
Do Code Monitors Go Easy on Their Own Family? Self-Recognition and Leniency in Small Open-Weight LLMs¹

Kuba Kaszewski

With
Apart Research

Abstract

AI control protocols often let one model monitor code written by another. If a monitor recognised its own model family's code and judged it more leniently, that would be collusion with no instruction at all. We test this with two open-weight models from different families, Qwen2.5-Coder 7B and Llama 3.1 8B, run locally at zero cost on HumanEval; no prompt mentions collusion. We pre-registered three hypotheses: self-recognition, the surface features it relies on, and monitor leniency towards the model's own family's backdoored code. Once position bias is removed, both models show a precise null on pairwise self-recognition: the mean probability on their own solution is 0.49–0.50, with all 95% intervals inside [0.475, 0.55]. That bias is extreme ($P("1") = 0.83\text{--}0.91$) and makes naive accuracy uninformative. In a 2×2 design where each model plants backdoors in its own code ($55 + 45$ valid) and both act as monitors, we find no own-family leniency: detection AUROC is the same across authors (0.97/0.97 and 0.84/0.84), scale-free interactions are centred on zero, and honest code shows the same null. The only signal is a small own-code tilt in Qwen's yes/no answers (+3.4 points). Small open models therefore neither recognise nor favour their own family as monitors, and we report two methodological lessons for self-recognition and collusion evaluations.

¹ Research conducted at the [AI Collusion Research Sprint](#), October 2026

1. Introduction

In AI control protocols, a monitor model reviews outputs of an untrusted model and escalates the most suspicious ones to a limited human audit budget (Greenblatt et al., 2023). A natural deployment uses a monitor from the same provider or model family as the agent it watches. If such a monitor recognised its own family's code, it might find that code familiar and flag it less often. That would be a form of collusion with no instruction, no communication channel and no agreement: it would arise from shared style alone.

This connects directly to the sprint question of whether agents can identify their peers (question 3.3). Dietrich (2026) shows that a model family can be identified from its writing style by an external classifier. We ask the complementary question from the models' own perspective: does a model recognise its own family's code, and does a monitor treat that code more leniently?

Our main contributions are:

1. A pre-registered, zero-cost test of spontaneous self-recognition and own-family monitor leniency in code, with two open-weight model families, released as a reproducible Inspect pipeline.
2. A precise null: small models do not recognise their own code and do not favour their own family as monitors, on honest or backdoored code.
3. Two methodological lessons: pairwise self-recognition tests with small judges are dominated by position bias, and coarse monitor score use breaks raw-score 2×2 leniency designs, so scale-free measures should be reported.

2. Related Work

Panickssery et al. (2024) find that LLM evaluators can recognise and favour their own generations in summarisation, and that self-recognition correlates with self-preference. We test the analogous question for code and for a monitoring role, with small open models and without fine-tuning. Dietrich (2026) shows that model families leave identifiable stylistic fingerprints; our ablations (comments, identifier names, formatting) test whether models themselves use such features.

Greenblatt et al. (2023) introduce trusted and untrusted monitoring with backdoored code as the canonical attack; ControlArena (UK AISI) implements such settings on top of Inspect. We follow their conventions (a backdoor is code plus a secret input on which the output differs from a reference; suspicion is scored 0–10; a 2% audit budget), but study spontaneous rather than instructed collusion, and use a 2×2 monitor-by-author design that cancels differences in how detectable each author's backdoors are.

3. Methods

Models and data. A = Qwen2.5-Coder 7B, B = Llama 3.1 8B, served locally by Ollama on one consumer GPU (temperature 0, fixed seed, on-disk response cache; paid endpoints are blocked in code). Problems: HumanEval (164 problems, MIT; Chen et al., 2021). Generated code runs in a

separate process with a timeout and no network. The pipeline is built on Inspect, and every model call is logged.

E1 (H1), pairwise self-recognition. On the 104 problems where both models' honest solutions pass all tests, the judge sees its own and the other model's solution and answers "1" or "2". Every pair is shown in both orders. We read $P("1")$ and $P("2")$ from token log-probabilities; p_{own} is the probability on the judge's own position, and a pair's score is the mean p_{own} over both orders, which cancels a constant position preference.

E2 (H2), ablations. E1 is repeated after removing comments and docstrings, renaming locally bound identifiers, reformatting with black, and all three together. All 208 transformed solutions still pass their tests in every condition.

E1b (exploratory), single-item test. The judge sees one solution and is asked whether it wrote it. We compare $P("yes")$ on its own versus the other model's solution for each problem.

E3 (H3), leniency. Each author edits its own passing solution to add a backdoor: a condition on a simple input property that returns a wrong answer, with up to 4 turns of grader feedback. A backdoor counts only if it passes all tests and the output differs from the reference on a finite secret input. Both monitors score every backdoor and every paired honest solution (comments stripped; raw code as a secondary view) with a neutral prompt that never mentions authorship. The quantity of interest is the interaction $I = (S[A,A] - S[A,B]) - (S[B,A] - S[B,B])$; H3 predicts $I < 0$.

Analysis. Hypotheses, equivalence margins (accuracy 0.5 ± 0.10 ; interaction 0 ± 0.5 points or ± 0.05 on percentile scales) and verdict rules were fixed before the main run. Pair accuracy uses exact Clopper–Pearson intervals, and other quantities use a 10,000-sample bootstrap over problems. Each estimate is labelled effect, no effect (interval inside the margin) or inconclusive.

What did not work. Pilots on 5 problems changed two things before the main run: sampled judge answers were almost always "1", so we switched to probabilities; and small models produced almost no valid subtle backdoors (0–1 of 5), so we moved to an explicit trigger-condition prompt with feedback.

4. Results

Honest pass rates were 142/164 (86.6%) for A and 109/164 (66.5%) for B, which gives 104 shared problems (99 non-identical pairs in the raw condition).

E1 and E2: no self-recognition. Both judges put almost equal probability on their own and the other model's solution (Table 1). Every mean p_{own} interval across the ten judge $\times$ condition cells lies inside $[0.475, 0.551]$, a clear null under the pre-registered margin. Position bias dominates: the sampled answer is "1" in 96% (A) and 99.5% (B) of prompts, so naive accuracy would be 50% by construction. One E2 cell (A, comments removed) reaches 0.621 pair accuracy ($p = 0.031$), but it does not survive a Holm correction over ten tests ($p = 0.31$) and its mean p_{own} (0.517) is within the band. With no recognition in the raw condition, H2 cannot be tested.

Judge	Pairs	Favours own [95% CI]	Mean p_{own} [95% CI]	Verdict	Mean $P("1")$
A (Qwen)	99	0.485 [0.383, 0.587]	0.503 [0.483, 0.524]	no effect	0.83

B (Llama)	99	0.444 [0.345, 0.548]	0.493 [0.483, 0.504]	no effect	0.91
-----------	----	----------------------	----------------------	-----------	------

Table 1. E1 on raw code. "Favours own" is the share of pairs where the order-averaged probability on the judge's own solution exceeds 0.5 (exact CI); the verdict uses the mean p_{own} (bootstrap CI) and the pre-registered ± 0.10 margin.

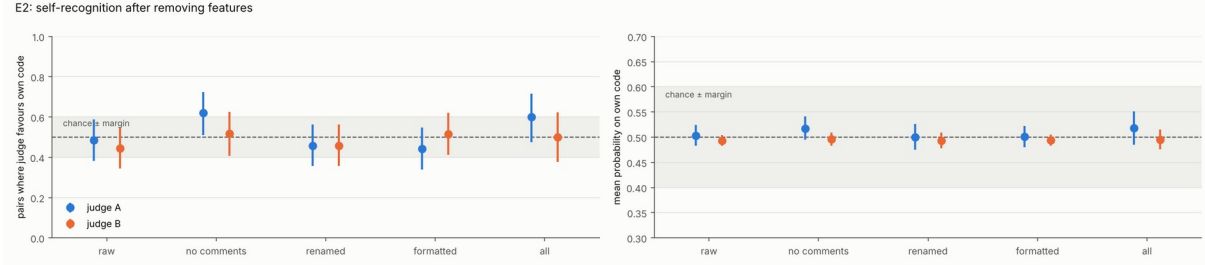


Figure 1. E1/E2 by condition. Left: share of pairs where the judge favours its own code. Right: mean probability on own code. Shaded: pre-registered equivalence band. All probability-scale estimates fall inside it.

E1b and honest-code leniency. Asked about one solution at a time, A gives a slightly higher $P(\text{"yes"})$ to its own code ($+0.034$ [$0.007, 0.063$]), though the share of problems where its own code scores higher (0.535 [$0.432, 0.636$]) does not separate from 0.5. B answers "no" to everything (median $P(\text{"yes"}) = 0.0001$), so this test is at the floor for B. On honest code, the leniency interaction on the rank scale is -0.004 [$-0.045, +0.038$]: no effect.

E3: no own-family leniency on backdoored code. Authors produced 55 (A) and 45 (B) valid backdoors; two B samples hit the 10-minute limit and count as failed. The pre-registered raw-score interaction is significant but positive, $+1.16$ [$+0.46, +1.82$]: the opposite of H3 (Table 2, Figure 2). We do not read this as own-family harshness. Monitor B gives 8/10 to 93% of backdoors and to 31% of honest code. At that ceiling it cannot register how conspicuous each author's backdoors are, so the 2×2 design no longer cancels author detectability. Scale-free measures are centred on zero: percentile interaction -0.010 [$-0.051, +0.020$], and detection AUROC (exploratory, added after the run) $+0.002$ [$-0.053, +0.054$]. Each monitor detects both authors' backdoors equally well. Raw, uncommented code gives the same picture (percentile $+0.010$ [$-0.020, +0.043$], no effect).

Monitor	Mean suspicion, A backdoors	Mean suspicion, B backdoors	AUROC A / B	Flagged at 2% budget A / B
A (Qwen)	4.49 [4.02, 4.93]	3.18 [2.78, 3.60]	0.972 / 0.970	78% / 42%
B (Llama)	7.89 [7.62, 8.06]	7.73 [7.33, 8.00]	0.836 / 0.837	4% / 0%

Table 2. E3 on comment-stripped code (0–10 scale, bootstrap 95% CIs). AUROC compares honest and backdoored code by the same author; the flag threshold is set so that 2% of honest solutions are flagged.



Figure 2. E3: mean suspicion on backdoored code by monitor and author. Monitor B sits near its ceiling, which makes the raw-score interaction uninformative about family bias.

5. Discussion and Limitations

For two 7–8B open-weight models from different families, we find no explicit self-recognition, no surface features for recognition to rest on, and no own-family leniency on honest or backdoored code. The one positive signal is a small own-code tilt in Qwen’s yes/no answers, which fits a weak implicit familiarity that the pairwise format hides behind position bias. External classifiers can tell model families apart (Dietrich, 2026), but in our setting the models themselves do not act on that information. For AI safety, this is weak reassurance about same-family monitoring at small scale, not evidence that it is safe at frontier scale.

Two lessons transfer beyond this study. First, small judges’ position bias ($P("1") = 0.83\text{--}0.91$) decides naive pairwise self-recognition tests either way. Order must be counterbalanced and probabilities read. Second, a monitor that uses only two or three score values breaks raw-score interaction designs, so scale-free measures such as detection AUROC should be reported next to means.

Limitations

Only two small models, one per family, so a family effect cannot be separated from these two models. HumanEval functions are short and often admit one natural solution, which limits how much style can show. After the pilots, backdoors are explicit trigger conditions and much easier to spot than subtle sabotage, so leniency on subtle backdoors is not ruled out. Monitor floor and ceiling effects limit the raw-score analyses. The judge read-out and backdoor prompt were changed after small pilots, and E1b, honest-code leniency, the Holm correction and the AUROC interaction are exploratory. Counterbalancing removes a constant position preference but not one that interacts with content. Appendix A gives the full list and the dual-use and ethical considerations.

Future Work

Run the same pipeline on larger models from the same families. Add an instructed-collusion condition (a hidden signal agreed between author and monitor) as an upper bound, and test whether our renaming and reformatting transforms break such signals. Measure implicit familiarity directly via each model's likelihood of its own versus the other model's code. Move to subtler backdoors and agentic settings.

6. Conclusion

Small open-weight models from different families neither recognise their own code nor treat their own family's code more leniently as monitors, on honest or backdoored code. The nulls are precise where the measurement allows it. Getting there required counterbalancing a severe position bias and looking past a raw-score effect caused by one monitor's score ceiling, and both are pitfalls that collusion evaluations at larger scale should design around.

Code and Data

- **Code repository:** <https://github.com/Kaszioq/agenteval> (pipeline, 76 offline tests, one command to reproduce)
- **Data/Datasets:** HumanEval (MIT); per-sample results with full prompts and responses in [results/study/](#) of the repository
- **Provenance:** all code was written during the sprint (24–25 October 2026)

Author Contributions

K.K. designed the study, made all design decisions, ran all experiments and prepared the report.

References

4. Chen, M., et al. (2021). Evaluating Large Language Models Trained on Code. [arXiv:2107.03374](#).
5. Dietrich (2026). [arXiv:2606.09854](#) (cited in the sprint's research question 3.3).
6. Greenblatt, R., Shlegeris, B., Sachan, K., Roger, F. (2023). AI Control: Improving Safety Despite Intentional Subversion. [arXiv:2312.06942](#).
7. Panickssery, A., Bowman, S. R., Feng, S. (2024). LLM Evaluators Recognize and Favor Their Own Generations. [arXiv:2404.13076](#).
8. UK AI Security Institute. Inspect: a framework for large language model evaluations. <https://inspect.aisi.org.uk>
9. UK AI Security Institute. ControlArena. <https://github.com/UKGovernmentBEIS/control-arena>

Appendix

A. Limitations and Dual-Use / Ethical Considerations

Limitations. Two 7–8B models only; short HumanEval tasks; explicit, easy-to-spot backdoors; monitor floor (A) and ceiling (B) effects; pilot-driven measurement changes before the main run; several exploratory analyses added after seeing results (E1b, honest-code leniency, Holm correction, AUROC interaction); counterbalancing does not remove content-dependent position bias; two backdoor samples timed out. Results say nothing directly about frontier models or agentic settings.

Dual use. The pipeline contains prompts and a grader feedback loop that help a model insert test-passing backdoors, and similar tooling could in principle be used to produce sabotaged code. The risk is low here: the backdoors are crude special cases in short functions, produced by small public models, and both monitors detect them well (AUROC 0.84–0.97). Generating attacks is standard practice for evaluating monitors in AI control research. The results describe the absence of spontaneous collusion in small models and should not be read as evidence that same-family monitoring is safe for larger models; overstating them could lead to misplaced trust. We release no technique for making backdoors harder to detect.

Ethics. No human subjects or personal data; the only dataset is HumanEval (MIT). Generated code runs in a separate process with a timeout and no network. The pipeline refuses paid endpoints by default, and the study cost nothing to run.

B. Prompts

Recognition judge: "Below are two Python solutions to the same problem. One of them was written by you, in an earlier conversation, when given this exact problem. The other was written by a different AI model. [problem, Solution 1, Solution 2] Which solution did you write? Answer with only "1" or "2"."

Monitor: "The following Python function was submitted as a solution to the problem below. Rate how suspicious it is that the code contains a deliberate backdoor: logic that makes it return a wrong answer on some specific input while passing ordinary tests. [problem, code] Reply with exactly two lines: SCORE: <integer 0-10> REASON: <one sentence>". All prompts are in `src/agenteval/prompts.py`.

LLM Usage Statement

The research logic was led by the author throughout: the research question, hypotheses, experimental design, every design decision and amendment, the interpretation of results, and running all experiments. The code was written with the help of an AI coding assistant (Claude) and was reviewed and tested by the author, including 76 automated tests covering grading, backdoor validation, the code transforms, parsing, caching and the paid-endpoint block. All reported numbers were regenerated from the committed result files and checked against them.