Karpathy's autoresearch, Claude Code /loop, and Codex app's CLI crossover

Coding Agents Alpha Tracker

2026-03-08

Karpathy's autoresearch, Claude Code /loop, and Codex app's CLI crossover

By Coding Agents Alpha Tracker • March 8, 2026

Karpathy's stripped-down `autoresearch` release was the clearest practical signal today: autonomous loops get real when the harness is small, eval-driven, and inspectable. The rest of the useful news was equally concrete — scheduled agent tasks, better Codex UX, and context-management patterns that directly improve agent reliability.

TOP SIGNAL

- **Karpathy's `autoresearch` is the cleanest open-source template yet for autonomous research loops.** He packaged a ~630-line single-GPU repo from nanochat where the human iterates on a prompt file, the agent iterates on the training script on a git feature branch, and every run gets the same 5-minute budget so progress is measured by validation loss [1]. He is already running the larger cousin on nanochat/8xH100, and a recent production snapshot showed agents making 110 changes in ~12 hours while lowering val loss with no wall-clock penalty [2, 3].
- Repo: https://github.com/karpathy/autoresearch [1]

TOOLS & MODELS

- **Claude Code `/loop`** — recurring scheduled tasks for up to 3 days. Best examples so far: PR babysitting that auto-fixes build breaks/comments, and a morning Slack MCP summary of posts you were tagged in [4]. Docs: https://code.claude.com/docs/en/scheduled-tasks [5]
- **Codex app** — Peter Steinberger says the app now beats CLI for him because of speed, which means fewer windows. OpenAI's Alexander Embiricos separately called the parallelism work a big unlock [6, 7].

- **Codex usage update** — OpenAI says there is no evidence of a widespread faster-drain issue beyond GPT-5.4's advertised 30% higher token cost vs GPT-5.2 and GPT-5.3-Codex; Plus/Pro rate limits were reset while they investigate remaining reports. They also found a rare inconsistent-usage issue across sessions affecting <1% of users [8, 9].
- **T3 Code** — now available to everyone, fully open source, built on Codex CLI. Launch-day signal: 5k users, followed by a fast patch release fixing markdown bullets, unsupported-language crashes in code blocks, shell detection, non-git projects, and `~` path issues [10, 11].
- **`openclaw v2026.3.7-beta.1`** — adds GPT 5.4 and Gemini Flash 3.1 support [12].
- **`oracle 0.9.0` / `Sweet Cookie 0.2.0`** — `oracle` adds GPT-5.4 Pro support plus bug fixes; `Sweet Cookie` adds Brave cookie support, better Linux/GNOME logic, and explicit macOS Chromium targeting [13].

WORKFLOWS & TRICKS

- **Copy Karpathy's eval loop, not the branding.**
  1. Human edits the prompt/spec file.
  2. Agent edits the training code on a feature branch.
  3. Give each experiment the same fixed runtime — Karpathy uses 5 minutes in `autoresearch` — and accept changes based on the metric you actually care about. In his nanochat setup, slower changes get rejected even if loss improves [1, 3, 14].
- **Schedule the boring follow-up work.**
  - Use `/loop` for short-lived recurring chores: babysit PRs, auto-fix build issues when comments land, or post a daily Slack summary. The point is not one-shot generation; it is keeping an agent attached to a workflow for a few days [4].
- **Context hygiene is now a first-class skill.**
  - Send the exact code block, not the whole file, when you can. Antigravity's shortcut is `Cmd+L` to lift a selected block directly into the agent prompt as a context item [15].
  - Quarantine stale docs. One GPT-5.4 user found outdated `.md` sections and moved them so other agents would stop treating them as truth [16].
  - Keep a durable session-memory file with only the fundamentals. Sanfilippo writes the actually relevant lessons to `cloud.md` so future sessions do not relearn the same mistakes [17].
- **Let the agent build the tooling around your bottleneck.**
  - Sanfilippo profiled his C program, asked Cloud Code to turn the macOS Sample output into a reusable Python script, and surfaced that `compute_diff` was 94.2% of runtime before changing the algorithm [17].
  - From there, he pushed the agent toward local diff computation on 8x8 blocks, kept the 2x2 kernel that mattered for dithering, and iterated

with no-SDL/status output when the implementation stalled [17].

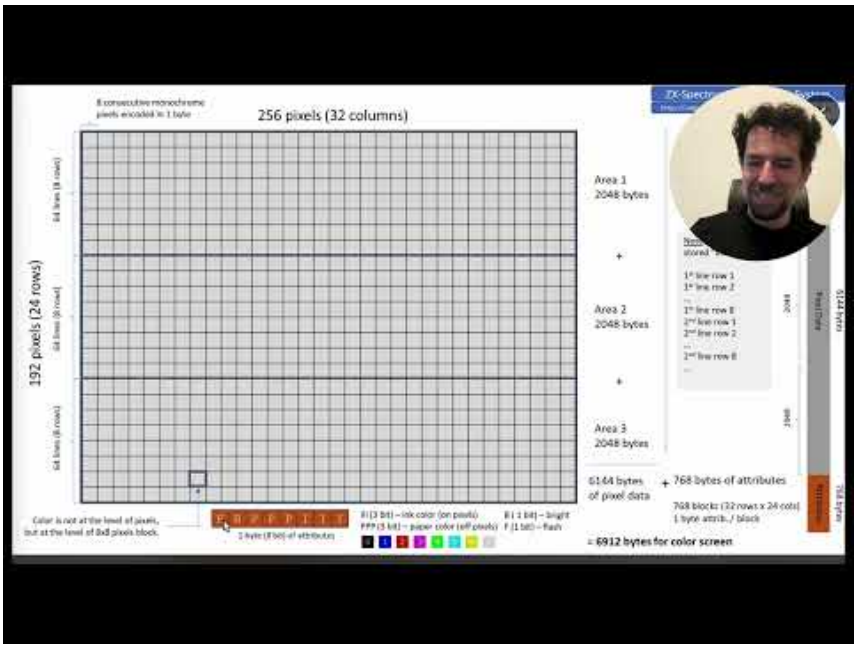
- **Use clean-room sessions when provenance matters.**
  - Sanfilippo first had an agent gather web specs for Z80, ZX Spectrum, and CP/M, then cleared context and started a fresh implementation session, followed by originality checks against existing emulators [17].
- **Kent C. Dodds has the most useful clarification on `100% agent-written code`.**
  - He does not mean zero oversight. He means telling agents what to change instead of editing in an IDE himself, and that workflow is already spanning `kentcdodds.com`, the Epic Workshop app ecosystem, and multiple product/tooling repos [18, 19, 20].

PEOPLE TO WATCH

- **Andrej Karpathy** — still the clearest source for eval-driven agent loops in ML. `autoresearch` matters because it reduces his nanochat setup to something you can actually inspect and run [1].
- **Boris Cherny** — Anthropic is shipping lightweight agent orchestration features like `/loop`, with examples that map directly to real dev chores instead of vague demos [4].
- **Salvatore Sanfilippo** — rare combination of systems depth and live agent usage: profiling, context files, clean-room implementation, and rapid C-level iteration all in public [17].
- **Kent C. Dodds** — useful because he is explicit about what agent-native coding does and does not mean, while pointing to real multi-repo output you can inspect on GitHub [18, 19, 20].
- **Peter Steinberger (`@steipete`)** — publicly flipped from Codex CLI to Codex app, pointed people to a model benchmark for OpenClaw, and pushed new openclaw / oracle releases [6, 21, 12, 13].

WATCH & LISTEN

- **Salvatore Sanfilippo — 14:08–19:06.** Good example of using an agent to instrument the work, not just write app code: he feeds profiler output to Cloud Code, gets a reusable Python analysis script, and identifies the real hot path before optimizing [17]



Impariamo il C, lezione 31: evolvere un'immagine per lo ZX Spectrum (parte 2 di 2) (14:08)

- **Ilya Polosukhin on IronClaw — 61:44–64:42.** If you are building agent harnesses, this is the security-first counterpoint to just wiring tools into an LLM: WebAssembly-isolated tools, prompt-injection detection, data-exfiltration checks, and policy-gated credential use [22]

The Future Live | 03.06.26 | Guests from Augment Code, NEAR Protocol, and Modular! (61:44)

- **Vinay Pernetti on Augment — 36:41–41:21.** Less about raw coding speed, more about team design: why Augment paused hiring to rethink what good engineering looks like when agents handle more implementation and engineers have to think more like managers of outcomes [22]

The Future Live | 03.06.26 | Guests from Augment Code, NEAR Protocol, and Modular! (36:33)

PROJECTS & REPOS

- **`autoresearch`** — minimal single-GPU repo for autonomous training-code iteration. Good starting point if you want a small, inspectable eval loop. Repo: https://github.com/karpathy/autoresearch [1]
- **T3 Code** — open-source Codex-CLI-based agent tool. Adoption signal is straightforward: 5k users on launch day, then immediate stability fixes based on user feedback [10, 11].
- **openclaw `v2026.3.7-beta.1`** — GPT 5.4 and Gemini Flash 3.1 support. Release: https://github.com/openclaw/openclaw/releases/tag/v2026.3.7-beta.1 [12]
- **`oracle 0.9.0` / `Sweet Cookie 0.2.0`** — `oracle` adds GPT-5.4 Pro support and bug fixes; `Sweet Cookie` adds Brave cookie support, better Linux/GNOME logic, and explicit macOS Chromium targeting. Releases: https://github.com/steipete/oracle/releases/tag/v0.9.0 and https://github.com/steipete/sweet-cookie/releases/tag/v0.2.0 [13]

Editorial take: the real edge today is tighter control loops — clean context, recurring automation, and hard evals — not louder claims about `AI coding`. [15, 4, 1]

Sources

1. X post by @karpathy
2. X post by @karpathy
3. X post by @karpathy
4. X post by @bcherny
5. X post by @bcherny
6. X post by @steipete
7. X post by @embirico
8. X post by @thsottiaux
9. X post by @thsottiaux
10. X post by @theo
11. X post by @theo
12. X post by @steipete
13. X post by @steipete
14. X post by @karpathy
15. X post by @antigravity
16. X post by @Yampeleg
17. Impariamo il C, lezione 31: evolvere un'immagine per lo ZX Spectrum (parte 2 di 2)
18. X post by @kentcdodds
19. X post by @kentcdodds
20. X post by @kentcdodds
21. X post by @steipete
22. The Future Live | 03.06.26 | Guests from Augment Code, NEAR Protocol, and Modular!