CRUXEval

Code Reasoning, Understanding, and Execution Evaluation

Alex Gu

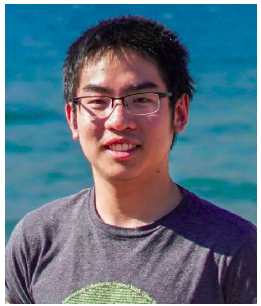
Baptiste Rozière

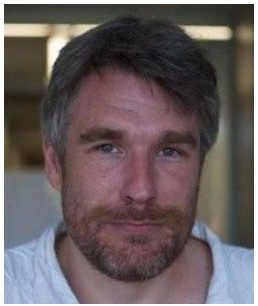
Hugh Leather

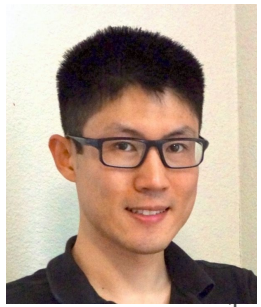
Armando Solar-Lezama

Gabriel Synnaeve

Sida Wang



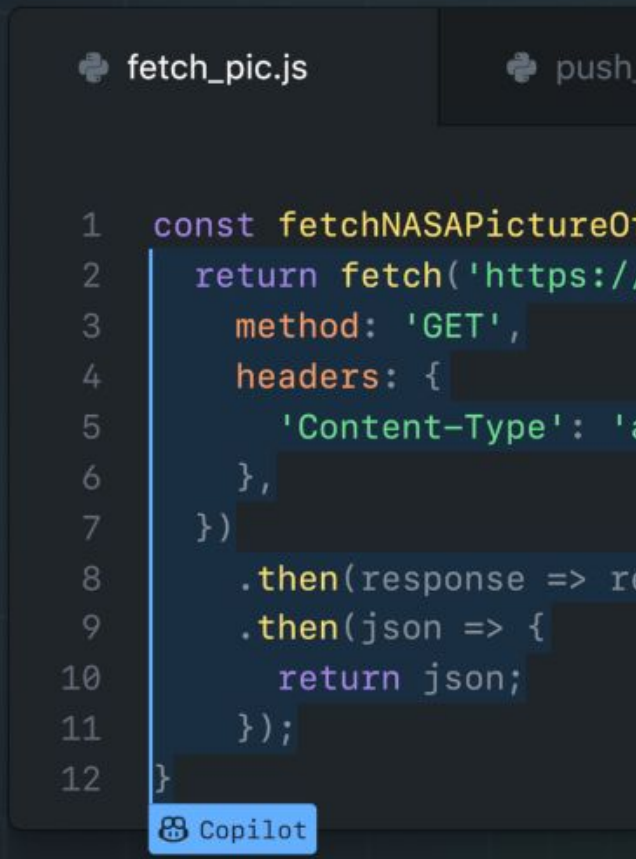


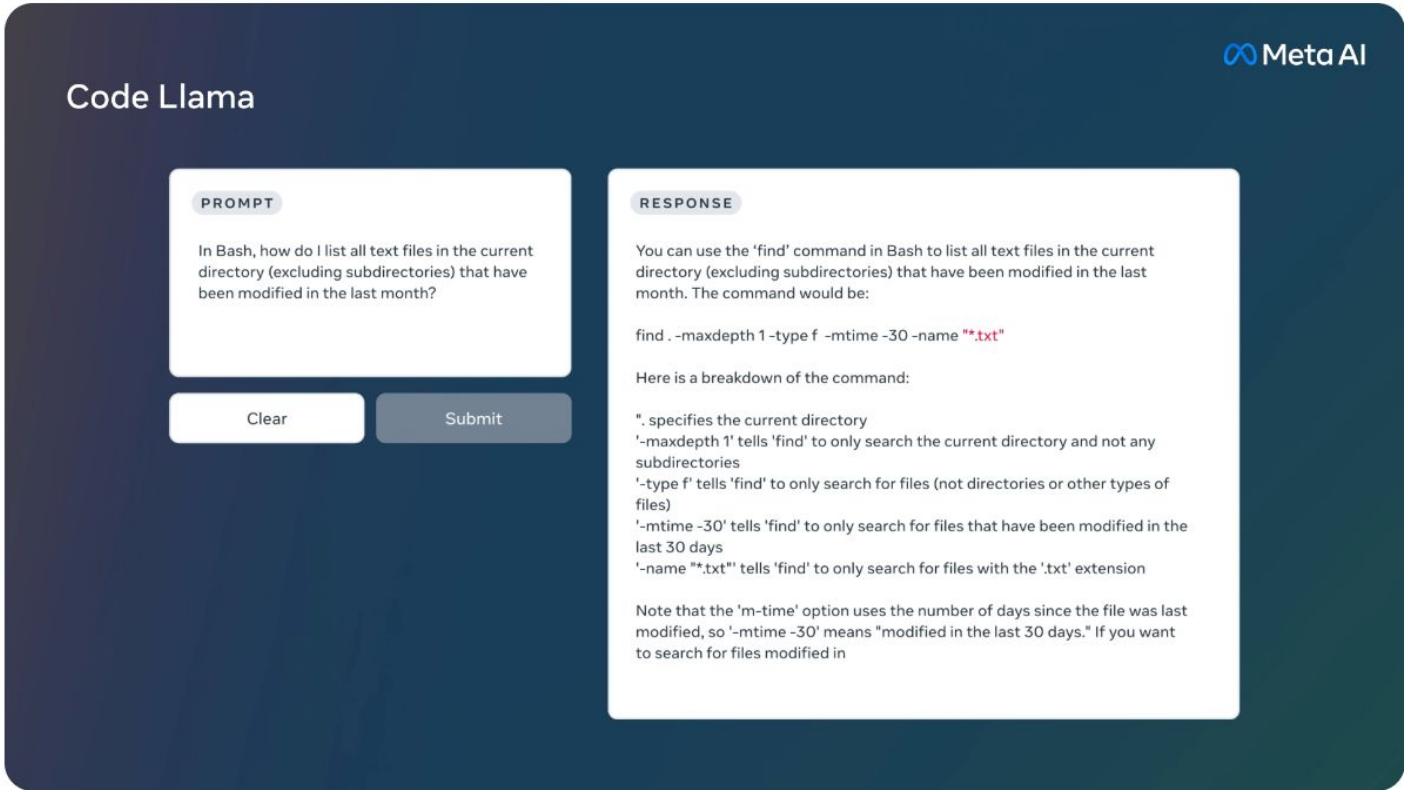



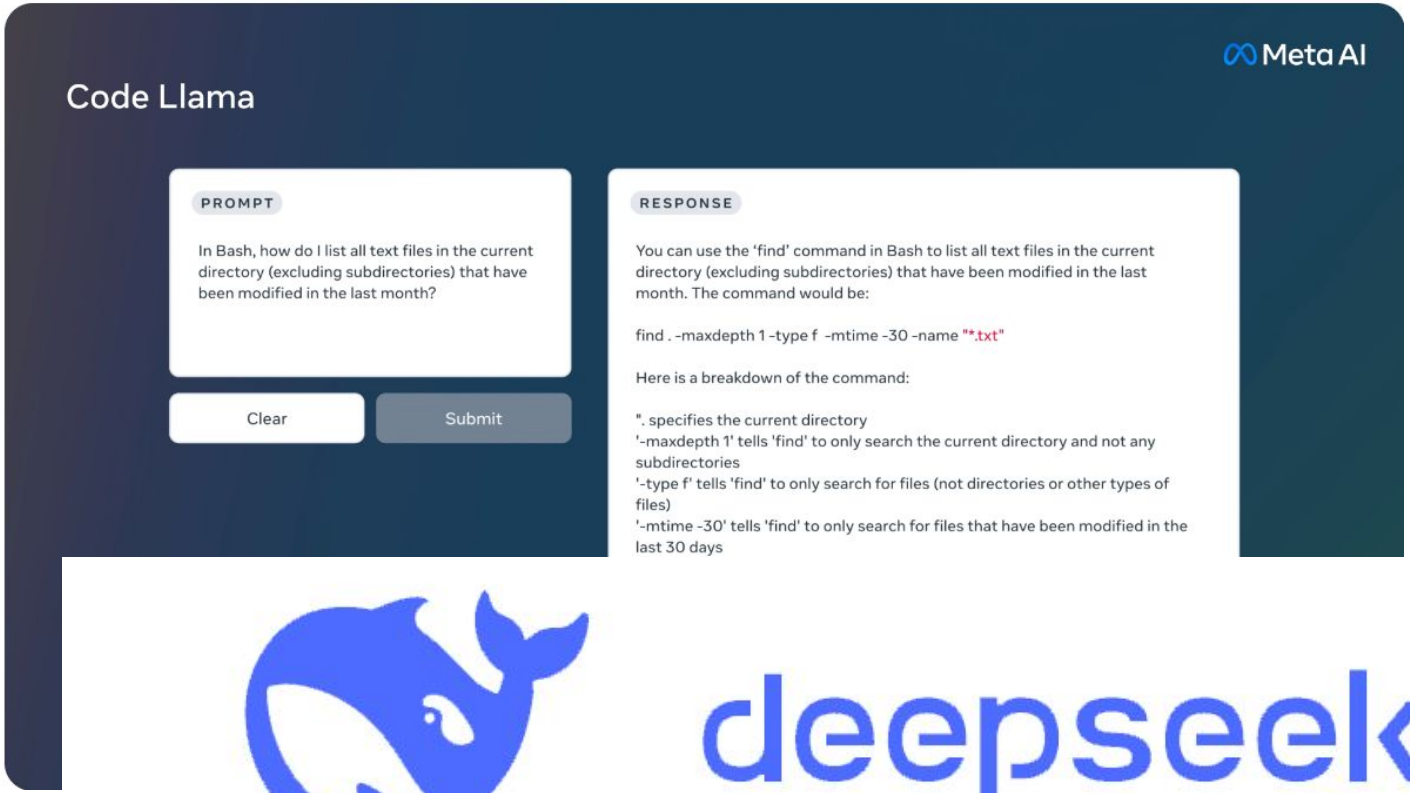
GitHub Copilot

Technical preview

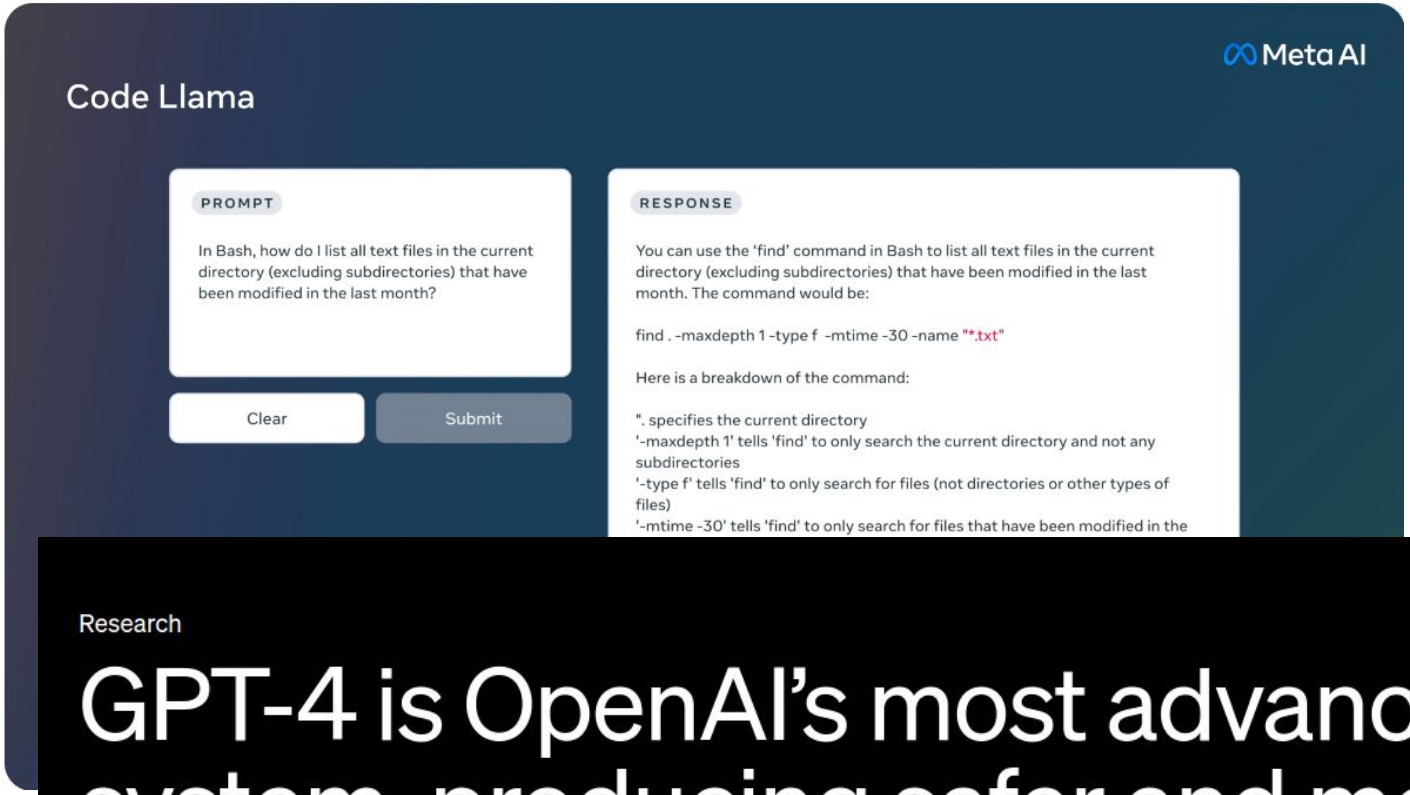
Your AI pair programmer




Meta AI
Code Llama
PROMPT
In Bash, how do I list all text files in the current directory (excluding subdirectories) that have been modified in the last month?
Clear
Submit
RESPONSE
You can use the 'find' command in Bash to list all text files in the current directory (excluding subdirectories) that have been modified in the last month. The command would be:
find . -maxdepth 1 -type f -mtime -30 -name "*.txt"
Here is a breakdown of the command:
". specifies the current directory
'-maxdepth 1' tells 'find' to only search the current directory and not any subdirectories
'-type f' tells 'find' to only search for files (not directories or other types of files)
'-mtime -30' tells 'find' to only search for files that have been modified in the last 30 days
'-name "*.txt"' tells 'find' to only search for files with the '.txt' extension
Note that the 'm-time' option uses the number of days since the file was last modified, so '-mtime -30' means "modified in the last 30 days." If you want to search for files modified in



deepseek coder



Research

GPT-4 is OpenAI's most advanced system, producing safer and more useful responses

HumanEval

- 164 Programming Problems
- Metric: pass@k

```
from typing import List
def has_close_elements(numbers: List[float], threshold: float) ->
bool:
    """Check if in given list of numbers, are any two numbers closer
    to each other than given threshold.

    >>> has_close_elements([1.0, 2.0, 3.0], 0.5)
    False

    >>> has_close_elements([1.0, 2.8, 3.0, 4.0, 5.0, 2.0], 0.3)
    True
    """
```

Sebastien Bubeck
@SebastienBubeck

New LLM in town:

phi-1 achieves 51% on HumanEval w. only 1.3B parameters & 7B tokens training dataset

Any other >50% HumanEval model is >1000x bigger (e.g., WizardCoder from last week is 10x in model size and 100x in dataset size).

How?

Textbooks Are All You Need

Sebastien Bubeck
@SebastienBubeck

New LLM in town:

***phi-1 achieves 51% on
training dataset***

Any other >50% HumanE
from last week is 10x in r

How?

***Textbooks Are All You

Phind
@phindsearch

We beat GPT-4 on HumanEval with fine-tuned CodeLlama-34B!

Here's how we did it: phind.com/blog/code-llam...

🚀 Both models have been open-sourced on Huggingface:

Phind
@phindsearch

WizardLM
@WizardLM_AI

🔥🔥🔥

Introduce the newest **WizardCoder 34B** based on Code Llama.

✅**WizardCoder-34B** surpasses **GPT-4**, **ChatGPT-3.5** and **Claude-2** on HumanEval with **73.2% pass@1**

🖥️Demo: http://47.103.63.15:50085/
🏇Model Weights: huggingface.co/WizardLM/Wizar...
🏇Github: github.com/nlpxucan/Wizar...

Sebasti
@Sebas

New LLM in tc

***phi-1 achie
training datas

Any other >5(
from last wee

How?

***Textbooks Are All You

Phind

WizardLM

anton
@abacaj

Releasing mistral-7b-sft, an initial fine-tune for code completion, explanation and repair. MMLU scores dropped from the base mistral model (as expected) but still much higher than codellama model and comparable to llama-2 7b

HumanEval: pass@1 54.27%
MMLU: 45.89%

***Textbooks Are All You

Do LLMs solve HumanEval like us?

Do LLMs solve HumanEval like us?

Code Reasoning

Code Understanding

Code Execution

Do LLMs solve HumanEval like us?

Code Reasoning	Code Understanding	Code Execution

We can perform these tasks when solving HumanEval, but can models?

CRUXEval-I and CRUXEval-O

Input Prediction (I)

Find an input producing a given output

(code understanding + reasoning)

Output Prediction (O)

Execute the function on a given input

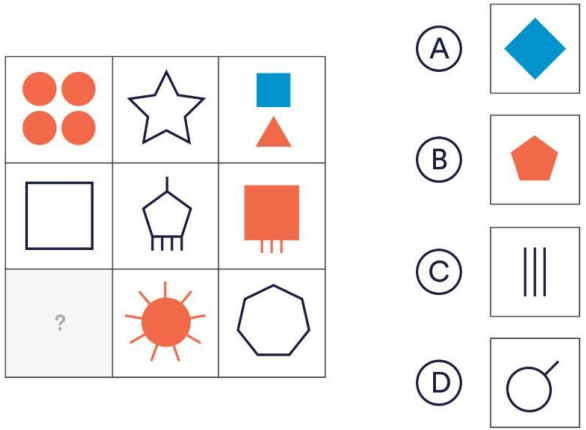
(code execution)

```python
def f(string):
    string_x = string.rstrip("a")
    string = string_x.rstrip("e")
    return string

# output prediction, CRUXEval-O
assert f("xxxxaaee") == ??
## GPT4: "xxxx", incorrect

# input prediction, CRUXEval-I
assert f(??) == "xxxxaa"
## GPT4: "xxxxaae", correct
```

The Importance of CRUXEval

LLMs for Software Engineering

LLMs for Symbolic Reasoning



Part 1: Benchmark Creation

Creating a benchmark to measure code execution

Part 2: Benchmark Evaluation

Can code LMs understand, reason, and execute code like us?

Part 3: CoT and Fine-Tuning

Are standard tricks enough to solve CRUXEval?

Part 1: Benchmark Creation

Creating a benchmark to measure code execution

Part 2: Benchmark Evaluation

Can code LMs understand, reason, and execute code like us?

Part 3: CoT and Fine-Tuning

Are standard tricks enough to solve CRUXEval?

Benchmark Creation

Scalability

Methodology allows for future benchmarks

Simplicity

Ensure benchmark samples are reasonable

Diversity

Tests reasoning for a variety of constructs

Data Model	Evaluation Model	Input Pass@1	Output Pass@1
CL 13B	CL 13B	28.1%	28.4%
CL 13B	CL 34B	33.8%	29.2%
CL 34B	CL 13B	25.1%	24.3%
CL 34B	CL 34B	29.9%	25.4%
CL 34B	GPT-3.5	40.5%	36.6%
GPT-3.5	CL 13B	42.3%	49.7%
GPT-3.5	CL 34B	52.1%	50.7%
GPT-3.5	GPT-3.5	67.1%	67.2%
GPT-4	CL 13B	28.1%	42.4%
GPT-4	CL 34B	37.0%	44.6%

Weak evidence that data-generating model does not seem to have significant impact on performance

Benchmark Creation

Scalability

Methodology allows for future benchmarks

Simplicity

Ensure benchmark samples are reasonable

Diversity

Tests reasoning for a variety of constructs

Benchmark Creation

Scalability

Methodology allows for future benchmarks

Simplicity

Ensure benchmark samples are reasonable

Diversity

Tests reasoning for a variety of constructs

Benchmark Candidate Filtering

1. No imports in function
2. Length of code is between 75 and 300 characters
3. All arithmetic operations must be integer operations
4. No arithmetic operations involving two numerics over 4 in absolute value.
5. Finish running in 2 seconds

Benchmark Creation

Scalability

Methodology allows for future benchmarks

Simplicity

Ensure benchmark samples are reasonable

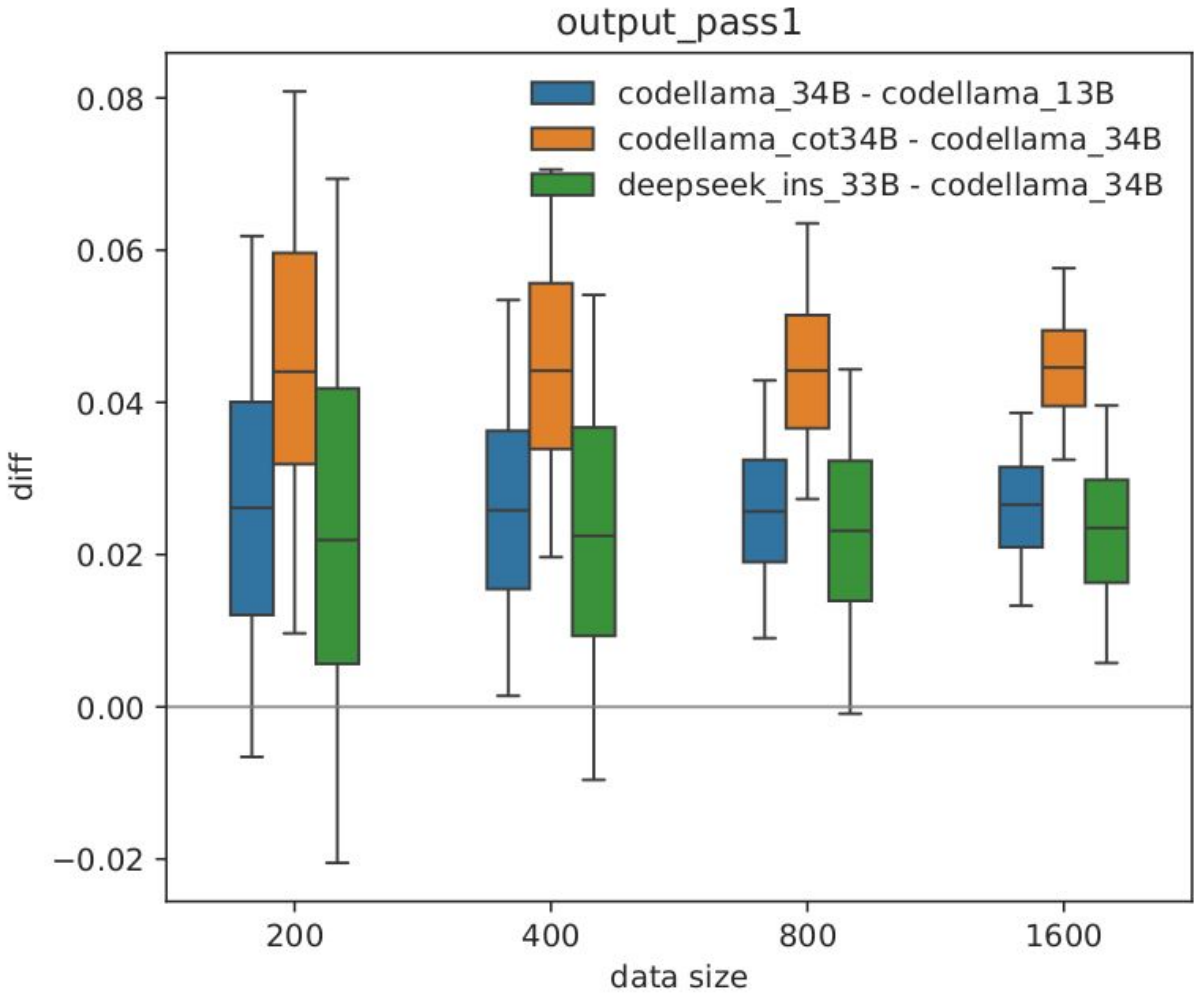
Diversity

Tests reasoning for a variety of constructs

```
You will be given a function name between [TASK] and [/TASK] tags. Following the examples given, write
    ↪ a Python function that makes use of the given function and 5 test inputs for that function.

[TASK]
list.append
[/TASK]
[PYTHON]
def f(nums):
    count = len(nums)
    for i in range(-count+1, 0):
        nums.append(nums[i])
    return nums
[/PYTHON]
[TEST]
assert f([2, 6, 1, 3, 1]) == ??
assert f([7, 1, 2, 6, 0, 2]) == ??
assert f([4, 3, 2, 1, 2, -1, 4, 2]) == ??
assert f([0, 6, 2, -1, -2]) == ??
assert f([-6, -2, 1, -3, 0, 1]) == ??
[/TEST]

[TASK]
str.zfill
[/TASK]
[PYTHON]
```

n_samples = 800

n_generations = 10

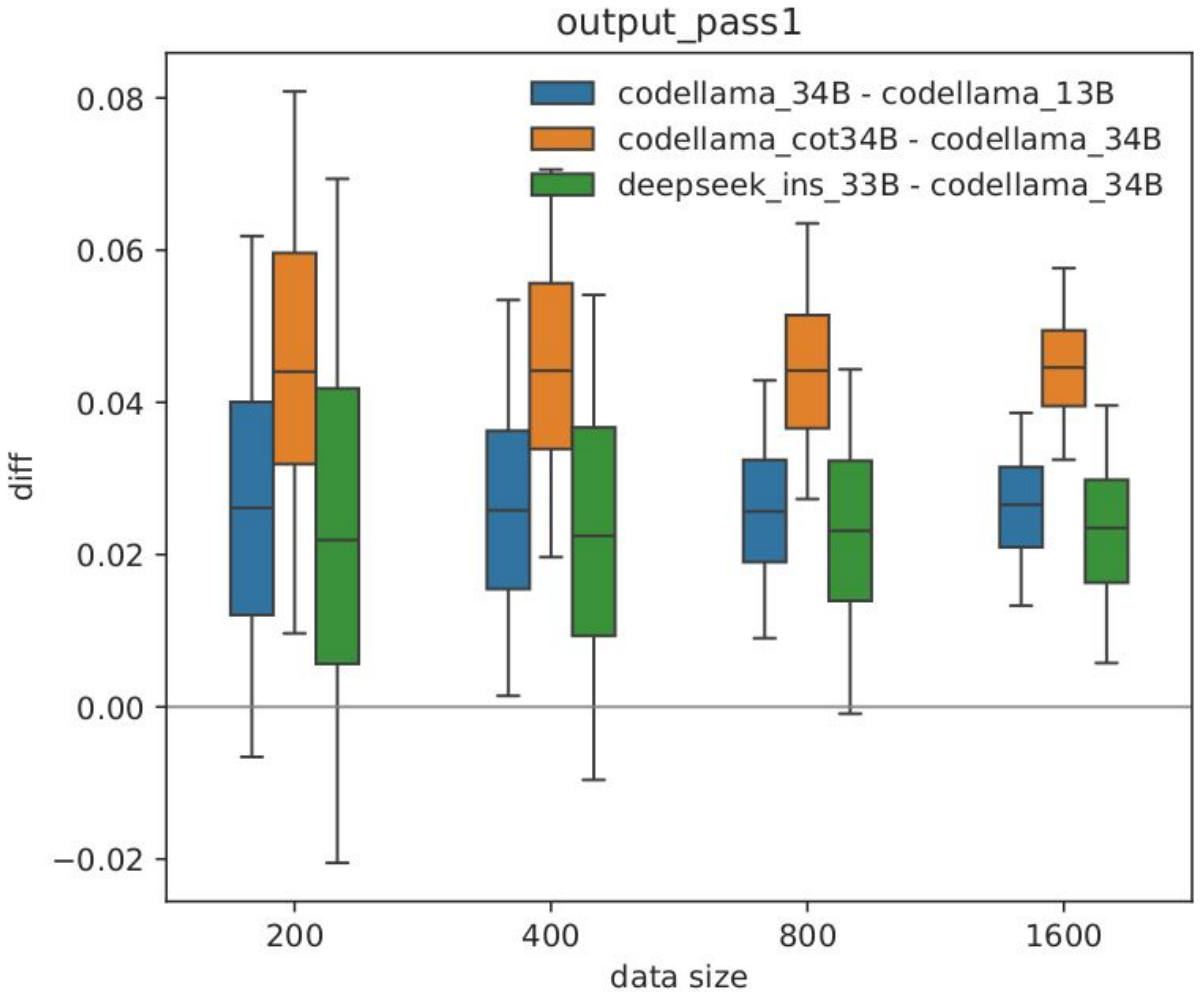


n_samples = 800

n_generations = 10

data noise: 1.5%

model noise: 0.2%



Part 1: Benchmark Creation

Creating a benchmark to measure code execution

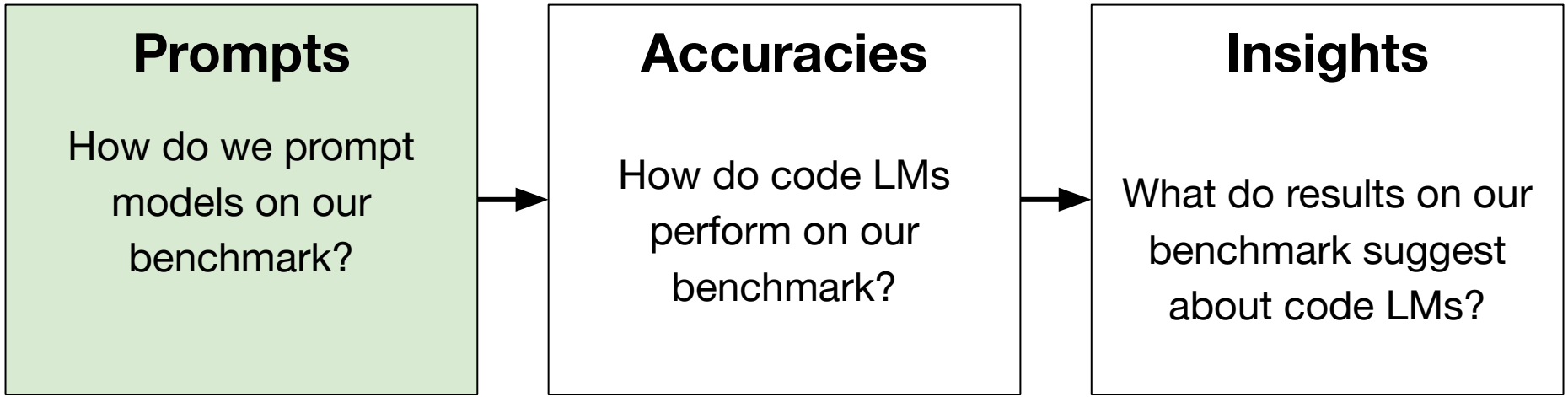
Part 2: Benchmark Evaluation

Can code LMs understand, reason, and execute code like us?

Part 3: CoT and Fine-Tuning

Are standard tricks enough to solve CRUXEval?

Benchmark Evaluation



```
You will be given a function f and an output in the form f(??) == output. Find any input such that
    ↪ executing f on the input leads to the given output. There may be multiple answers, but you
    ↪ should only output one. Think step by step before arriving at an answer. Finally, surround the
    ↪  answer, with no additional words, with [ANSWER] and [/ANSWER] tags. Express your answer as a
    ↪ function call that when executed will give the output.
```

instruction

```
[PYTHON]
def f(my_list):
    count = 0
    for i in my_list:
        if len(i) % 2 == 0:
            count += 1
    return count
assert f(??) == 3
[/PYTHON]
[ANSWER]
f(["mq", "px", "zy"])
[/ANSWER]
```

example

```
[PYTHON]
{function}
assert f(??) == {output}
[/PYTHON]
[ANSWER]
```

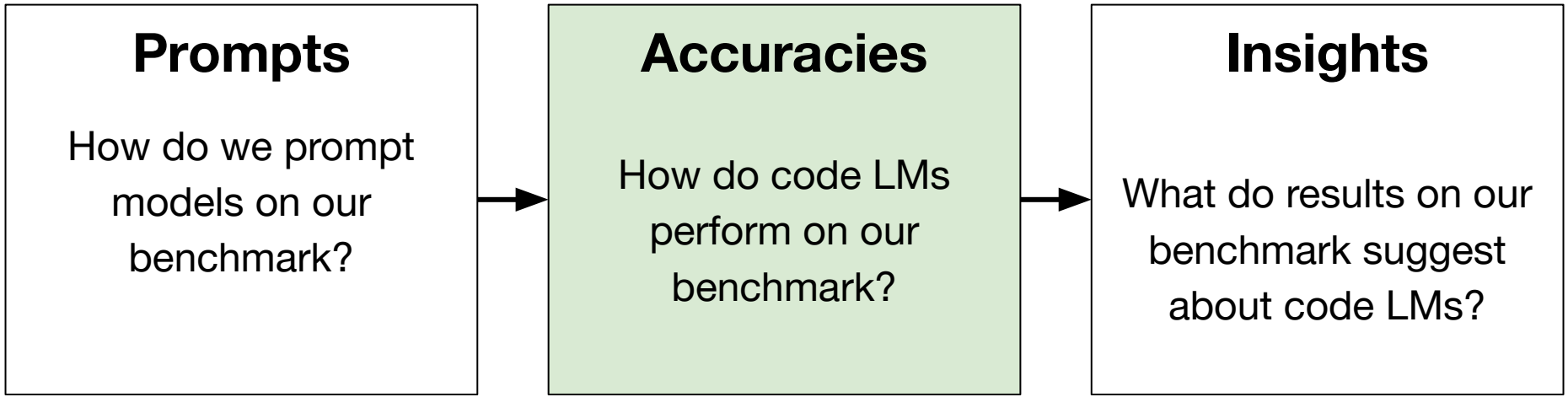
query

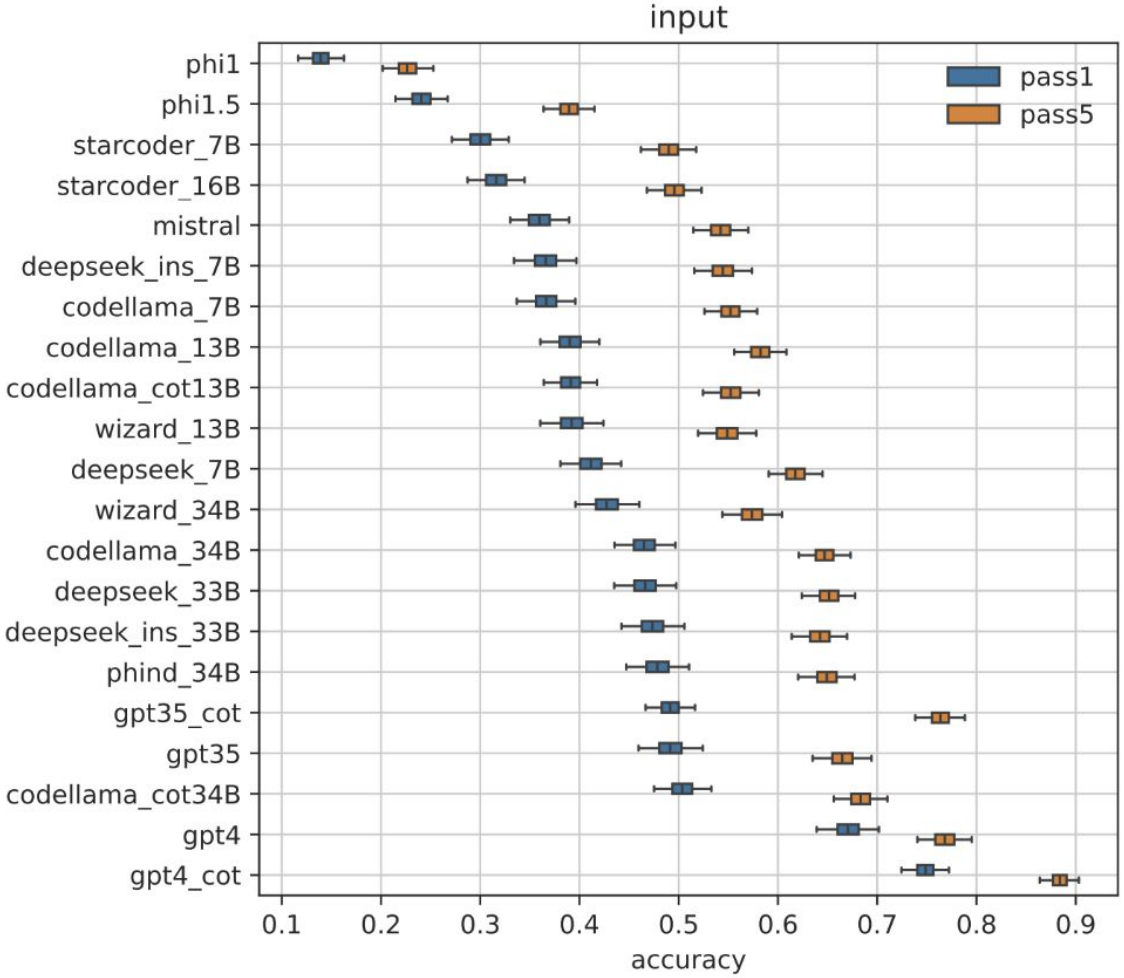
Input Prediction Prompt

```
Based on the given Python code, which may contain errors, complete the assert statement with the
    ↪ output when executing the code on the given test case. Do NOT output any extra information,
    ↪ even if the function is incorrect or incomplete. Do NOT output a description for the assert.

def f(n):
    return n
assert f(17) == 17

{function}
assert f({input}) ==
```

Output Prediction Prompt

Benchmark Evaluation




input
phi1
phi1.5
starcoder_7B
starcoder_16B
mistral
deepseek_ins_7B
codellama_7B
codellama_13B
codellama_cot13B
wizard_13B
deepseek_7B
wizard_34B
codellama_34B
deepseek_33B
deepseek_ins_33B
phind_34B
gpt35_cot
gpt35
codellama_cot34B
gpt4
gpt4_cot
pass1
pass5
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
accuracy

GPT-4 + CoT Failures

```
def f(text, suffix):
    if suffix == '':
        suffix = None
    return text.endswith(suffix)
assert f('uMeGndkGh', 'kG') == ??
# GPT-4 CoT: True
# should be False
```

```
def f(num):
    if 0 < num < 1000 and num != 6174:
        return 'Half Life'
    return 'Not found'
assert f(6173) == ??
# GPT-4 CoT: 'Half Life'
# should be 'Not found'
```

```
def f(text, repl):
    trans = str.maketrans(text.lower(), repl.
    ↪ lower())
    return text.translate(trans)
assert f(??) == 'lwwer case'

# GPT4 CoT: 'lower case', 'ow'
# could be 'lower case', 'lwwer case'
```

```
def f(text):
    string = ''
    for char in text:
        string += char + char.lower()
    return string
assert f(??) == 'llaallaakk'
# GPT-4 CoT: 'LAK'
# should be 'lalak'
```

CRUXEval-I

#	Model	pass@1	pass@5
1	gpt-4-0613+cot	75.5	88.9
2	gpt-4-0613	69.8	76.8
3	gpt-3.5-turbo-0613+cot	50.3	74.9
4	codellama-34b+cot	50.1	73.8
5	codetulu-2-34b	49.3	68.0
6	gpt-3.5-turbo-0613	49.0	63.2
7	codellama-13b+cot	47.4	68.4
8	codellama-34b	47.2	66.6
9	phind	47.2	63.9
10	deepseek-base-33b	46.5	64.9

CRUXEval-O

#	Model	pass@1	pass@5
1	gpt-4-0613+cot	77.1	88.2
2	gpt-4-0613	68.7	73.0
3	gpt-3.5-turbo-0613+cot	59.0	76.7
4	deepseek-instruct-33b	49.9	61.8
5	gpt-3.5-turbo-0613	49.4	59.3
6	deepseek-base-33b	48.6	61.6
7	codetulu-2-34b	45.8	58.9
8	magicoder-ds-6.7b	44.4	57.5
9	codellama-34b+cot	43.6	69.4
10	deepseek-base-6.7b	43.5	54.8

Function Anonymization

Original

```
def f(s):
    nums = ''.join(filter(lambda c:c.isdecimal(), s))
    if nums == '': return 'none'
    m = max([int(num) for num in nums.split(',')])
    return str(m)
assert f('01,001') == '1001'
```

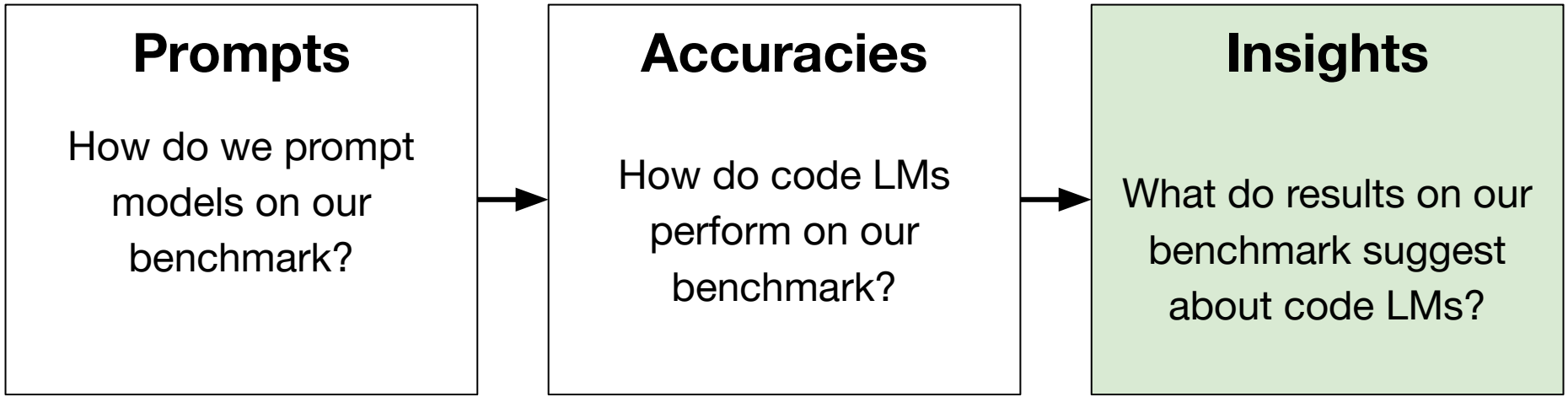
Anonymized

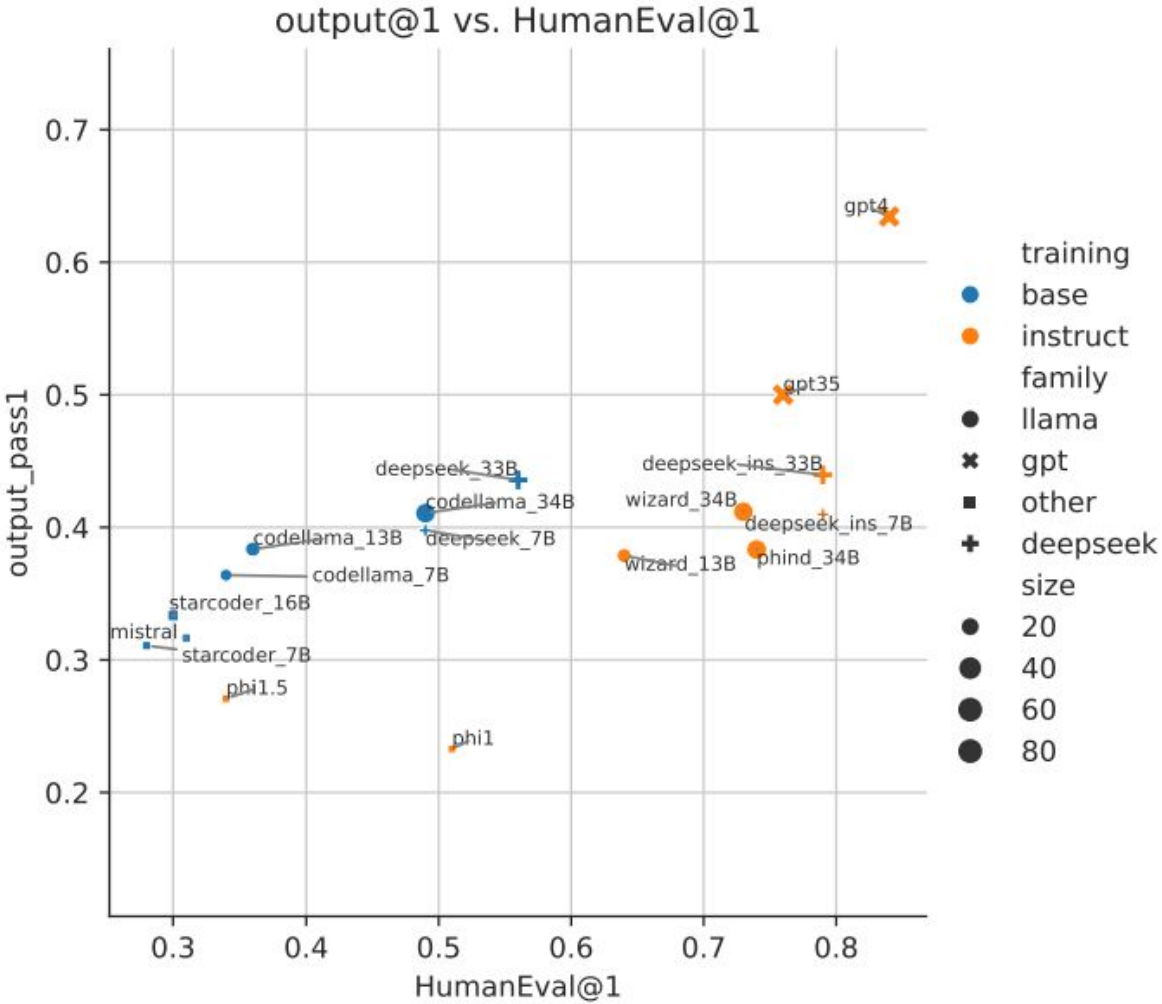
```
def f(x0):
    x1 = ''.join(filter(lambda x2: x2.isdecimal(), x0))
    if x1 == '':
        return 'none'
    x3 = max([int(x4) for x4 in x1.split(',')])
    return str(x3)
assert f('01,001') == '1001'
```

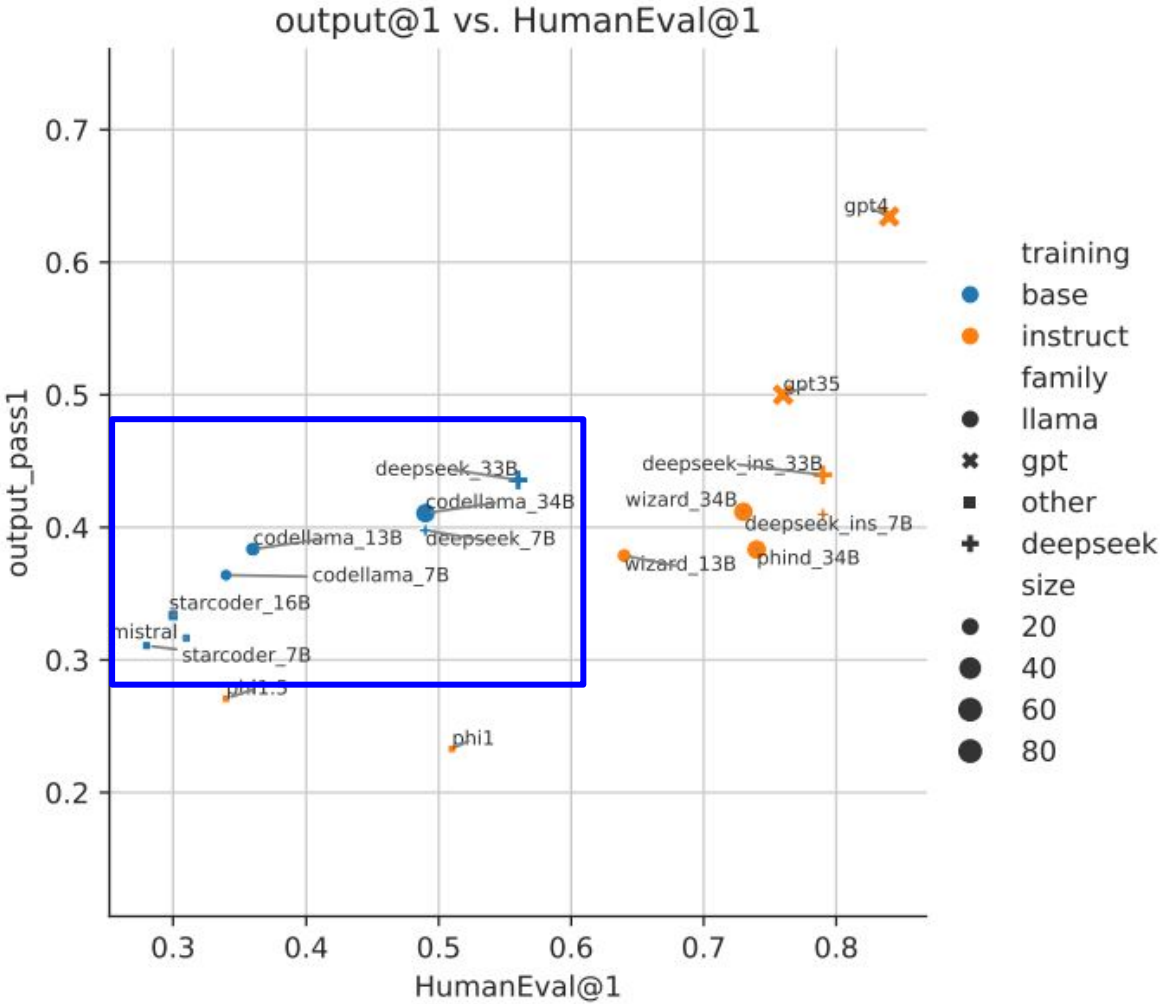
No significant drop from anonymizing variable names!

Model	Anonymized	Input Prediction		Output Prediction	
		Pass@1	Pass@5	Pass@1	Pass@5
CodeLlama 7B	✗	36.6%	48.0%	36.4%	43.5%
	✓	37.5%	53.3%	34.0%	46.9%
	Δ	+0.9%	+5.3%	-2.4%	+3.4%
CodeLlama 13B	✗	39.0%	50.2%	38.3%	44.7%
	✓	40.0%	55.8%	36.1%	50.6%
	Δ	+1.0%	+5.6%	-2.2%	+5.9%
CodeLlama 34B	✗	46.5%	57.4%	41.1%	47.5%
	✓	48.0%	63.8%	39.1%	54.0%
	Δ	+1.5%	+6.4%	-2.0%	+6.5%

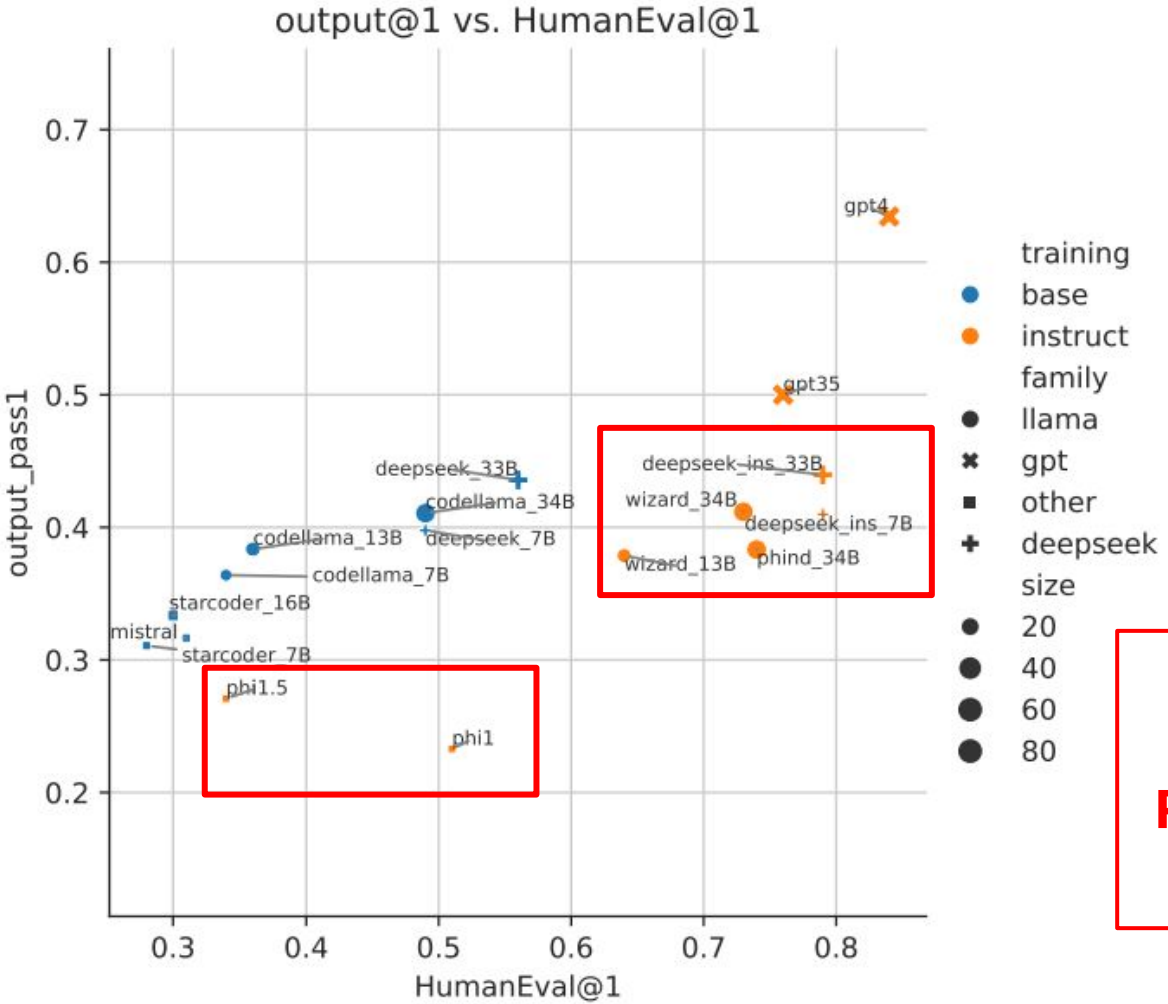
Benchmark Evaluation



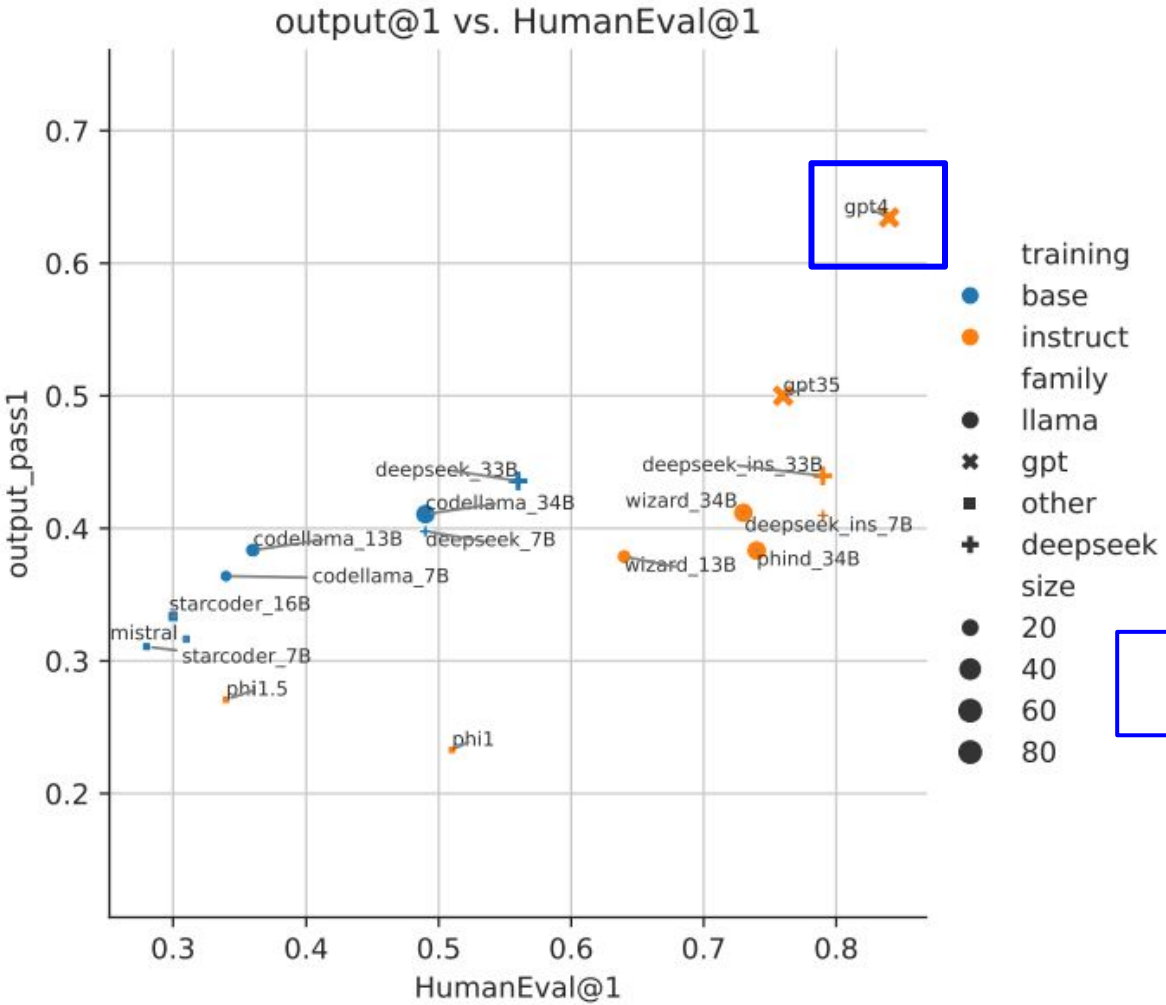

output@1 vs. HumanEval@1
output_pass1
HumanEval@1
0.2
0.3
0.4
0.5
0.6
0.7
0.3
0.4
0.5
0.6
0.7
0.8
gpt4
gpt35
deepseek_33B
codellama_34B
deepseek_ins_33B
wizard_34B
deepseek_ins_7B
codellama_13B
deepseek_7B
wizard_13B
phind_34B
codellama_7B
starcoder_16B
mistral
starcoder_7B
phi1.5
phi1
training
base
instruct
family
llama
gpt
other
deepseek
size
20
40
60
80



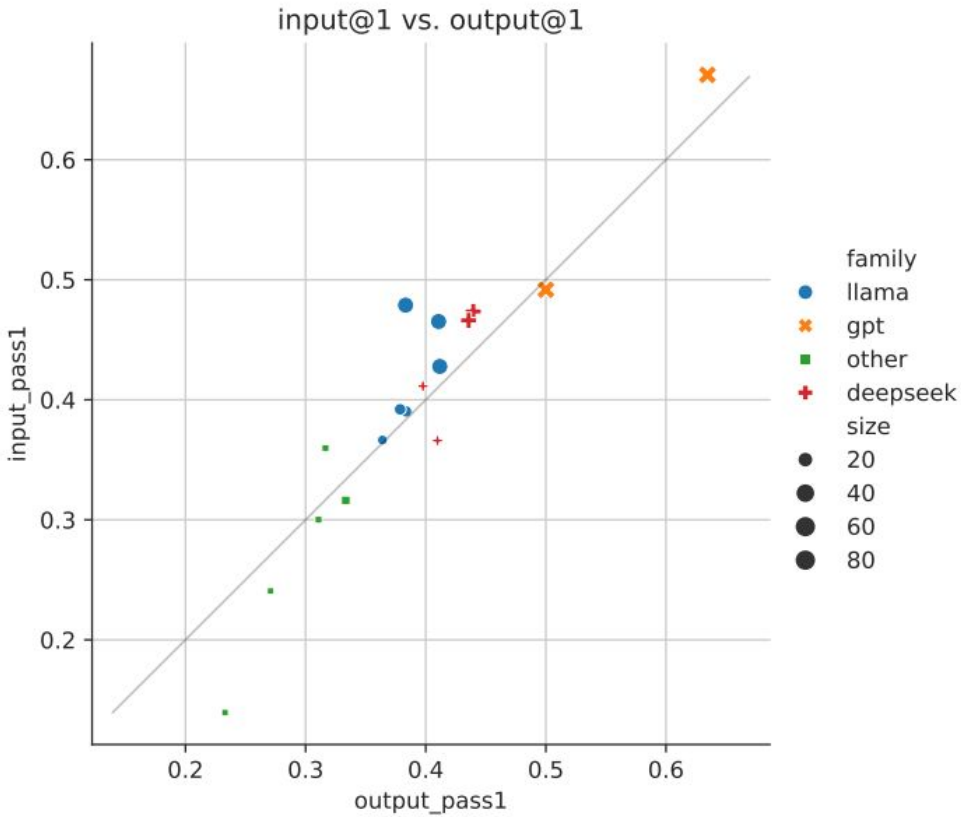
Base models show a correlation.

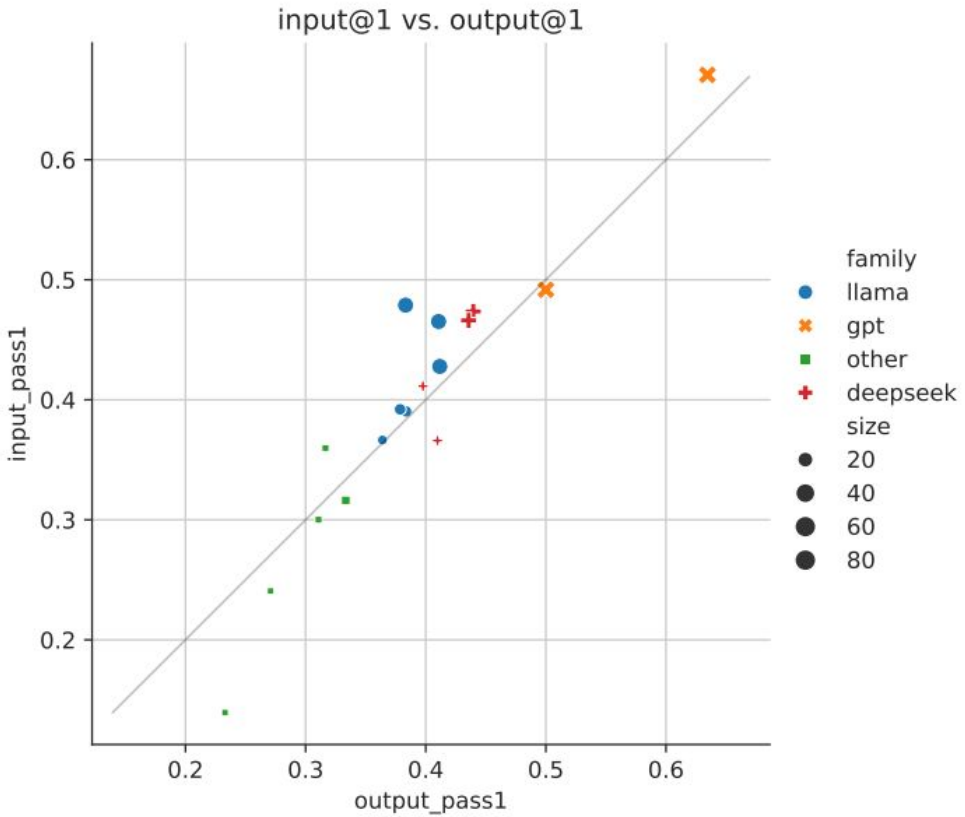


Models distilled on GPT data (WizardCoder, Phind, Phi) do NOT perform better than their base models!



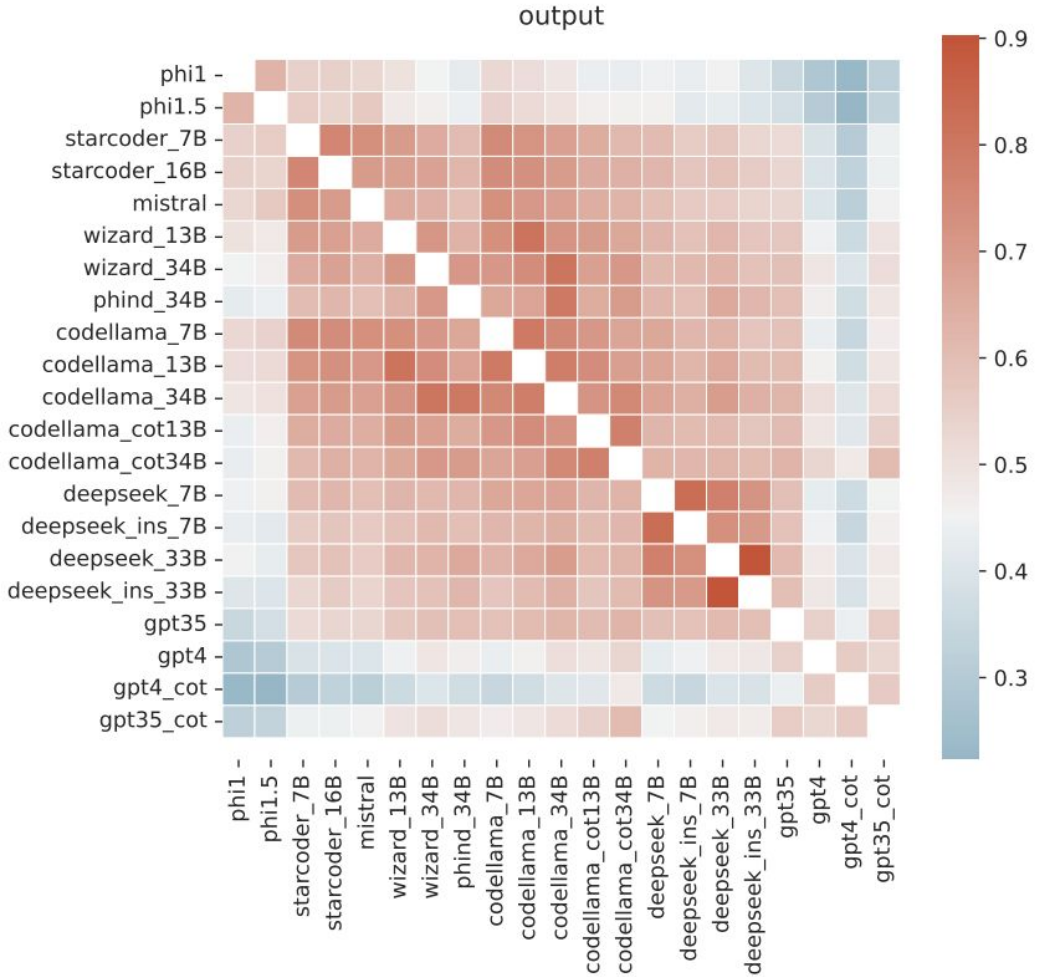
GPT-4 is a clear outlier!


input@1 vs. output@1
input_pass1
output_pass1
0.2
0.3
0.4
0.5
0.6
family
llama
gpt
other
deepseek
size
20
40
60
80



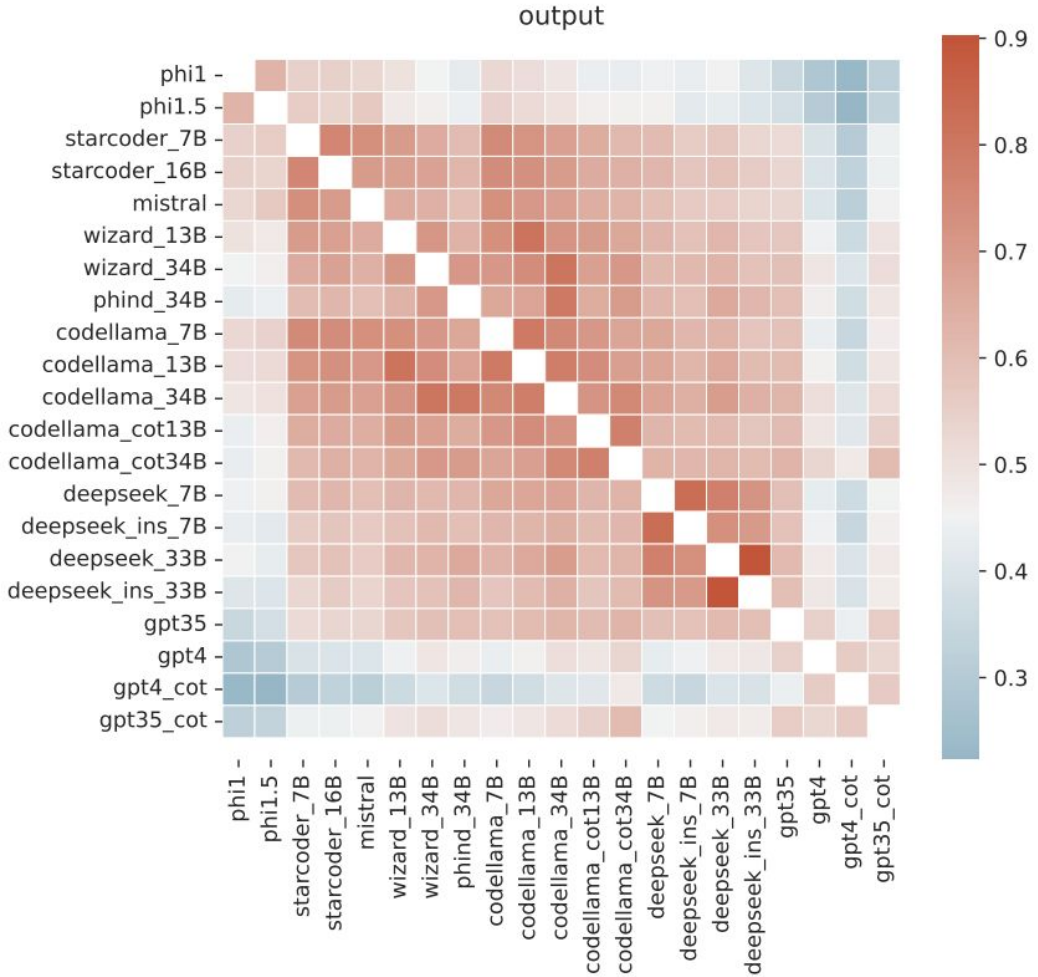
1. **Input and output prediction scores are correlated!**

2. **Clear improvements from model scaling!**



Strong correlations between:

- **Sizes of same model**
- **Models of same size**
- **Instruct-base pairs**



Strong correlations between:

- **Sizes of same model**
- **Models of same size**
- **Instruct-base pairs**

However, still many examples where worse models succeed but better models fail completely!

Part 1: Benchmark Creation

Creating a benchmark to measure code execution

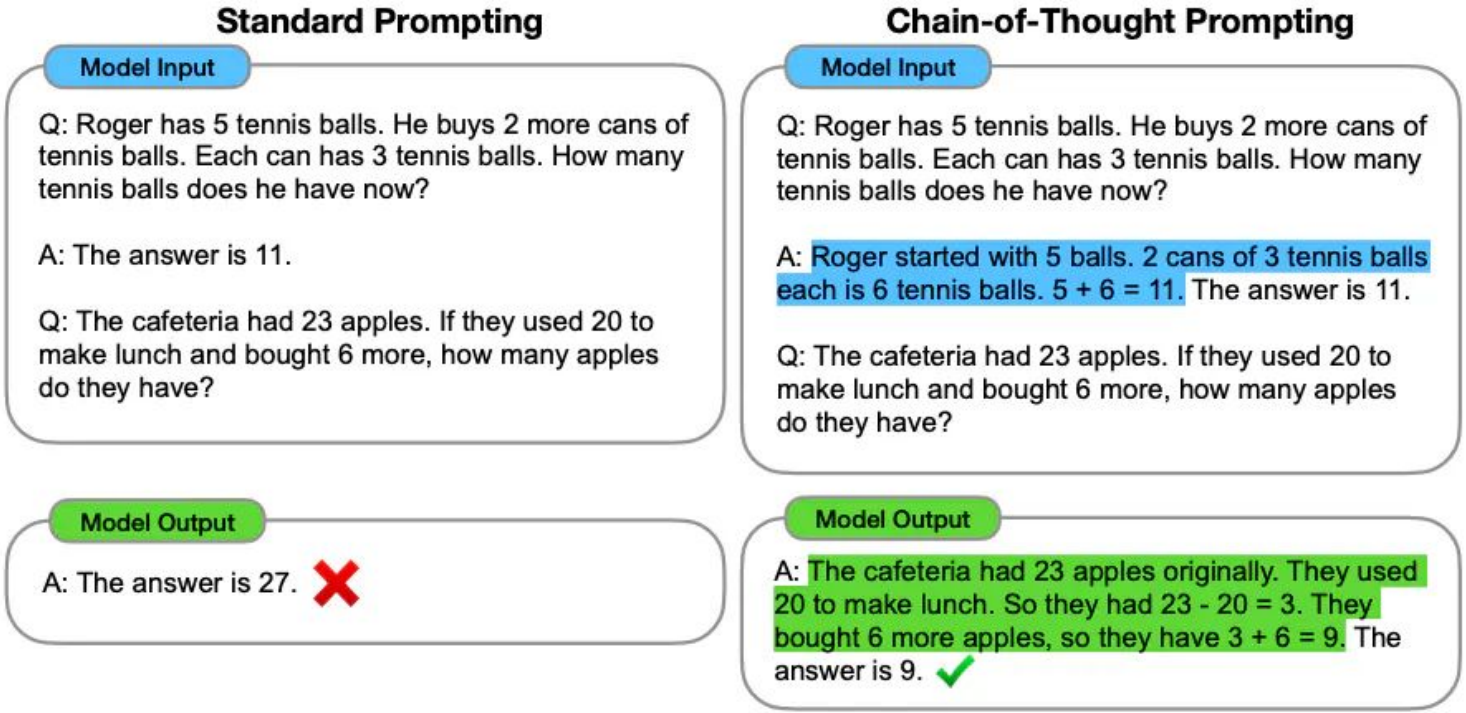
Part 2: Benchmark Evaluation

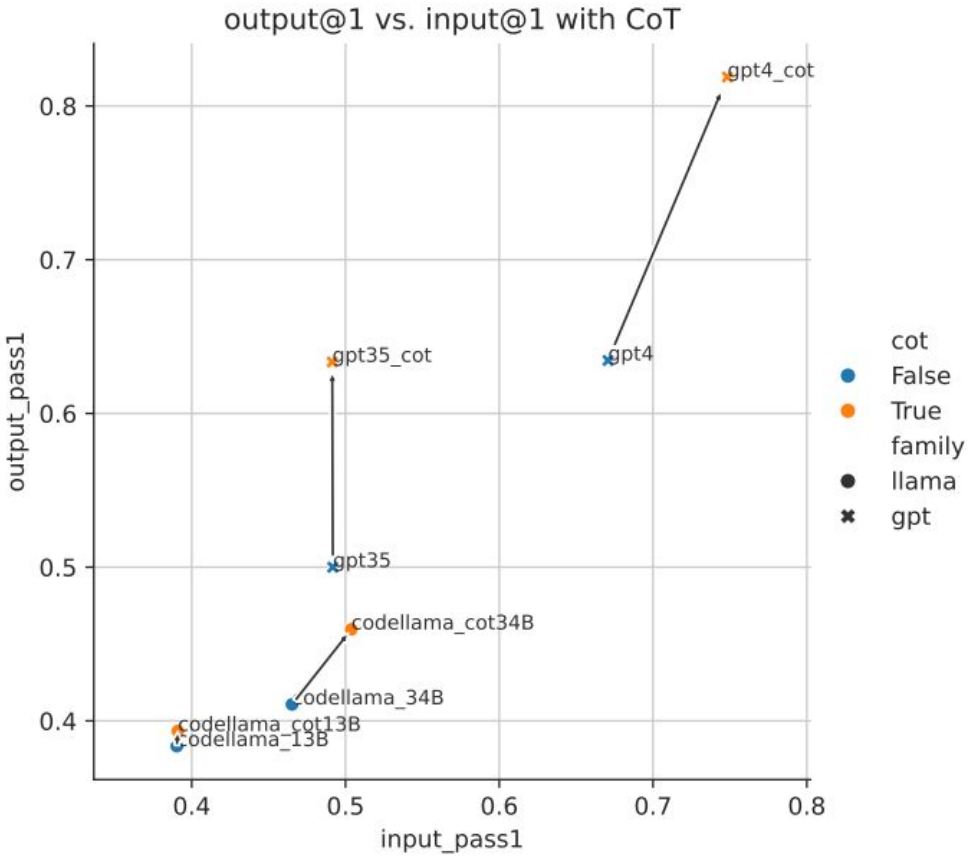
Can code LMs understand, reason, and execute code like us?

Part 3: CoT and Fine-Tuning

Are standard tricks enough to solve CRUXEval?

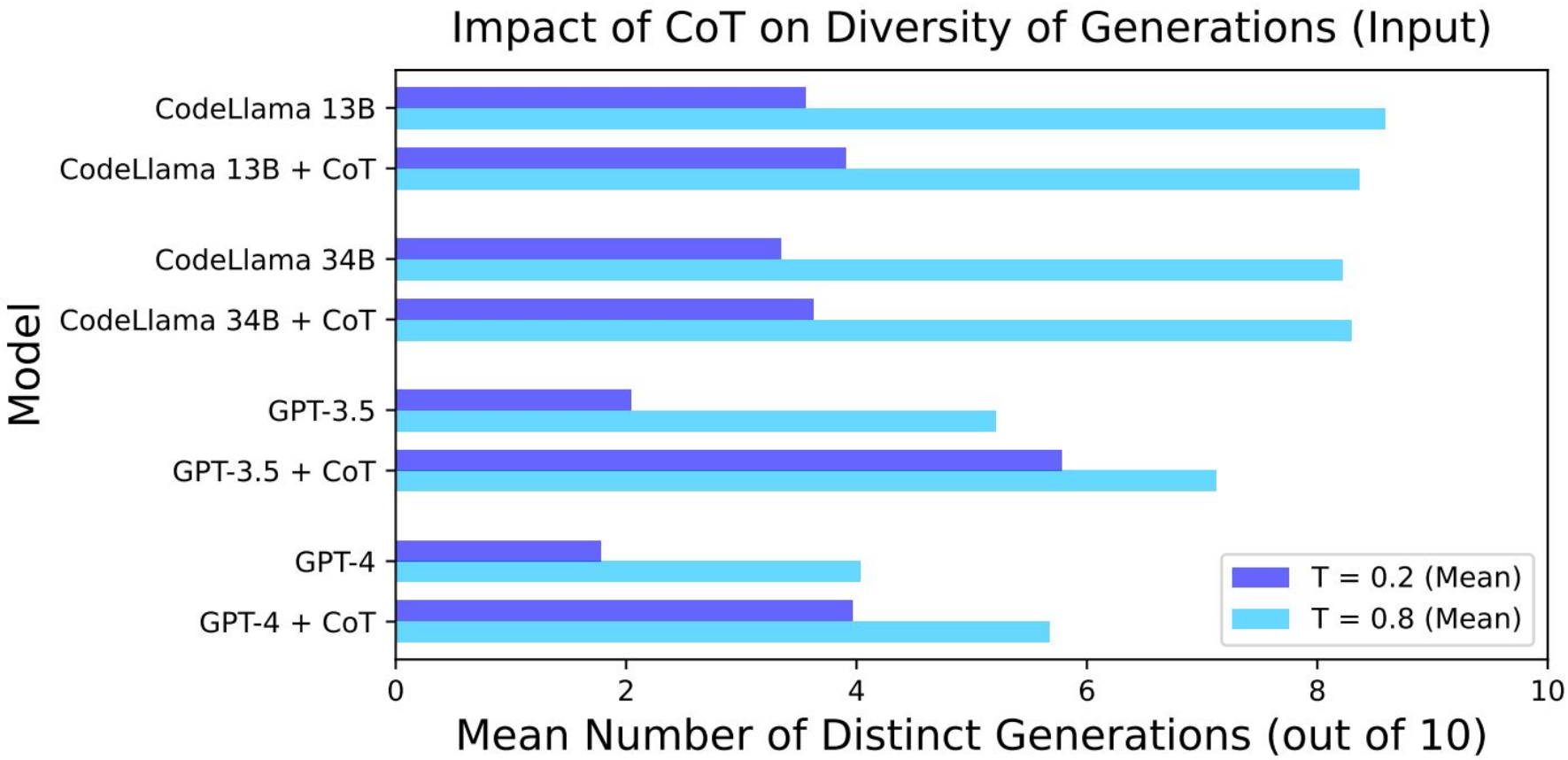
Chain-of-Thought Prompting



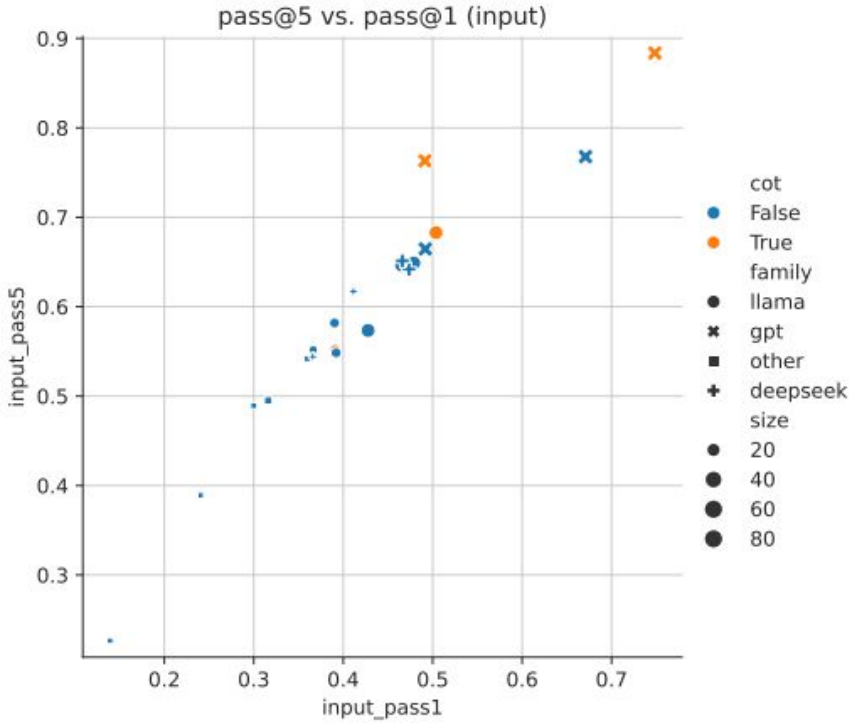


1. **GPT-4, GPT-3.5 output benefits significantly more than CoT than others**
2. **CoT helps more for output prediction than input prediction**
3. **Code Llamas benefit very slightly from CoT**

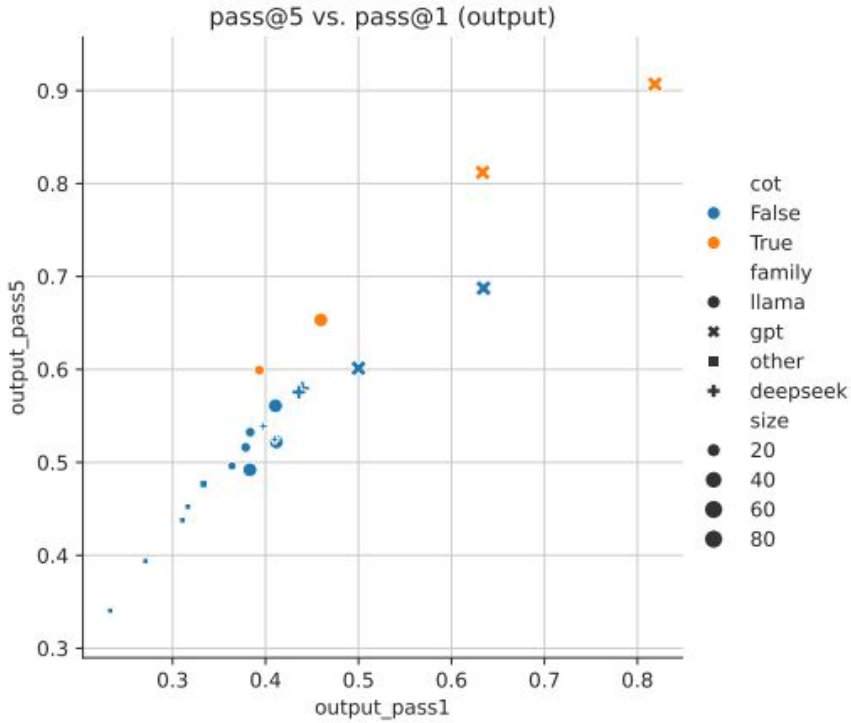
Model	CoT	Input Prediction		Output Prediction	
		Pass@1	Pass@5	Pass@1	Pass@5
Code Llama 13B	✗	39.0%	58.2%	38.4%	53.2%
	✓	39.1%	55.2%	39.3%	59.9%
	-	+0.1%	-3.0%	+0.9%	+6.7%
Code Llama 34B	✗	46.5%	64.7%	41.1%	56.1%
	✓	50.4%	68.3%	46.0%	65.3%
	-	+3.9%	+3.6%	+4.9%	+9.2%
GPT-3.5	✗	49.2%	66.5%	50.0%	60.1%
	✓	49.1%	76.3%	63.3%	81.2%
	-	-0.1%	+9.8%	+13.3%	+21.1%
GPT-4	✗	67.1%	76.8%	63.4%	68.7%
	✓	74.8%	88.4%	81.9%	90.7%
	-	+7.7%	+11.6%	+18.5%	+22.0%



CoT increases diversity of generations for GPT!

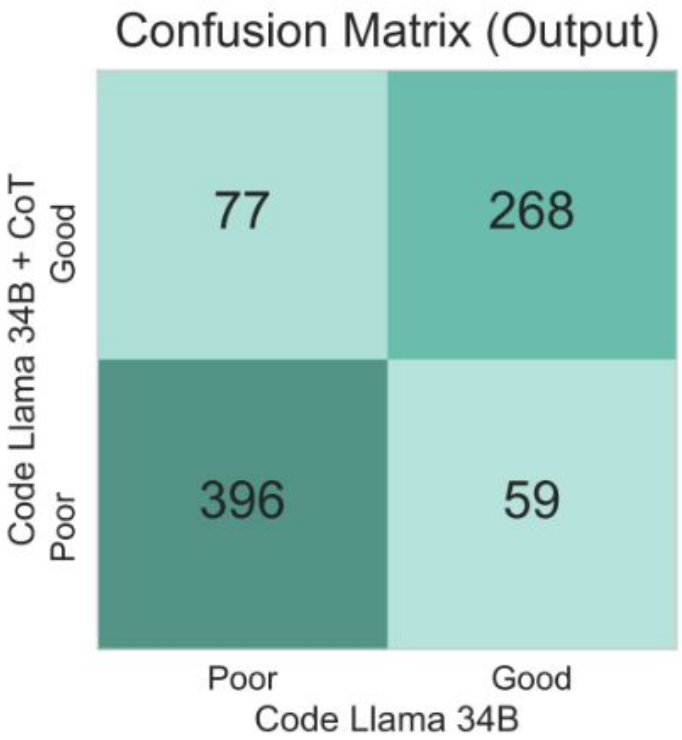


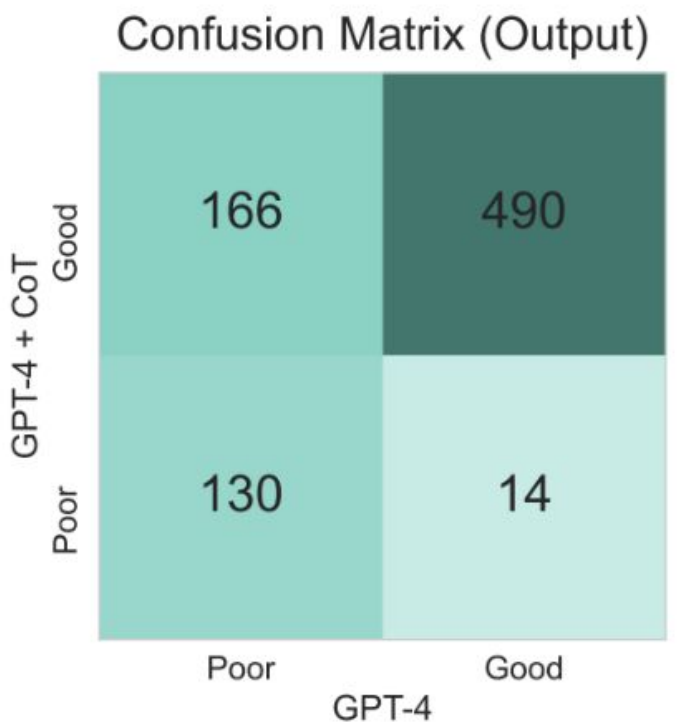
(a) Input prediction

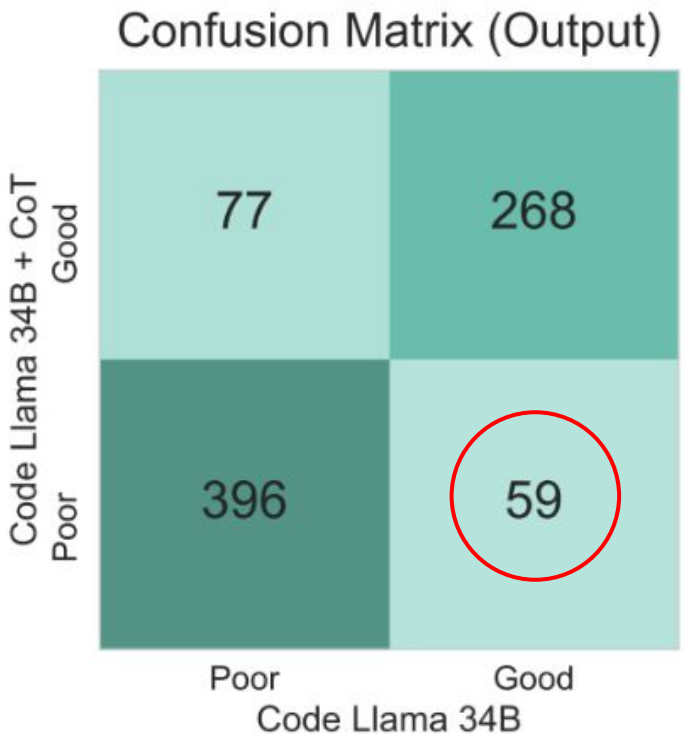


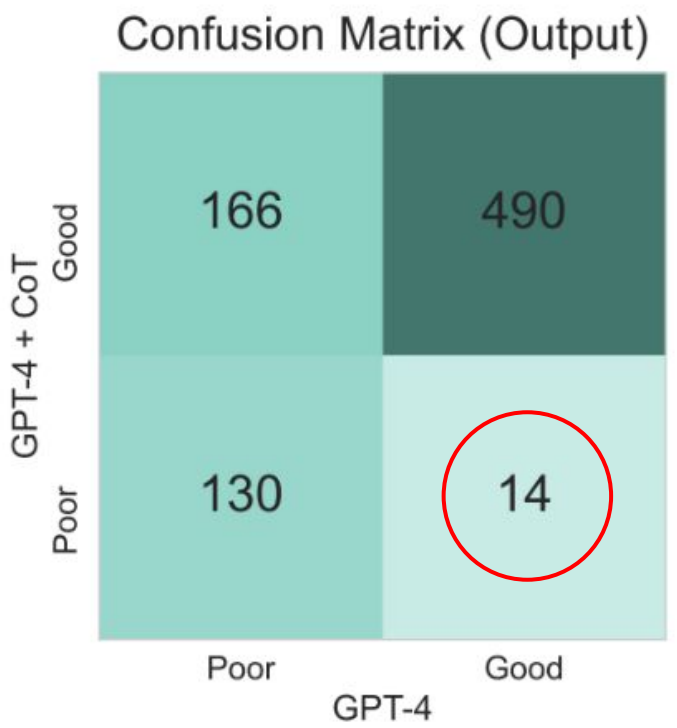
(b) Output prediction

CoT increases the gap between pass@1 and pass@5


Confusion Matrix (Output)
77
268
396
59
Code Llama 34B + CoT
Good
Poor
Poor
Good
Code Llama 34B


Confusion Matrix (Output)
166
490
130
14
GPT-4 + CoT
Good
Poor
Poor
Good
GPT-4





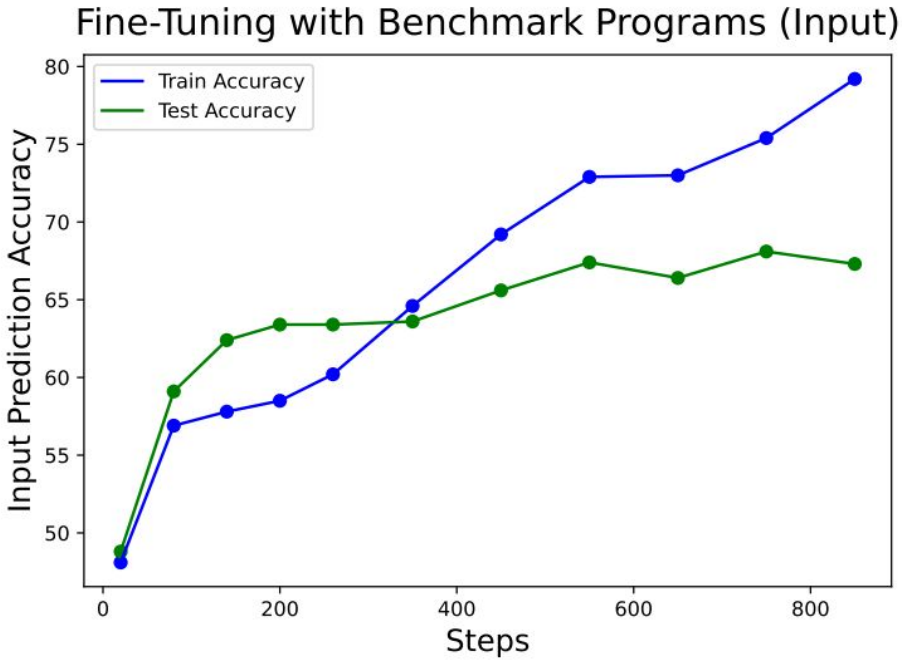
For CodeLlama 13B, 34B, and GPT-3.5, we observe a large number of samples where direct prediction succeeds but CoT fails.

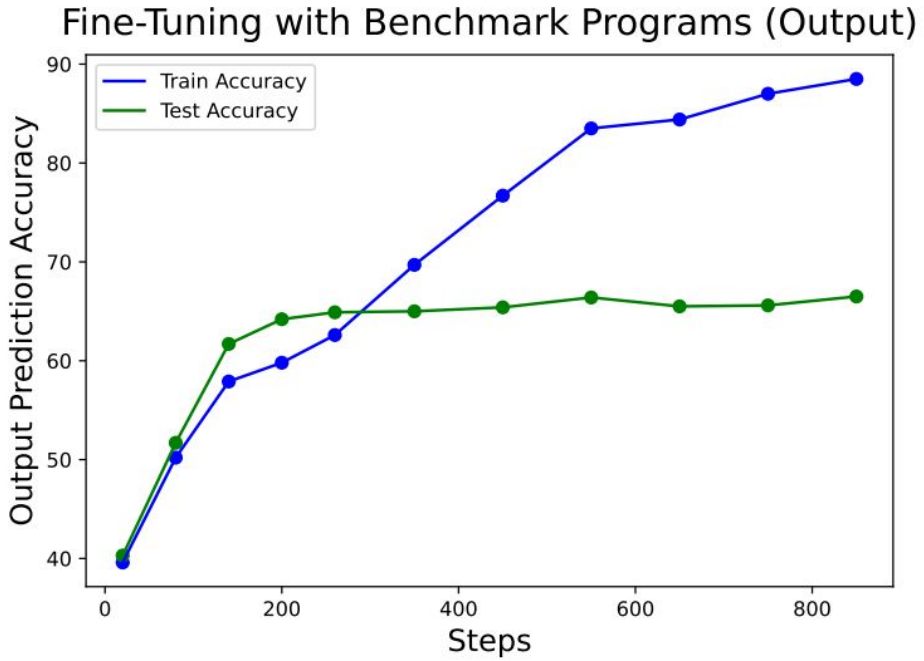
Fine-Tuning

Fine-tuned Code Llama 34B on ~140K function, input, and output samples

Decontamination: 50% function level, 50% test level

Separate fine-tuning for input and output tasks





1. **Easy to overfit training set**
2. **Significant improvements from baseline, but plateaus at 70%**

Our contributions

CRUXEval Benchmark	Evaluation on Popular LLMs	Enhancements and Insights
measuring code understanding skills that are easy for humans	*impact of distilling from GPT, correlation with HumanEval*	*chain-of-thought, fine-tuning*

Impact of the Work

Capabilities Beyond Generation

existing benchmarks are not enough to faithfully evaluate software engineering applications

Framework for Benchmarking

our benchmarking principles allow this methodology to be used to evaluate other aspects of code models

Highlighting Existing Progress

GPT-4 failures, limits of fine-tuning, CoT effectiveness gap

CRUXEval

Code **R**easoning, **U**nderstanding, and E**x**ecution **Eval**uation

💻 Website: https://crux-eval.github.io/

🏆 Leaderboard: https://crux-eval.github.io/leaderboard.html