Parallel agent work hardens: Claude Code reviews PRs, Codex fans out tasks, Karpathy logs an 11% gain

Coding Agents Alpha Tracker

2026-03-10

Parallel agent work hardens: Claude Code reviews PRs, Codex fans out tasks, Karpathy logs an 11% gain

By Coding Agents Alpha Tracker • March 10, 2026

Parallelism—not just better raw models—was the clearest coding-agent signal today. Karpathy showed measurable gains from autonomous experiment loops, Anthropic shipped multi-agent PR review, and practitioners shared concrete fan-out, skills, and documentation patterns that make these systems reliable.

TOP SIGNAL

Parallelism is becoming the real lever. Karpathy's `autoresearch` loop ran ~700 autonomous experiments, found ~20 additive changes that transferred from smaller to larger nanochat models, and cut "Time to GPT-2" from 2.02h to 1.80h (~11%) [1]. Anthropic productized the same pattern with Claude Code's new **Code Review**, which spawns a team of agents on every PR because internal code output per engineer is up **200%** and review became the bottleneck [2, 3]. Francesco reports the practitioner-side version: switching to Codex and parallelizing more aggressively made February his most productive month ever, nearly **4x** August [4].

TOOLS & MODELS

- **Claude Code — Code Review:** When a PR opens, Claude dispatches a team of agents to hunt for bugs [2, 3]. Anthropic says they built it for themselves first because code output per engineer is up **200%** this year and review became the bottleneck; Boris Cherny says it catches bugs

he would have missed, and Alex Albert says it has been a game changer internally [3, 5].

- **Codex xhigh reasoning:** Francesco's Typefully setup gets the first prompt right **95%** of the time, and his output jumped nearly **4x** once he switched to Codex and pushed more work in parallel [4].
- **Harness > raw model:** Dylan Patel says the *same* Claude 4.6 model performs very differently in Claude Code vs Cursor agent mode, and his team mostly prefers Claude Code because of the harness [6]. Simon Willison and Kent C. Dodds report that, with a good agent harness plus repo docs/examples, agents handle private or brand-new tools just fine, including Remix 3 [7, 8].
- **Long-running loop reliability check:** In a public `autoresearch` test, Claude Opus 4.6 (high) ran **12+ hours** and completed **118 experiments**, while GPT-5.4 xhigh stopped after **6** despite a `LOOP FOREVER` instruction [9]. Karpathy says Codex currently does not work with `autoresearch` as configured and that he prefers interactive `tmux` sessions over headless loops [10].
- **Cloud-only dissent:** Theo says T3 Code will not support local models because he does not think they can do meaningful engineering work, and because one of the product's advantages is running lots of work in parallel [11, 12].

WORKFLOWS & TRICKS

- **Copy Francesco's low-babysitting Codex loop**
  1. Put each task in **Linear** [4].
  2. Use **Git worktrees** so agents stay off `main` [4].
  3. Open **Ghostyy**, paste a Linear task ID, then repeat for more tasks [4].
  4. Review PRs while other agents keep working [4].
  - His claim: Codex fits this parallel workflow better than Claude Code because it needs less steering and feedback [4].
- **Run cheap-to-expensive research loops**
  1. Let agents explore on a smaller model first [1].
  2. Optimize for a metric you can evaluate cheaply, or for a smaller-network proxy [1].
  3. Promote only promising ideas to larger scales [1].
  4. Keep only changes that transfer additively; Karpathy's round 1 found ~20 that did [1].
  - He says `autoresearch` is best treated as a **recipe/idea** you hand to your agent, not something you use directly [13].
- **Teach the agent the stack inside the repo**
  - Kent says agents had **zero problem** with Remix 3 once the repo had the right documentation [8].
  - Simon's trick is explicit: tell the agent to read `--help` output for unfamiliar tools before it starts solving the task [7].

  - Emerging pattern: projects are now shipping official **skills** repos to package this knowledge for agents [7].
- **Turn specialist knowledge into shared skills**
  - Dylan Patel says his team keeps reusable skills in internal GitHub, so a specialist's workflow—like data-center permit analysis—can be reused by non-experts [6].
  - He also describes a non-programmer hedge-fund user teaching Claude Code a tone-analysis skill from books, then running it across earnings transcripts without writing code [6].
- **Auto-ship low-risk work; gate the risky stuff**
  1. Edit inside the product's **designer mode** [14].
  2. Hit **Launch Agent** to ship via Cursor Cloud Agents and Workflow Automations [14].
  3. Stop for manual review only when the risk matrix says to—e.g. database schema migrations [14].
  - Geoffrey Huntley's framing is good: stay **on the loop, not in the loop** [14].
- **If you're building agents, evals first beats prompt-tweaking**
  - LangChain starts by defining success scenarios, then runs rule-based checks plus an LLM judge in CI [15].
  - Every human action becomes training signal: send, edit, and cancel are logged against traces and reused later [15].

PEOPLE TO WATCH

- **Andrej Karpathy** — still the clearest public source on eval-driven agent loops. Today's reason: ~700 autonomous experiments, ~20 additive fixes, an ~11% nanochat speedup, plus blunt feedback on where headless loops break [1, 10].
- **Dylan Patel** — unusually concrete on production agent use: real spend numbers, same-model harness differences, shared skills, and non-programmer adoption inside his firm [6].
- **Francesco (Frank Dilo) / Romain Huet** — strongest public Codex workflow today: nearly 4x output, 95% first-prompt hit rate, and a task fan-out system you can copy tomorrow [4, 16].
- **Simon Willison + Kent C. Dodds** — good antidote to the "agents only work on boring stacks" meme. Their shared point: docs, examples, and harness quality matter more than whether the framework was in the training data [7, 8].
- **swyx** — worth tracking if long sessions keep degrading. He keeps open-sourcing tooling around Claude compaction and session hygiene instead of just complaining about it [17, 18, 19].

WATCH & LISTEN

- **Dylan Patel on "coding tools" vs agent orchestration systems — 32:34-36:34.** Best clip of the day if you still think Claude Code or Codex are just for programmers: he walks through reusable skills, non-programmer workflows, and why the category is bigger than code generation [6].

Dylan Patel: They Don't See It Coming (32:33)

- **Dylan Patel on cost shock vs output — 4:20-5:46.** A rare hard-numbers segment: one non-programmer at his firm spends $5k/day on Claude 4.6 fast 1M context, one engineer spent $8k in a single go, and the company still accepted the burn because the output justified it [6].

Dylan Patel: They Don't See It Coming (4:19)

PROJECTS & REPOS

- autoresearch — Karpathy says this is a **recipe/idea**, not a turnkey app. The latest proof point is his nanochat round 1: ~700 autonomous experiments surfaced ~20 additive improvements and cut time-to-GPT-2 by ~11% [13, 1].
- nanochat round-1 commit — concrete patch set from that pass: QKnorm scaler, value-embedding regularization, less conservative banded attention, AdamW beta fixes, weight-decay tuning, and initialization tuning [1].
- claude-compaction-viewer — swyx open-sourced this after repeated bad Claude Code compactions, and noted it could likely extend to Codex compactions too [17].
- Official skills repos are now showing up from maintainers, not just users: Remotion, Supabase, Vercel, and Prisma [7].

Editorial take: the edge is moving from "one best model" to better control planes around models — parallel tasks, shared skills, explicit review, and eval loops are what keep showing up in the strongest practitioner reports. [3, 4, 6, 15]

Sources

1. X post by @karpathy
2. X post by @claudeai
3. X post by @bcherny
4. X post by @frankdilo
5. X post by @alexalbert___
6. Dylan Patel: They Don't See It Coming
7. Perhaps not Boring Technology after all
8. X post by @kentcdodds
9. X post by @Yuchenj_UW
10. X post by @karpathy
11. X post by @theo
12. X post by @theo
13. X post by @karpathy
14. a sneak preview behind an embedded software factory. I suspect rapid application dev is back
15. How we built LangChain's GTM Agent
16. X post by @romainhuet
17. X post by @swyx
18. X post by @swyx
19. X post by @swyx