

Peer Review

Review of: "Enhancing Code LLMs with Reinforcement Learning in Code Generation: A Survey"

Evgenii Arsentev¹

1. Independent researcher

1. What the work claims

This is a survey. The authors set out to map how reinforcement learning is being used to improve code generation and code optimization with large language models, and to hand researchers a single entry point into that literature. The paper walks through background on code generation and RL (§2), the training pipeline for Code LLMs from dataset construction to post-training (§3), two application areas (§4), evaluation metrics and benchmarks (§5), and four open challenges (§6). The technical spine is PPO, DPO, and GRPO as alignment methods, RLHF and RRHF as feedback regimes, and unit-test execution as the reward signal that makes RL a natural fit for code. Three systems carry most of the argument: CodeRL[10], PPOCoder[36], and PanGu-Coder2[37]. One results table (Table 1) compares base models, RL-based models, and what the authors call "Multi Reinforcement Learning-based Models" on the APPS benchmark at Pass@1, Pass@5, and Pass@1000. Five figures illustrate the memory-management controller (Figure 1), a 2021-2024 timeline (Figure 2), the RL feedback loop (Figure 3), the Code LLM training flow (Figure 4), and the dataset build process (Figure 5). The reference list runs to 51 entries.

Let me say first what works. The writing is clear, and the paper is easy to follow from start to finish, which is not always true of surveys assembled from many hands. The reference list is complete and hygienic: I checked every citation marker in the body against the list, and all 51 numbers from [1] through [51] are both used in the text and present in the bibliography, with no gaps and no orphans. That is better discipline than I usually see. The comparison in §3.3 between PPO and DPO is genuinely useful because it frames the choice as online exploration against offline efficiency rather than as a ranking, and it names the engineering cost honestly. The explanation in §2.3 of why BLEU and ROUGE fail on code, and why

unit-test outcomes are the right reward, is compact and correct. Table 1 is also more internally consistent than it first appears, which I show below. My problems with the paper are structural, and they start with the abstract.

2. Whether the conclusion holds on the data

2.1 The abstract describes a different paper. The abstract states that the survey highlights the role of RL "in enhancing compiler optimization, resource allocation, and the development of frameworks and tools", and then promises a specific running order: the sections "first" cover compiler optimization, and the discussion "then progresses to the function of RL in resource allocation, emphasizing register allocation and system optimization". None of that exists in the body. There is no section on compiler optimization. There is no section on resource allocation. The word "register" occurs exactly twice in the whole text, both times in §1, in the single sentence describing Figure 1. Section 3, the one titled "Frameworks in Code Generation and Optimization", turns out to be about dataset construction, pre-training, and post-training, not frameworks. A reader who comes to this paper for RL-driven compiler passes, which is a real and active literature, will find one sentence in §1 citing autotuning[3] and a deep-learning phase-ordering work[4], plus a reference to an MLIR RL environment sitting unused in the bibliography. The same gap shows up in Figure 2, whose caption advertises "key projects such as CompilerDream": CompilerDream appears once in this entire document, in that caption. It is never described, never cited, never placed.

2.2 A survey with no stated method is not a systematic survey. The abstract calls this "a systematic survey". I could not find any search protocol in the text. There is no statement of which databases or preprint servers were queried, no query strings, no date window, no inclusion or exclusion criteria, no count of records screened against records retained, no PRISMA-style flow, and no statement of who screened what. For a primary study, that omission would be awkward; for a survey, it removes the only thing a reader can check. With 51 references covering roughly four years and at least five sub-areas, the selection is the result. Without the protocol, I have no way to tell whether the three systems in §4.1 are the three most important RL-for-code systems or simply the three the authors knew best.

2.3 Table 1 has no provenance. The table reports twelve numbers per model across three Pass@k settings, and nowhere does the text say where those numbers came from. No citation is attached to the table, to its caption, or to the sentence in §5.1 that points at it ("For an evaluation of the code generated by Code LLMs enhanced with reinforcement learning, see Table 1"). Were the models re-run by the authors,

or are the figures copied from the original papers? The answer changes how the table should be read, and it is not given. The evaluation configuration is missing too: Eq. (4) makes Pass@k an estimator over n samples per problem, and the text even notes that "a larger n results in a more precise estimate," yet n is never reported for any row. Neither is the sampling temperature, the decoding method, the APPS split used, or any measure of spread. Pass@1000 numbers, in particular, are cheap to quote and expensive to reproduce, and a single standard deviation column would tell a reader how much of the gap between CodeRL at 17.78 and RLTF at 19.92 (Table 1, Pass@1000, column All) is signal.

2.4 The "All" column mostly reconciles, and where it does not, the errors cluster. This is the one thing in the paper I could check numerically, so I did. The table never states how Intro, Inter, and Comp combine into All, but the APPS test set is split 1000 introductory / 3000 interview / 1000 competition, and that 1:3:1 weighting reproduces almost every cell. CodeRL, Pass@1: $(6.20 \times 1000 + 1.50 \times 3000 + 0.30 \times 1000) / 5000 = 11000 / 5000 = 2.20$, which is the published All. Codex 12B, Pass@1: $(4.14 + 0.14 \times 3 + 0.02) / 5 = 4.58 / 5 = 0.916$, published 0.92. RLTF, Pass@1000: $(38.30 + 15.13 \times 3 + 15.90) / 5 = 99.59 / 5 = 19.918$, published 19.92. Of the 39 All cells I could recompute, 34 agree to rounding. Five do not, and four of the five sit in one row. GPT-2 (1.5B), Pass@5: its own entries give $(5.50 + 1.03 \times 3 + 0.00) / 5 = 8.59 / 5 = 1.718$, but the table prints 1.58. GPT-2 (1.5B), Pass@1000: $(27.90 + 9.27 \times 3 + 8.80) / 5 = 64.51 / 5 = 12.902$, but the table prints 12.32. Note that the printed 1.58 is exactly the correct All for the GPT-Neo (2.7B) row below it $((5.50 + 0.80 \times 3 + 0.00) / 5 = 1.58)$, and that the two rows also share Pass@5 Intro = 5.50 and Pass@1000 Intro = 27.90. That pattern reads like a copy slip between adjacent rows rather than a rounding artifact. The remaining two: GPT-3 175B, Pass@5, All is printed 1.02 against 0.978 implied; PPOCoder, Pass@1000, All is printed 17.77 against 17.83 implied, and that cell is duplicated into the last row of the table.

2.5 The third block of Table 1 does not hold together. "Multi Reinforcement Learning-based Models" is defined nowhere in the paper. The heading appears in the table and in the caption, and the text never explains what makes a model "multi". Worse, the row "PPOCoder + CodeT5, 770M" is identical to the row "PPOCoder, 770M" in all twelve numeric cells: 5.20 / 1.00 / 0.50 / 1.74, then 9.10 / 2.50 / 1.20 / 3.56, then 35.20 / 13.35 / 13.90 / 17.77. That is expected, since PPOCoder is CodeT5 trained with PPO, as §4.1 itself says. The row therefore contributes nothing and implies a distinction that does not exist. The other row in the block, "CodeRL + CodeT5", is worse than plain CodeRL on Pass@1 All (1.71 against 2.20) and on Pass@5 All (3.51 against 3.10, where it is better), which the paper does not comment on at all. A reader is left to guess whether these are different checkpoints, different decoding budgets, or a labeling mistake.

2.6 Two models are evaluated but never discussed. RLTF and B-Coder appear in Table 1 with full result rows. Neither is mentioned anywhere in the prose, and neither carries a citation. B-Coder also carries a strange size field, " $\leq 770\text{M}/\text{stage3}$ ", which is not explained. RLTF has the best Pass@1000 All in the table (19.92), so it is not a minor entry; it is the strongest RL system the authors report, and the survey says nothing about how it works.

2.7 Claims about deployed commercial systems carry no citation. Section 4.2 asserts that GitHub Copilot "employs Reinforcement Learning from Human Feedback (RLHF) to improve functions such as code completion, generation, refactoring, and documentation", that Zhipu AI's CodeGeeX is built on ChatGLM with RLHF, and that Huawei's CodeArts Snap uses PanGu-Coder2 and RRHF. The CodeGeeX reference is in the bibliography but is not attached to this sentence, and the Copilot claim has no supporting reference of any kind. These are statements about the internals of closed products. They need a source, or they need to be softened to what the vendors have published.

2.8 Terminology drifts within the paper. RRHF is expanded as "Ranking Reinforcement from Human Feedback" in §4.1 and as "Reinforcement Learning from Human Reverse Feedback" in §4.2. Those are not the same phrase, and neither matches the expansion in the source literature. In §2.1, the decoding strategies are given as "greedy search and bundle search", while §3.3 correctly says "beam search"; "bundle search" is not a term of art. The same sentence in §2.1 calls Top-P sampling "kernel" sampling, where the standard name is nucleus sampling. Each of these individually is small. Together they make the vocabulary unreliable for exactly the reader the paper is written for, someone entering the area who cannot yet tell a typo from a distinction.

2.9 Equation (1) has an indexing error. The sentence introducing it defines the model as predicting $p_{LM}(x_t \mid x_{1:t-1})$, which is right. Equation (1) then writes the program likelihood as the product over t of $p_{LM}(x_t \mid x_{1:t})$, conditioning each token on a prefix that already contains it. The fix is one character, but it is in the first equation of the paper.

2.10 The survey is about RL and contains no RL objective. This is my main technical objection. Equations (1) through (4) are, in order: the autoregressive likelihood, the few-shot conditioning rewrite, an argmax over parameters for the best program, and the Pass@k estimator. Not one of them is a reinforcement learning equation. There is no expected-return objective, no policy gradient, no PPO clipped surrogate with its ratio and ϵ , no KL penalty written out despite the KL term being discussed three separate times in §2.2 and §3.3, no DPO log-likelihood-ratio loss, and no GRPO group-relative advantage. §2.2 tells the reader that GRPO "uses the average reward from different solutions to the same

problem as a baseline", which is the right sentence, and then stops. A survey can reasonably choose prose over formalism, but this paper chose formalism for the parts that are not its subject and prose for the parts that are. A single table with one row per algorithm (PPO, DPO, GRPO, RRHF, Online RFT) giving the objective, the reward source, whether a value model is required, and the reported cost would carry more of the survey's weight than §§3.1-3.3 currently do.

2.11 Coverage is thinner than the framing suggests. Section 4.1, the section where the survey should be at its strongest, describes three systems in three long sentences each. Section 4.2 then leaves research entirely for IDE products. Between them, the paper covers fewer RL-for-code systems in its body than its own Table 1 evaluates. There is also no discussion of repository-level or agentic evaluation anywhere, although §4.2 is titled "The Impact of Reinforcement Learning on End-to-end Software Development" and §6 asks for benchmarks that "replicate the complexities of real-world software development". The benchmark list in §5.2 stops at function-level suites (HumanEval, HumanEval+, MBPP, MBPP+, APPS, APPS+, LiveCodeBench). The gap between what §6 asks for and what §5.2 lists is the most interesting thing in the paper, and it goes unremarked.

3. Reproducibility

For a survey, reproducibility means two things: can I redo the literature selection, and can I trace every reported number to its source. Neither is possible from this text.

Selection. Covered in §2.2 above. No protocol, no queries, no criteria, no counts, no cut-off date. I cannot reproduce the set of 51 references, and I cannot tell what was considered and rejected.

Table 1 provenance. No source citation for any row. No n for the Pass@ k estimator of Eq. (4), no temperature, no Top-p or Top-k setting, no seeds, no APPS split identifier or revision, no unit-test runner version, no hardware, no wall-clock or GPU-hour figures. The paper cannot say whether the differences between rows are larger than run-to-run variance because variance is not reported. I want to be fair here: if these numbers are transcribed from the original papers, the missing configuration is inherited rather than introduced, and a citation per row would resolve most of it at very low cost.

Model and version identifiers. Model names throughout are bare family names: Codex, StarCoder, CodeT5, CodeLlama, CodeGemma, Qwen2.5-Coder, OpenCoder, DeepSeekMath, ChatGLM. No checkpoint tags, no dates, no parameter counts outside Table 1, no links. For the closed models (Codex, the GPT series, Copilot), no snapshot can even exist retrospectively, which is worth stating rather than passing

over. This is close to universal in LLM surveys, and I flag it here because the paper is otherwise careful with its citations and the fix is mechanical.

Figures. Figures 1 through 5 are schematics with no underlying data, which is fine, but Figure 2 is presented as a timeline of projects "from 2021 to 2024" and so does encode claims: which projects, from which organizations, in which year. As rendered on Qeios, I cannot read the contents of Figure 2, and the body text does not restate them, so the timeline's claims are unverifiable from the paper. That figure should have a companion table.

Artifacts and declarations. No repository, no supplementary material, no artifact link anywhere in the text. The Affiliations and Declarations blocks on the Qeios page are collapsed in the version I read, so I could not check funding or competing-interest statements and take no position on them.

4. *What specifically to fix*

1. **Abstract, sentences 2 and 3.** Either write the missing sections or rewrite the abstract. As it stands, the abstract promises compiler optimization, resource allocation, and register allocation, and the body delivers dataset construction, pre-training, post-training, two application vignettes, and a metrics chapter. I would rewrite the abstract because the paper the authors actually wrote is a coherent one: RL as a post-training method for Code LLMs, with unit tests as the reward. Say that.
2. **New §1.x or §2.0: the survey protocol.** Half a page. Which sources were searched, with what terms, over what date range, what was included and excluded and why, and how many papers were screened to reach 51. If the selection was opportunistic rather than systematic, drop the word "systematic" from the abstract and say the survey is a curated introduction. Both are legitimate; only one of them is currently claimed.
3. **Table 1 caption and rows.** Attach a citation to every row. State n , the sampling temperature, and the decoding strategy used for the Pass@k estimates of Eq. (4), and state whether the numbers were re-run or transcribed. Give the APPS split.
4. **Table 1, GPT-2 (1.5B) row.** Pass@5 All prints 1.58 where the row's own Intro/Inter/Comp entries give $(5.50 + 1.03 \times 3 + 0.00)/5 = 1.72$; Pass@1000 All prints 12.32 where the entries give $(27.90 + 9.27 \times 3 + 8.80)/5 = 12.90$. The printed 1.58 equals the correct All of the GPT-Neo (2.7B) row directly below, and the two rows share Intro values in both columns. Check the transcription of both rows against the source.

5. Two smaller arithmetic mismatches in the same column, same weighting: GPT-3 175B, Pass@5, All (printed 1.02, implied 0.978) and PPOCoder, Pass@1000, All (printed 17.77, implied 17.83, and duplicated into the final row).
6. **Table 1, third block.** Delete the row "PPOCoder + CodeT5": it is cell-for-cell identical to the "PPOCoder" row, and PPOCoder is CodeT5 plus PPO by the paper's own description in §4.1. Then define "Multi Reinforcement Learning-based Models" in the text, or drop the block. If "CodeRL + CodeT5" is a distinct configuration, explain why it scores below CodeRL on Pass@1 All (1.71 against 2.20).
7. **§4.1.** Add RLTF and B-Coder to the prose with citations. They carry result rows in Table 1; RLTF has the best Pass@1000 All in the table (19.92), and a reader currently has no way to find out what either system does.
8. **§4.2, the Copilot / CodeGeeX / CodeArts Snap paragraph.** Cite each claim about a commercial product, or restate it as what the vendor has published. The assertion that Copilot uses RLHF for completion, generation, refactoring, and documentation is currently unsourced.
9. **§4.1 and §4.2, RRHF.** Pick one expansion of the acronym and use it in both places. Also, fix "bundle search" to "beam search" and "kernel sampling" to "nucleus sampling" in §2.1, so that they match §3.3.
10. **Equation (1), §3.1.** The conditioning should be $x_{1:t-1}$, matching the definition given in the sentence above the equation. As printed, each token conditions on itself.
11. **New table in §2.2 or §3.3: the algorithms.** One row each for PPO, DPO, GRPO, RRHF, and Online RFT; columns for the optimization objective written out, the reward source, whether a separate value model is needed, whether it is online or offline, and the reported compute cost. This is the table a survey of RL for code is expected to have and does not.
12. **§3.3, the OpenCoder / DeepSeekMath / CORGI / Qwen2.5-Coder paragraph.** Four systems described in four consecutive sentences with no citation markers on any of them. CORGI in particular is expanded and described but is untraceable from the bibliography.
13. **Figure 2.** Add a companion table listing each project in the timeline with its year, organization, and citation, so the timeline's claims survive outside the image.
14. **§5.2 and §6.** Connect them. Section 6 asks for benchmarks that reflect real software development; §5.2 lists only function-level suites. Say plainly that the benchmark set the field uses does not yet

answer the question §6 poses, and name the repository-level and agentic evaluations that have appeared, or state that they are out of scope and why.

15. §3, title. "Frameworks in Code Generation and Optimization" announces frameworks and delivers training stages. Retitle it something like "Training Pipeline for Code LLMs".

16. Minor, §6, first line: "RL emerges as a transform strategy" should read "transformative".

5. Recommendation

Major revisions.

I want to be clear about what I am and am not saying. This is not a paper with a broken argument, because a survey's argument is its organization, and the organization here is defensible once the abstract stops describing a different one. The prose is readable, the citation hygiene is unusually good (all 51 references cited, none orphaned), the PPO-versus-DPO discussion in §3.3 is worth reading, and Table 1 survived a full recomputation of its aggregate column in 34 of 39 cells, which tells me the numbers were handled with care even though their provenance is missing.

What stops me short of minor revisions is that three defects each independently require new writing rather than editing. The abstract describes sections that do not exist (§2.1 above), so either the abstract or the body must change substantially. A survey that calls itself systematic has no stated selection method (§2.2), so a protocol section has to be written or the claim withdrawn. And a survey of reinforcement learning presents four equations, none of which is a reinforcement learning equation (§2.10), so the formal core of the subject is absent and has to be added. On top of those, Table 1 needs row-level citations, a duplicated row removed, an undefined category defined, and two transcription errors in the GPT-2 (1.5B) row checked against the source.

None of this is beyond reach. The authors clearly know the area, and most of my list is a matter of writing down what they already know. I would be glad to look at a revised version, and I expect that with the protocol section, the algorithm table, and a corrected Table 1, this becomes a useful orientation piece for people entering the field.

Declarations

Potential competing interests: No potential competing interests to declare.