Benchmarking Code Generation of Claude, Grok, and O3 on HumanEval+

Aryan Sawhney (ryd2fx@virginia.edu) Aditya Kakkar (zjq5mr@virginia.edu)

Abstract—**Large language models (LLMs) have demonstrated remarkable ability to generate code from natural language descriptions. In this work, we benchmark three state-of-the-art LLMs — Anthropic's Claude 3.7, xAI's Grok 3, and OpenAI's O3 — on the HumanEval+ Python coding challenge benchmark. HumanEval+ is an extended version of OpenAI's HumanEval that includes substantially more test cases per problem for rigorous evaluation. We evaluate each model in a zero-shot setting (no examples given) with deterministic sampling (temperature 0), and test the generated code in a sandboxed environment for functional correctness. Our results show that OpenAI's O3 model achieves the highest pass@1 success rate (88.4%), outperforming Claude (85.4%) and Grok (82.3%). These findings highlight the current landscape of top-tier LLM code generation capabilities and provide insight into each model's strengths. We conclude with a discussion on the implications of these results and directions for future work in LLM-based code generation.**

I. Introduction

Recent advances in large language models have led to impressive capabilities in generating correct computer code from natural language prompts. This has significant implications for software development, enabling tools like GitHub Copilot and other AI pair programmers. Evaluating the functional correctness of code generated by LLMs is crucial to quantify progress. The HumanEval benchmark introduced by Chen *et al.* [1] was an early standard for this purpose, consisting of 164 handcrafted Python programming problems each with a corresponding unit test. OpenAI's Codex model (an earlier GPT-based code model) solved only about 28.8% of the HumanEval problems in a single try [1], illustrating the difficulty of the task. Subsequent models have greatly improved this performance; for example, the GPT-4 model [4] achieved dramatically higher success rates and demonstrated a deep understanding of coding tasks.

Alongside model advancements, there have been efforts to improve evaluation benchmarks. The Mostly Basic Python Problems (MBPP) benchmark introduced by Austin *et al.* [2] provided 974 crowd-sourced Python tasks of varying difficulty to test code synthesis. While HumanEval focuses on correctness of relatively short solutions, MBPP tests a model's ability to generate correct code for straightforward prompts, both in few-shot and fine-tuned settings. More recently, Liu *et al.* [3] highlighted that existing benchmarks may underestimate error rates due to limited test cases. They proposed *EvalPlus*, which automatically augments benchmarks with additional test cases. In particular, EvalPlus extends HumanEval with around 80$\times$ more tests per problem to create *HumanEval+* [3]. This enhanced benchmark catches many previously undetected mistakes, reducing reported accuracy (pass@k) by roughly 20–30% for models like ChatGPT and GPT-4 [3]. Meanwhile, other complementary benchmarks such as BigCodeBench [6] have been introduced to evaluate coding abilities on more diverse and realistic tasks (including using external libraries and multiple function calls). These efforts underscore the importance of robust evaluation as code generation models continue to advance.

In this paper, we compare three cutting-edge LLMs from different organizations on the HumanEval+ benchmark. Specifically, we evaluate OpenAI's latest code-capable model **O3**, Anthropic's **Claude 3.7**, and xAI's **Grok 3**. Each of these models represents a state-of-the-art system: O3 is an advanced successor in the GPT series (with improvements upon GPT-4), Claude 3.7 is Anthropic's flagship model focused on helpful and harmless reasoning [5], and Grok 3 is a newly released model by xAI aimed at high-level reasoning and coding. By testing these models under identical conditions on HumanEval+, we aim to illuminate their relative strengths in code generation. The contributions of this work are: (1) an evaluation methodology for fair comparison of the three models on a rigorous code benchmark, (2) quantitative results (pass@1 rates) highlighting which model currently leads on code generation, and (3) analysis of the results to provide insight into each model's capabilities and areas for future improvement.

II. Related Work

The original HumanEval benchmark [1] was introduced by OpenAI to evaluate code synthesis. Chen *et al.* released a set of 164 Python programming problems and used it to measure the performance of the Codex model (a GPT-3 variant). Codex solved only 28.8% of the tasks with a single sample (pass@1). They also proposed the pass@k metric to capture success rates under multiple sampling. Austin *et al.* (2021) presented the MBPP dataset of 974 "mostly basic" Python problems designed for entry-level programmers. This benchmark has been widely used to measure scaling of model performance in program synthesis.

More recently, Liu *et al.* (2023) proposed EvalPlus, a framework for more rigorous evaluation of LLM-generated code. EvalPlus automatically generates many additional test cases for existing benchmarks. In particular, they extend HumanEval into HumanEval+, adding roughly 80 times more tests per problem. They found that pass@k scores for many models dropped significantly when using the enhanced tests. Our study uses the HumanEval+ benchmark from EvalPlus to ensure a thorough correctness evaluation.

III. Methodology

Our evaluation uses the HumanEval+ benchmark [3] as a testbed for code generation. HumanEval+ consists of the original 164 programming problems from HumanEval [1], each accompanied by a significantly expanded set of test cases (automatically generated and validated as per the EvalPlus framework [3]). Each problem is described by a function signature and a docstring specification (including example usage) that the model must read and then produce the function implementation.

We evaluated three LLMs: OpenAI **O3**, Anthropic **Claude 3.7**, and xAI **Grok 3**. For each model, we used a zero-shot prompt format, providing only the problem description (docstring and function definition) and instructing the model to output the solution code. No few-shot examples or additional context were given, in order to assess the model's ability to synthesize correct code from the problem description alone. We set the sampling temperature to 0 for all models, making the generation deterministic (i.e., the model does not randomly explore different solutions). This ensures that results are reproducible and that each model's output for a given problem is consistent across runs.

Each code solution produced by the model was then executed against the suite of tests for that problem to determine functional correctness. We used an automated evaluation harness in a sandboxed Python environment to run the unit tests safely. If the model's code passes *all* test cases for the problem, it is counted as a success for that problem (pass@1 = 1); if it fails any test or raises an error, it is a failure for that problem (pass@1 = 0). We computed the aggregate **pass@1** metric for each model, which is the percentage of the 164 problems solved correctly on the first attempt. This metric reflects how often a model produces a correct and fully working solution without needing any retries.

All three models were accessed via their respective APIs or platforms using their latest available versions as of early 2025. The O3 model is an experimental OpenAI model that builds on the capabilities of GPT-4; Claude 3.7 is accessed through Anthropic's API, and Grok 3 via xAI's interface. While the internal architectures and training details of these models are proprietary, they are all large-scale transformers with extensive training on code and natural language. This benchmark is provided via a Colab notebook[1] that loads all tasks and test cases. Our use of a single benchmark and uniform settings for prompting and evaluation helps ensure a fair comparison among the models.

IV. Results

Table I summarizes the pass@1 results achieved by each model on HumanEval+. Out of 164 problems, OpenAI's O3 solved the largest fraction, with a pass@1 of 88.4%. Anthropic's Claude was a close second at 85.4%, and xAI's Grok 3 achieved 82.3%. All three models demonstrate strong code generation performance, each correctly solving over 80% of the challenges on this rigorous benchmark. Figure 1 provides a visual comparison of the models' success rates.

TABLE I
Pass@1 performance of each model on the HumanEval+ benchmark (164 Python problems).

Model	**Pass@1 (%)**
O3 (OpenAI)	88.4
Claude 3.7 (Anthropic)	85.4
Grok 3 (xAI)	82.3

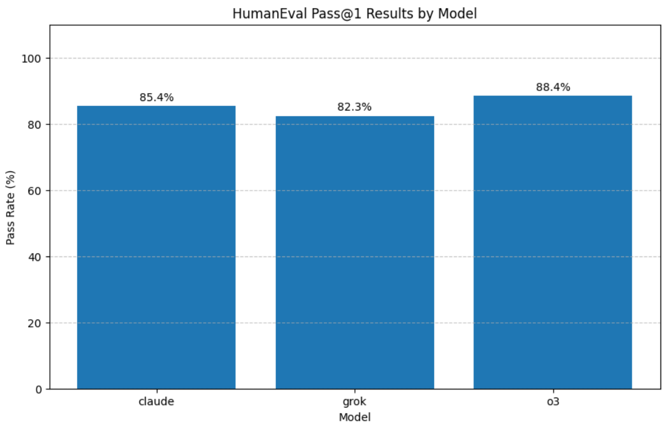

Fig. 1. HumanEval+ pass@1 success rates for each model. The bar chart highlights that O3 achieves the highest success rate, followed by Claude and then Grok. Error bars are not shown as each model was evaluated deterministically (single attempt per problem).

From the results, we see that O3 slightly outperforms the other two models. The absolute gap between O3 and Claude is about 3 percentage points, and O3 leads Grok by roughly 6 percentage points. In practical terms, this means O3 solved 4–5 more problems correctly than Claude, and about 10 more problems than Grok, out of 164. Given the difficulty of HumanEval+ (which includes many edge-case tests), these differences indicate a meaningful but not enormous advantage for O3. All three models perform remarkably well on this benchmark compared to earlier generation models; for context, the original Codex model in 2021 had a pass@1 below 30% on the simpler HumanEval benchmark [1], and even an improved model like GPT-4 was reported to achieve on the order of 80–90% on HumanEval [4]. That all three models here exceed 80% on the much more challenging HumanEval+ suggests that state-of-the-art LLMs have dramatically pushed the frontier of code generation capability.

V. Discussion

The evaluation results highlight the relative strengths of the latest LLMs in code generation. OpenAI's O3 model currently has a slight edge in pass@1 accuracy, indicating it may have the most advanced training or optimization for coding tasks among the models tested. This advantage could stem from various factors: O3 might have a larger or more specialized

[1]https://colab.research.google.com/drive/1McpZg2vcmeHVE0p7QSxd7IxU0efzDRrj?usp=sharing

training dataset (potentially including extensive GitHub code and problem solutions), or it could employ a more effective alignment or reasoning technique that helps it avoid mistakes on edge cases. The GPT-4 technical report [4] noted that careful engineering and massive scale contributed to GPT-4's coding proficiency; O3, as a successor, likely continues this trend with further refinements.

Claude 3.7's performance at 85.4% pass@1 demonstrates that Anthropic's model is nearly on par with O3 for functional code generation. Claude has been known for its strong reasoning abilities and a focus on harmlessness and reliability [5]. Its solid showing here suggests those qualities translate into generating mostly correct code for a wide range of problems. The few problems where Claude fell short of O3 might involve particularly tricky logic or math, where a slight lapse could cause a test failure. It is also possible that Claude's alignment constraints (aimed at preventing risky outputs) could occasionally interfere with optimal code generation (for instance, if a problem involves certain edge-case behaviors). Nonetheless, the difference between 88.4% and 85.4% is modest, implying that for most standard coding tasks, Claude and O3 are comparably reliable.

Grok 3, the newest entrant from xAI, achieved an 82.3% success rate, trailing the leader by around 6 points. This result is still impressive, considering xAI's model is a fresh effort in a landscape where OpenAI and Anthropic have had more time to refine their systems. Grok's performance indicates a high level of competence in understanding problem descriptions and generating correct code. The slightly lower pass rate might point to areas where Grok can be improved. For example, Grok may have struggled with some of the more complex test cases or tricky algorithmic problems. It is also plausible that Grok's training data or fine-tuning regimen for code was not as extensive, given the model's relatively recent development. The gap suggests that xAI's model, while strong, may benefit from further iteration and more exposure to diverse coding tasks. Encouragingly, the fact that Grok is within a few points of models from more established labs hints at a quickly closing competition in the LLM field.

It is worth noting that the absolute performance of all three models on HumanEval+ is very high. An 80–88% pass@1 on this benchmark means that the majority of the time, these models produce correct solutions immediately. This level of accuracy approaches human-level competence on short programming challenges. However, certain limitations remain. First, pass@1 does not capture how models might perform with multiple attempts (pass@k); in practice, users could re-prompt or refine outputs if the first attempt fails. Second, the benchmark tasks are self-contained functions with clear specifications. Real-world programming often involves handling more complex, multi-module projects, integration issues, or ambiguous requirements, which are not captured by HumanEval+. Third, although HumanEval+ has more thorough tests than the original, there is always the possibility of models exploiting patterns or overfitting to known problems. We took steps to mitigate direct data leakage by using a benchmark that was freshly augmented [3], but given that these models have been trained on vast internet data, we cannot entirely rule out that they have seen similar problems before. Nonetheless, the improvements on HumanEval+ relative to prior benchmarks give strong evidence of genuine advancement in coding ability.

Our results also align with other evaluations reported in the community. For instance, the BigCodeBench project [6] evaluates models on tasks involving external libraries and more complex scenarios, and found that even the best models (like GPT-4/O3) solve only around 50–60% of such tasks. In comparison, the near-90% scores on HumanEval+ show that our tested models excel at contained algorithmic problems, but there remains a gap when moving to broader software engineering tasks. This suggests that while LLMs have mastered many of the basics of coding, challenges such as using new APIs, managing long-term program state, or orchestrating multiple components still leave room for improvement.

VI. Future Work

Future research could extend this analysis in several ways:

- **Harder Benchmarks:** Evaluate models on more challenging code tasks, such as LeetCode problems or the new BigCodeBench [6] benchmark, which includes complex, multi-tool programming tasks beyond simple functions.
- **Iterative Debugging:** Explore prompting strategies that allow models to self-correct or iteratively debug their outputs (e.g., asking the model to fix failing tests), which may improve overall solve rates.
- **Multi-language and Multi-paradigm:** Test models on code generation in other programming languages or paradigms, as well as non-algorithmic tasks (e.g., API usage or configuration scripts).
- **Few-shot and Fine-tuning:** Investigate the effect of few-shot prompting or model fine-tuning on these benchmarks, to see if performance gains can be achieved beyond zero-shot.
- **Expanded Evaluation Metrics:** Measure other aspects such as pass@k for larger k, execution efficiency, or adherence to style guidelines, to get a more nuanced picture of each model's coding proficiency.

VII. Conclusion

We presented a comparative study of three leading LLMs — OpenAI O3, Anthropic Claude 3.7, and xAI Grok 3 — on the HumanEval+ code generation benchmark. Using a uniform zero-shot, deterministic evaluation, we found that O3 achieved the highest pass@1 rate (88.4%), with Claude and Grok not far behind. These results reflect the rapid progress in AI code generation: all three models can produce correct solutions for the majority of non-trivial programming problems in a single attempt.

Our analysis indicates that OpenAI's model currently holds a small advantage in coding capability, though the competition is close. The differences observed may point to the impact of training data quality, model size, or fine-tuning strategies. As models like Claude and Grok continue to evolve, we may see the gap further narrow.

This work highlights the value of thorough benchmarks like HumanEval+ in differentiating top-tier models. For future work, we suggest expanding the evaluation to other benchmarks such as MBPP+ (the EvalPlus-augmented MBPP) and BigCodeBench, to cover an even wider spectrum of coding tasks and domains. Evaluating pass@k performance and multi-turn interaction (where the model can be prompted to fix errors) would also give a more complete picture of each model's practical coding utility. Moreover, assessing these models on different programming languages and on real-world coding tasks (for example, debugging existing code or writing larger programs) would help determine how well these high pass@1 rates translate beyond scripted challenges.

In conclusion, the state-of-the-art LLMs from OpenAI, Anthropic, and xAI all show extraordinary coding abilities on benchmarks, with O3 slightly in the lead on HumanEval+. Continued benchmarking and analysis will be critical as new models emerge, to understand their capabilities and guide the development of even more reliable AI for code generation.

REFERENCES

[1] M. Chen *et al.*, "Evaluating Large Language Models Trained on Code," *arXiv preprint* arXiv:2107.03374, 2021.

[2] J. Austin *et al.*, "Program Synthesis with Large Language Models," *arXiv preprint* arXiv:2108.07732, 2021.

[3] J. Liu, C. S. Xia, Y. Wang, and L. Zhang, "Is Your Code Generated by ChatGPT Really Correct? Rigorous Evaluation of LLMs for Code Generation," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

[4] OpenAI, "GPT-4 Technical Report," *arXiv preprint* arXiv:2303.08774, 2023.

[5] Anthropic, "Claude Documentation," 2024. [Online]. Available: `https://docs.anthropic.com`

[6] BigCode, "BigCodeBench: Benchmarking Code Generation with Diverse Function Calls and Complex Instructions," *arXiv preprint* arXiv:2406.15877, 2023.