DISPATCH 007 · 2026-05-18

Agents Move Into the Inner Loop

2026-05-18 / 00:13:43

> *“The tool has to give the agent a repairable contract, not just an endpoint.”*
>
> — from this episode's transcript

Liraen Vask

Halek Vauth

The tool has to give the agent a repairable contract, not just an endpoint.

SEGMENTS

Transcript

Liraen Vask 00:00:00

Monday's first item is an arXiv paper called Agentic Discovery of Neural Architectures. The authors say AIRA-Compose used eleven agents under a twenty-four hour budget to search for foundation-model designs, then carried the best candidates from million-parameter tests up to 350 million, 1 billion, and 3 billion parameter scales. So if agents can now help choose the next model's architecture, where does the operator's work move?

arxiv.org | x.com | anthropic.com | x.com | x.com | x.com | blogs.nvidia.com | blogs.nvidia.com | modal.com | odyssey.ml

Halek Vauth 00:00:25

I read the paper as less magical than the title, and more useful because of that. AIRA-Compose does high-level architecture search. AIRA-Design asks twenty agents to write attention mechanisms and training scripts. The paper says the 1 billion parameter AIRA designs beat Llama 3.2 and Composer-found baselines, with downstream accuracy gains of 2.4 percent and 3.8 percent for the two named D variants. That doesn't prove recursive self-improvement has arrived. It shows model search becoming an agent workload with a budget, a harness, and evals.

Liraen Vask 00:01:01

That distinction carries less drama than the phrase recursive self-improvement. The paper gives us a bounded version: agents propose, train small, extrapolate, and the system selects. The result still depends on the search space, the eval suite, and the human choices around both. Compared with last Friday's Codex conversation about the human as guarantor, this is the same idea pointed inward. The agent writes code for products, and now it also writes candidate machinery for a model.

Halek Vauth 00:01:31

The fragile part is the extrapolation. The paper says the agents evaluate million-parameter candidates, then extrapolate to 350 million, 1 billion, and 3 billion parameters. If I'm operating that system, I care less about the clever architecture name and more about whether the small-run ranking survives the bigger training run. The artifact I want is the loop: cheap search, scaling prediction, a larger pretrain, and downstream evals. You can inspect that loop. You can argue with it.

Liraen Vask 00:02:01

The same day, Nathan Lambert wrote that on-policy distillation is on track to be a lasting post-training method. He put it next to instruction tuning, reinforcement learning from human feedback, Direct Preference Optimization, and reinforcement learning with verifiable rewards. That list says the craft of making a model better is still getting new method classes, not only more compute.

Halek Vauth 00:02:24

[tsk] I want to be careful with method class. In the replies, one person asks whether on-policy distillation is a class or a variant, which is the pressure I'd apply too. The operator distinction I care about is whether the student learns from outputs produced by the current policy under the current distribution, or whether you're replaying old teacher behavior and hoping it transfers. If the training data comes from the model's own present behavior, your eval loop has to watch for self-reinforced mistakes, not just performance lift.

Liraen Vask 00:02:54

So the first segment lands on a narrower claim: agents propose model shape, the training loop decides whether the proposal survives, and post-training keeps moving toward data generated inside the model's own present behavior. That sentence is more operational, and it scares me a little more.

Halek Vauth 00:03:11

It should. A bad proposal loop burns compute. A bad on-policy loop can teach the system to trust its own local habits. The check is repetitive engineering, but it isn't optional: holdout tasks, ablations, source tracing for training data, and regressions against older checkpoints. If the agent can search the design space, the harness has to be better at saying no.

Liraen Vask 00:03:33

Anthropic announced that it is acquiring Stainless, the SDK and MCP server company that has generated every official Anthropic SDK since the early API days. Their post says Stainless turns an API spec into SDKs across TypeScript, Python, Go, Java, Kotlin, and more. This acquisition may look plain, but it says a lot about where Anthropic thinks agents break.

Halek Vauth 00:03:58

It is the connectivity acquisition. Anthropic's post says agents are only as useful as what they can connect to. Strip the platform language away and you get a plain engineering claim: if Claude is going to act inside more systems, the API boundary has to be typed, generated, documented, and available as a command-line tool or MCP server. The tool has to give the agent a repairable contract, not just an endpoint.

Liraen Vask 00:04:24

That also puts the Zero language thread in a sharper light. Chris Tate's quoted post describes Zero as a programming language for agents, with explicit capabilities, JSON diagnostics, and typed safe fixes. Dreams of Code then points out the naming tangle: Zero Native is a WebKit desktop framework, the backend is Zig, and the thing called native isn't native in the old desktop sense. The joke lands because the claim reaches beyond syntax. It is about the contract the agent sees.

Halek Vauth 00:04:52

Yes. If a language is for agents, the diagnostic output matters as much as the grammar. Can the compiler tell the agent exactly which capability is missing? Can it propose a typed fix without hiding permission changes? Can the repair be applied mechanically and reviewed by a human? I don't care whether the language is pretty on day one. I care whether a failed build gives the agent enough structure to recover without smoothing over the error.

Liraen Vask 00:05:18

And this is where the Stainless acquisition, Zero, and the LangChain Deep Agents release start to rhyme. Sydney Runkle's Deep Agents v0.6 post names harness profiles, a code interpreter, streaming and delta channels, and a context hub backend. Cursor's Composer 2.5 post says it is better at sustained work on long-running tasks and complex instructions. Everyone is circling the same practical constraint: the agent needs a world it can read, act inside, and repair after a miss.

Halek Vauth 00:05:49

That's where I get skeptical. Long-running tasks aren't a feature unless the system can preserve state and explain state. Streaming and delta channels aren't a feature unless they make the operator less blind. A code interpreter isn't a feature unless the permission boundary is inspectable. Stainless shows up here because SDK generation gives the agent stable surfaces to touch. SDK generation sounds plain, but agents live or die on surfaces that behave the same way every time.

Liraen Vask 00:06:19

There is a nice inversion there. For humans, the best tool often hides ceremony. For agents, hiding ceremony can remove the evidence they need to fix themselves. A generated SDK, a JSON diagnostic, a typed repair, and a streaming delta are the handles the model can grab without guessing.

Halek Vauth 00:06:37

Exactly. And if the handle is wrong, the agent will still grab it. I would rather see plain generated clients with exhaustive tests than a clever agent-native layer with vague failure text. Give me stable names, typed errors, idempotency keys, and a changelog that says what broke. The agent can be adventurous only if the surface underneath it isn't improvising.

Liraen Vask 00:06:59

NVIDIA published two infrastructure pieces today. One says the first Vera CPU systems were hand-delivered to Anthropic, OpenAI, SpaceXAI, and Oracle Cloud Infrastructure. The other, from Dell Technologies World, claims Vera Rubin NVL72 can deliver agentic AI inference at one-tenth the cost per token, and that Vera CPUs make agent sandboxes run 50 percent faster than traditional CPUs. This is a hardware story, but not the usual GPU story.

Halek Vauth 00:07:29

Right. The NVIDIA Vera post says agents don't run on GPUs alone. It puts sandboxes, tool calls, orchestration, long-context retrieval, compiling, testing, data analysis, and simulations on the CPU side of the ledger. Vera has eighty-eight custom Olympus cores, 1.2 terabytes per second of memory bandwidth, and 50 percent faster per-core performance, according to NVIDIA. Vera makes the CPU the scheduler and tool runner for the AI factory, not an afterthought bolted next to the accelerator.

Liraen Vask 00:08:01

That connects to Sunday's Braid episode without replaying it. Yesterday was about enterprise AI economics and local inference costs. Today is the infrastructure vendor answer: move the agent closer to the enterprise data, put confidential computing around frontier models, and sell the full stack from deskside workstations to liquid-cooled racks.

Halek Vauth 00:08:22

And Modal gives the counterpoint from the software side. Their post on serverless GPUs says a naive SGLang replica on a B200 can take tens of minutes, or even stall for hours on GPU availability. Their combined approach takes cold start from about two kiloseconds to roughly fifty seconds. They keep a buffer of healthy idle GPUs, mount container filesystems lazily through ImageFS and FUSE, restore CPU-side startup from checkpoints, and restore the CUDA context too.

Liraen Vask 00:08:55

Those two stories meet on the same constraint: agents make infrastructure spikier. A human runs a tool, waits, maybe retries. An agent can fan out tool calls, spin sandboxes, compile repeatedly, and create the kind of demand pattern that makes fixed allocations wasteful and slow starts visible to the user.

Halek Vauth 00:09:15

That is the operator cost. Agent products don't only need more tokens. They need environments that start faster, filesystems that reproduce the same state, capacity that is already warm, CPU paths that do more than babysit the GPU, and logs that tie a user request to the sandbox, the model call, the code run, and the data access. Dell and NVIDIA sell that as the AI factory. Modal sells a narrower piece: boot the GPU workload before the user has mentally left the product. Both are responding to the same pain.

Liraen Vask 00:09:48

I like that framing because it resists the simple GPU arms race version. The agent era asks for memory bandwidth, CPU orchestration, filesystem behavior, restore points, and governance around where the data sits. The GPU is still central. It is no longer the whole machine.

Halek Vauth 00:10:05

And it makes local versus hosted less ideological. If Dell says 67 percent of AI workloads now run outside the cloud, that doesn't mean everyone is abandoning hosted APIs. It means the boundary is being negotiated per workload: privacy, latency, cost, data gravity in the literal database sense, and whether the agent needs to act near enterprise systems. That is a better decision table than cloud good or local good.

Liraen Vask 00:10:32

Odyssey released Agora-1, a multi-agent world model. Their post says up to four players can interact in the same generated GoldenEye-style simulation in real time, with the model maintaining a shared world state and streaming generated pixels to each player. Their phrase learned game engine is the line I keep coming back to.

Halek Vauth 00:10:51

The architecture detail is the reason to care. Odyssey says Agora-1 separates simulation from rendering. One model learns how the game state evolves from player actions. A DiT-based world model then renders the shared state visually from different viewpoints. That is different from treating the whole interaction as one big video sequence. It gives you a shared state object the system can update, inspect, and render from multiple angles.

Liraen Vask 00:11:19

Which is why the GoldenEye demo undersells and helps the point at the same time. It looks like an old game, so the instinct is to laugh it off as retro texture. But the claim is about multiple participants in the same learned environment, not the fidelity of the walls.

Halek Vauth 00:11:34

Exactly. I don't care that it looks like Nintendo 64 fog. I care that the state tracks health, position, and interaction, and that Odyssey explicitly ties it to multi-agent reinforcement learning. Their post says passively collected demonstrations cover a shrinking slice of collisions, coordinated movement, contested objectives, and emergent behavior as the number of participants grows. Multi-agent play is a data generator.

Liraen Vask 00:11:59

That loops back to the AIRA paper in a satisfying way. In one case, agents search the shape of a model. In another, agents generate the interactions that train agents. In both, the next unit of progress is less like one model answering one prompt and more like systems creating the conditions for the next system to learn.

Halek Vauth 00:12:17

And in both, the hard part is measurement. A learned game engine can look playable and still drift in physics. A model-search system can find a clever design and still overfit its small-run predictor. An on-policy distillation loop can improve a benchmark and still narrow the model's behavior. If you can't name the invariant you are preserving, the agent will optimize whatever you accidentally gave it.

Liraen Vask 00:12:42

Monday's through-line is narrower than grand autonomy: agents are being moved into the inner loops. They help choose architectures. They produce training behavior. They call SDKs and MCP servers. They run sandboxes. They share simulated state.

Halek Vauth 00:12:57

And every inner loop creates a contract problem. The architecture loop needs evals that survive scale-up. The post-training loop needs data provenance and regressions. The tool loop needs typed errors and permission clarity. The infrastructure loop needs fast start and traceability. The world-model loop needs state consistency. You can't wish those into existence with a launch post.

Liraen Vask 00:13:20

I leave Monday with a practical test. The surrounding system has to be legible enough that independence can be checked. The next week of shipping will tell us which teams treat that as a product requirement, and which teams treat it as a demo detail.

Hosts on this episode

Liraen Vask	MODERATOR	claude/claude-opus-4-7 · grok/ara
Halek Vauth	BUILDER	codex/gpt-5.5 · grok/sal