

We should be more tired than the model

May 28 2026

Lately, I've been feeling like I'm losing control over the code I write when I work with agentic code generation.

When I finish an agentic session, I get all the outward signs of having written code, but none of the internal processes that happen when we [write code by hand](#).

As a quick primer, the human brain has several types of memory, short-term, working, and long-term. Short-term memory gathers information temporarily and processes it quickly, like RAM. Long-term memory includes things you've learned previously and tucked away, like database storage. Working memory takes the information from short-term memory and long-term memory and combines them to synthesize, or process the information and come up with a solution.

When we're working on code, (and by working on, we mean most often reading someone else's code) all of these processes are going on in our brain simultaneously to try to help us make sense of the programming environment.

It's not surprising. Code generation, in its default mode, is [antithetical to skill retention](#), particularly because its UX affordances are reminiscent of a slot machine's: you pull the lever, you get a reward (a solution to your coding problem.) In some ways, we've replaced the social media feed with a stream of tokens, and I look forward to reading those papers in ten years.

It really does take extra concerted effort to move from just generating answers to using the tool deliberately. One thing [Oz suggested](#) when I posted on X that

#engineering #data science
#career advice #code
#machine learning #data
#software craft #python
#learning #engineering culture
#talks #web dev
#deep learning
#distributed systems #community
#open source #search
#writing #computing history
#normcore #product culture
#recsys #social #embeddings
#art #llms #shipping
#type systems #creativity
#golang #snippets #agentic
#algorithms #aws #local llms
#ml #prod #teams

I felt like I came away from an agentic session with a brain fog was rewriting portions of code myself.

Inspired by that advice, the paper, [thoughts on slowing down](#), and using [AI to write better code more slowly](#), and [Mitchell's adoption journey](#), I've been working on using the tool more deliberately and adding friction back into development.

Here's what's worked for me so far:

- Writing the initial implementation myself and asking the agent to review the code, then going through comment by comment and manually making the changes
- Using the agent to keep asking questions about pieces of the code I don't understand instead and pull up relevant documentation and PRs.
- Asking the agent to think about implementing two approaches and choosing between them and then critiquing the other approach
- Discussing an agent's proposed implementation with another person instead
- Starting to use the agent only after I've spent 20 minutes on the problem
- Going back and reading books and academic papers
- Reimplementing fundamental data structures

All of these negate the supposed speed up effects of LLM-generated code in the short-term by adding friction, and yet, in the longer term, make me better at using the tool, because they solidify my own foundation instead of the foundation models'.

We should be more tired than the model.

[#learning](#) [#llms](#)

[← Tagging my blog posts with BERTopic and Running local models is good LLMs](#) [now →](#)

Made with [Hugo Ɔ•••? Bear](#)