Table 5: Overview of evaluated models. "Mon. DLs" in the Popularity column refers to the number of model downloads on HuggingFace for the last month (Apr. 2023).

	Model Name	Sizes	Release Date	Open-source	Popularity
Coding	CodeGen [34]	$\{2B,6B,16B\}$	2022	✓	3.5K stars
	CodeGen2 [33]	$\{1B,3B,7B,16B\}$	2023	✓	
	StarCoder [12]	$\{15B\}$	2023	✓	3.8K stars
	SantaCoder [2]	$\{1.1B\}$	2023	✓	1.4M Mon. DLs
	INCODER [16]	$\{1.3B,6.7B\}$	2022	✓	4K Mon. DLs
	PolyCoder [55]	$\{2.7B\}$	2022	✓	2K stars
General	GPT-4 [37]	N/A	Mar. 2023		
	ChatGPT [36]	N/A	Nov. 2022		100M+ users
	VICUNA [11]	$\{7B,13B\}$	Mar. 2023	✓	20K stars
	StableLM [46]	$\{7B\}$	Apr. 2023	✓	14K stars
	GPT-J [49]	$\{6B\}$	2021	✓	5.9K stars
	GPT-NEO [4]	$\{2.7B\}$	2021	✓	7.8K stars

A Detailed Experimental Setup

Evaluation of LLMs. Our goal is to comprehensively evaluate recent and widely used LLMs, both specialized for code generation [34, 55, 16, 2] and general-purpose tasks [37, 36, 11, 46, 49, 4]. Table 5 presents an overview of the studied models, with column **Sizes** reflecting the model sizes in billions of parameters, **Release Date** showing when the LLM is released, **Open-Source** marking the models whose weights are publicly available, and **Popularity** indicating the number of GitHub stars, users, or downloads of the LLM. In total, we evaluate 19 of the most representative and popular LLMs with a broad range of configurations to fully demonstrate the generalizability of our results.

Our hyper-parameter configurations follow prior work [10, 34]. For each model we randomly sample 200 programs and repeat the experiments over temperature ($\{0.2,0.4,0.6,0.8\}$) and greedy decoding with zero temperature. By default, we let each model generate at most 512 new tokens and truncate the produced code with end-of-string (EOS) identifiers suggested in HUMANEVAL [10], as well as those favoured by certain models (*e.g.,* "`<|endoftext|>`" and "`` \n``` ``"). For conversational models (*i.e.,* ChatGPT and GPT-4), we obtain the code fragments by parsing the code blocks (*i.e.,* within "` ``` `") in the output. We found ChatGPT tends to repeat problem description with detailed explanation, which can consume more than 512 new tokens to complete a solution for around 11% of problems. To align ChatGPT with other models, for tasks with very long problem descriptions, we extend the token limit from 512 to 1024. Furthermore, due to the time and cost of querying ChatGPT API, we only evaluate it using greedy decoding and 0.8 temperature. Additionally, GPT-4 is even more costly than ChatGPT ($\sim 10\times$), as such, we only evaluate it over greedy decoding. For model implementation, we run ChatGPT and GPT-4 via OpenAI APIs, and accelerate CodeGen-6B and -16B with NVIDIA Faster-Transformer via FauxPilot [13]. All other LLMs are based on the HuggingFace `transformers` library. Due to the discontinuation of CODEX APIs [10], unfortunately it is not included in the benchmark.

Test oracles. An LLM-produced solution is regarded to be correct if for all test inputs it returns values that match the expected outputs within a reasonable run time. We perform exact matching by default. For floating-point comparisons, we tolerate absolute differences to the degrees annotated in HUMANEVAL or 10^{-6} if not annotated. In original HUMANEVAL, the default timeout is set to three seconds to run the whole test-suite (*i.e.,* all test-cases) for each programming problem. Such a setting is neither suitable when having more test-cases nor reasonable as each problem could have its own run time characteristics. Consequently, we let the timeout for each test-case to be $\max(50\text{ms}, 2\times t_{gt})$ where t_{gt} refers to the execution time of the corresponding ground-truth solution. In other words, we expect the LLM-provided solution to be no slower than the ground-truth by two times or use a base 50-millisecond timeout when $2\times t_{gt} < 50\text{ms}$ to avoid variance caused by performance randomness.

B Checklist Information

In this section we detail additional information required by the submission checklist.

Limitations. Our automatic input generation does have a few limitations: (i) it assumes that ChatGPT or similar LLMs can produce high-quality seed inputs for a given programming task, while such a hypothesis might not hold for all problems; (ii) our proposed type-aware mutation strategies (Table 1) focuses on the commonly used general data types; therefore to support other user defined data types, it requires designing further mutation strategies; and (iii) while we propose to use program contracts to clarify the programming tasks, it may require human annotation.

Experiments and reproducibility. The detailed steps to reproduce our results are detailed in the `README` file of our supplementary material. We recommend the artifact reviewers to reproduce our results on a Linux operating system with adequate RAM (*e.g.,* 64 gigabytes) and system storage (*e.g.,* 500 gigabytes), as well as CPU cores (*e.g.,* 32 cores). The expected run time for reproducing results in the whole paper should be within 10 machine hours under our recommended setting, if directly using our pre-generated LLM code samples. To re-generate the LLM code sample from scratch (*i.e.,* 84 rounds of code generation), it could take several days depending on the available GPU resources.

Error bars. The statistical errors in experiments could happen for Table 3 due to the randomness of random sampling in code generation and flakiness of test-case failures due to timeout settings and test-bed system loads. For the randomness of sampling, though we did not run multiple generation due to the cost, we use a fairly large sample size (*i.e.,* 200) following the Codex paper [10] and an unbiased estimator which is of low variance. We also double checked our replicated results with that claimed in their original papers and did not observe any significant differences. For understand the statistical significance of code evaluation, we run the function correctness evaluation (*i.e.,* testing LLM-produced code) for three times and all observed pass@k differences are within 2 percentage points.

Compute. We by default run our evaluated LLMs (Table 5) on a test-bed with four NVIDIA A6000 GPUs with 50GB VRAM each, except that we run CodeGen2-16B on another test-bed with an NVIDIA H100 GPU with 80GB VRAM. Additionally, we run the commercial models (*i.e.,* ChatGPT and GPT-4) through OpenAI's APIs. We evaluate LLM-produced code on a test-bed with 32 CPU cores (*i.e.,* 64 threads), 256GB RAM and 2TB storage.

Safeguards. Executing arbitrary code generated by LLMs could pose security risks [10]. To alleviate the risk, we provide a `Dockerfile` to run code evaluation in a sandbox (*i.e.,* Docker) such that the host system will not be affected in theory. The detailed instruction to run code evaluation under our provided sandbox is detailed in the `README` file of our supplementary material.

Licenses. Our technique is essentially a dataset augmentation technique and its implementation is licensed under Apache License, Version 2.0 [15]. The augmented datasets still inherit its original license. For example, HUMANEVAL^{+} is still licensed under the MIT license [50] which is used by the original HUMANEVAL.