditional branches. If the flow involves backend APIs, data persistence, login authentication, payment, notifications, or third-party services, simulate them with mock functions in frontend JavaScript; the page behavior must still call these mocks and display reasonable request, response, loading, success, and error states.

If the flowchart contains multiple steps or roles, implement the corresponding multi-step UI and state transitions so that the user can complete the process end to end. Use the main language shown in the image for page copy; if it is unclear, use Chinese. Finally, output only one complete `index.html` single-file source code containing HTML, CSS, and JavaScript.

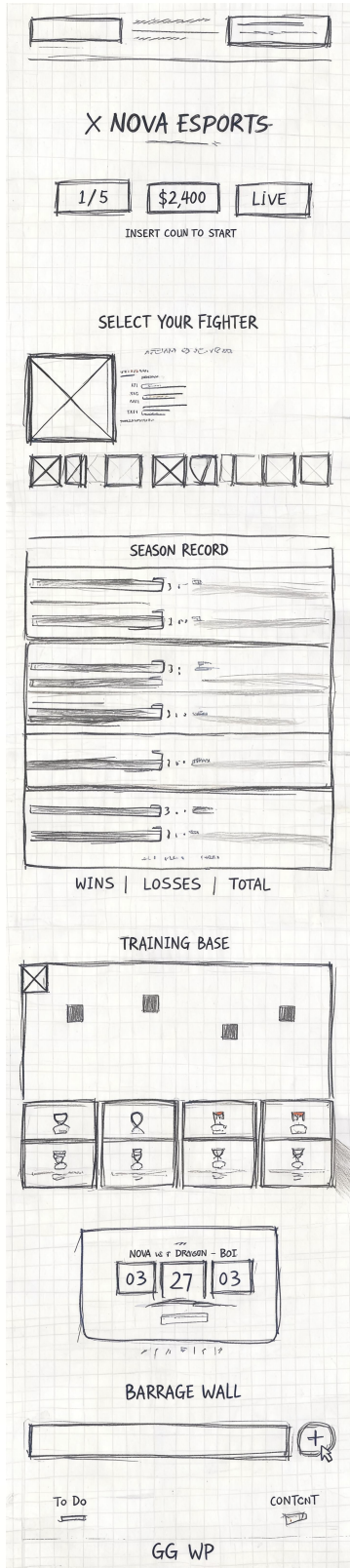


Reference image



Generated webpage

Figure 35 Reference input and generated result for the webpage recreation showcase (case 1).

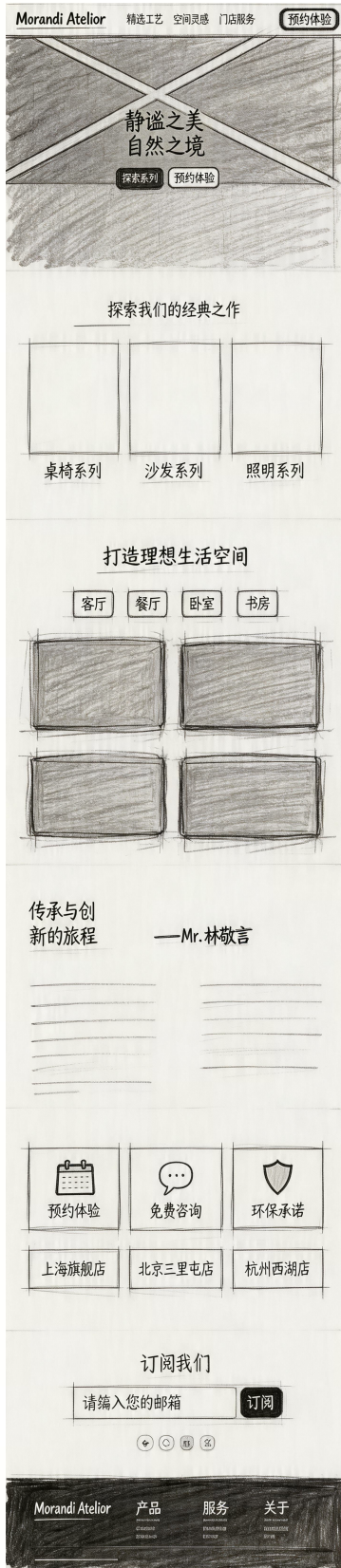


Reference sketch



Generated webpage

Figure 36 Reference input and generated result for the first sketch-to-webpage generation showcase (case 2).

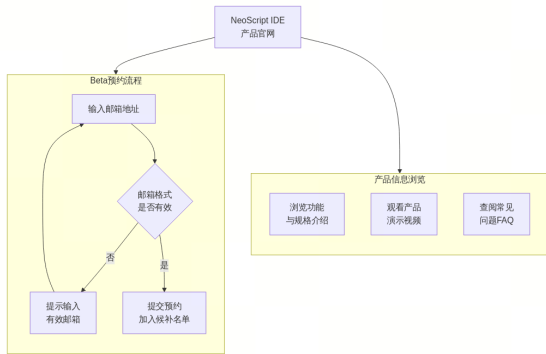


Reference sketch

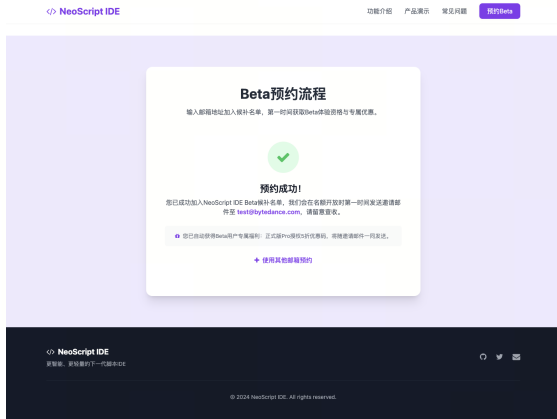


Generated webpage

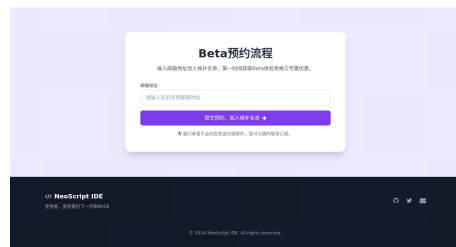
Figure 37 Reference input and generated result for the second sketch-to-webpage generation showcase (case 3).



Reference flowchart



Reservation success state in the generated webpage



Generated webpage

Figure 38 Reference input and generated result for the flowchart-to-webpage generation showcase (case 4).