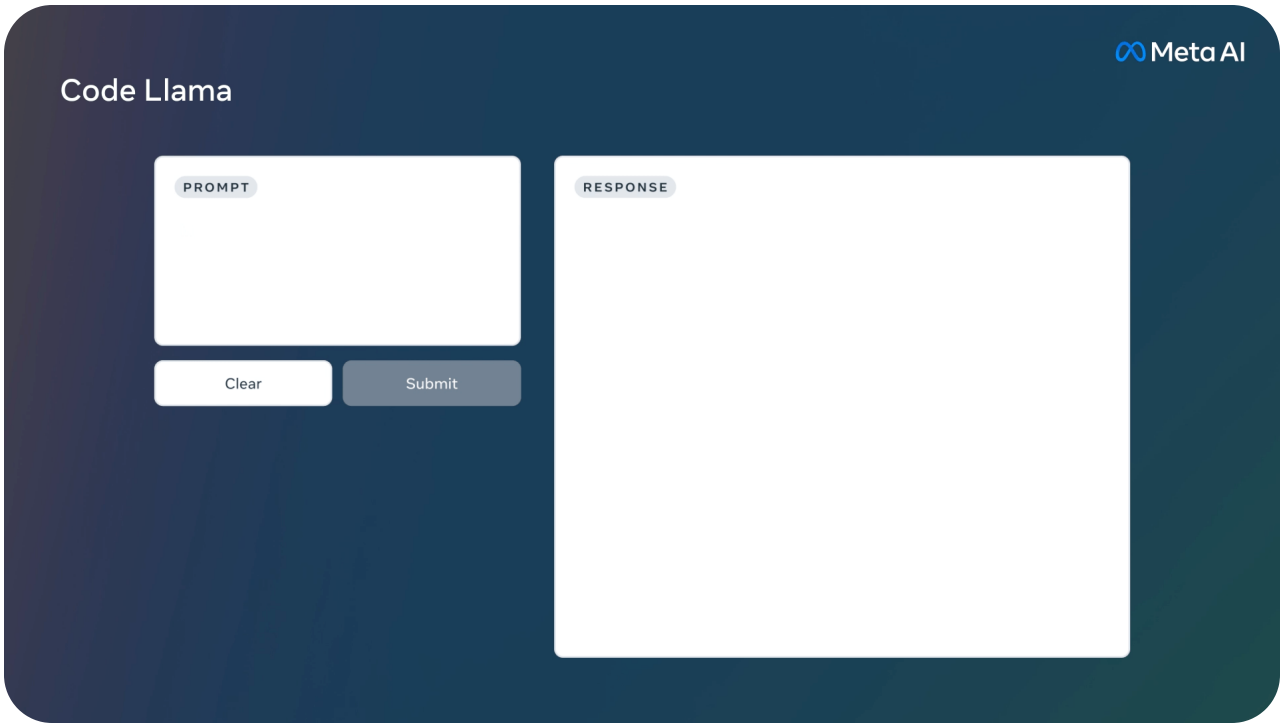
Meta

FEATURED

Large Language Model

Introducing Code Llama, a state-of-the-art large language model for coding

August 24, 2023



Takeaways

- Code Llama is a state-of-the-art LLM capable of generating code, and natural language about code, from both code and natural language prompts.
- Code Llama is free for research and commercial use.
- Code Llama is built on top of Llama 2 and is available in three models:
  - Code Llama, the foundational code model;
  - Codel Llama - Python specialized for Python;

Meta

- In our own benchmark testing, Code Llama outperformed state-of-the-art publicly available LLMs on code tasks

Today, we are releasing Code Llama, a large language model (LLM) that can use text prompts to generate code. Code Llama is state-of-the-art for publicly available LLMs on code tasks, and has the potential to make workflows faster and more efficient for current developers and lower the barrier to entry for people who are learning to code. Code Llama has the potential to be used as a productivity and educational tool to help programmers write more robust, well-documented software.

The generative AI space is evolving rapidly, and we believe an open approach to today's AI is the best one for developing new AI tools that are innovative, safe, and responsible. We are releasing Code Llama under the same community license as Llama 2.

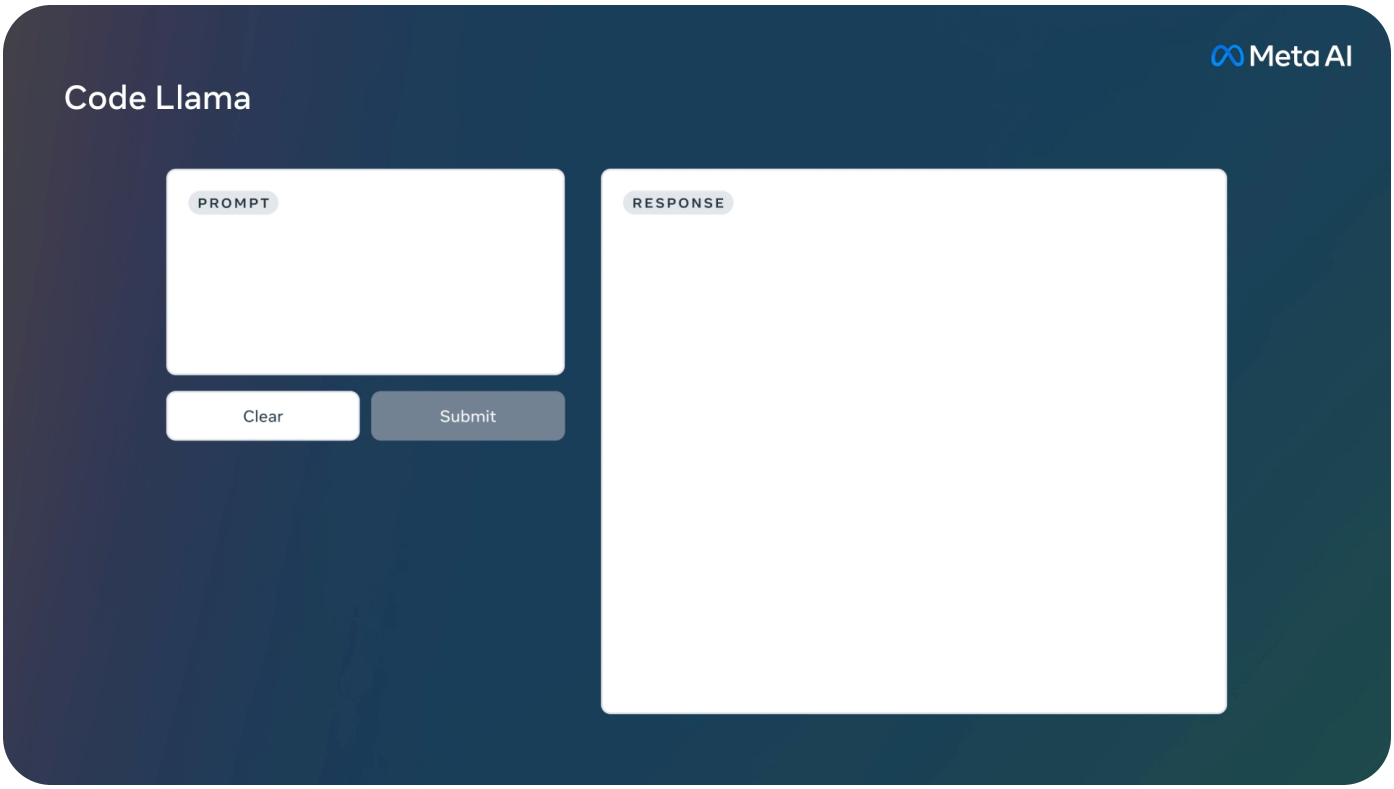
RECOMMENDED READS

- Code Llama research paper
- Code Llama GitHub
- Download the Code Llama model

How Code Llama works

Code Llama is a code-specialized version of Llama 2 that was created by further training Llama 2 on its code-specific datasets, sampling more data from that same dataset for longer. Essentially, Code Llama features enhanced coding capabilities, built on top of Llama 2. It can generate code, and natural language about code, from both code and natural language prompts (e.g., "Write me a function that outputs the fibonacci sequence.") It can

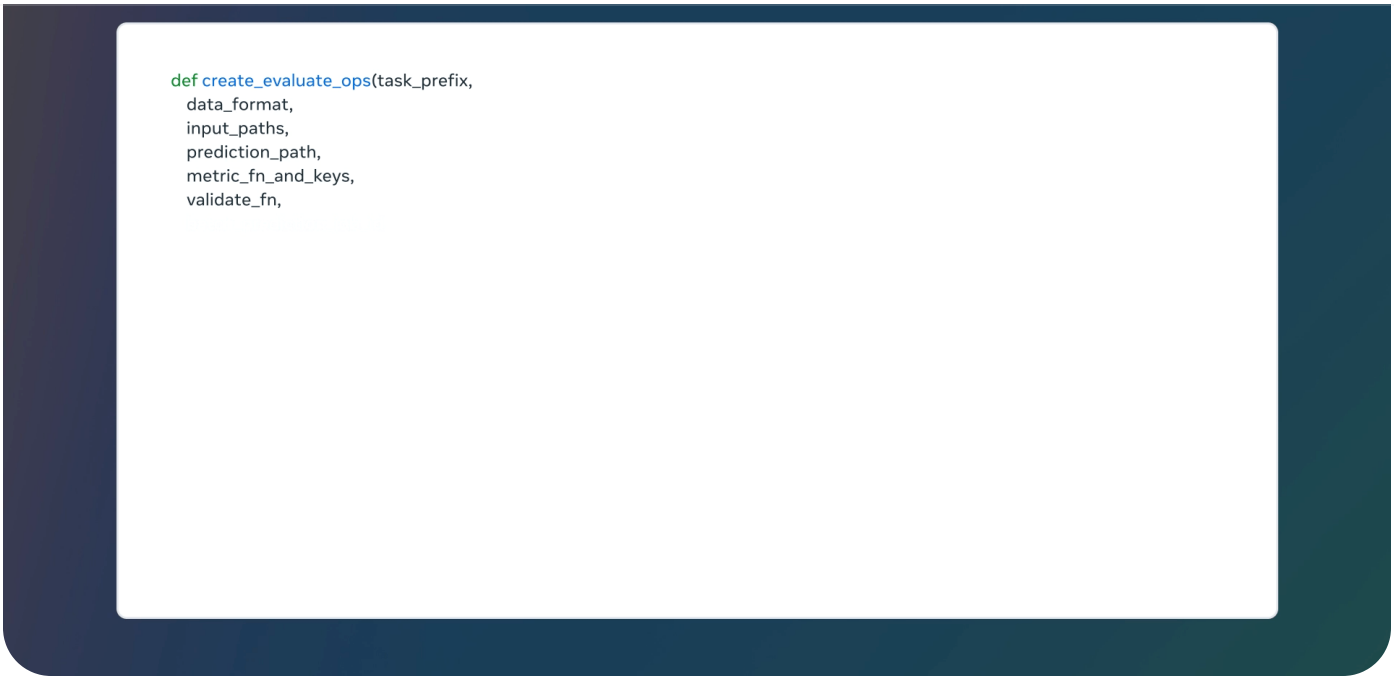
and Bash.



We are releasing three sizes of Code Llama with 7B, 13B, and 34B parameters respectively. Each of these models is trained with 500B tokens of code and code-related data. The 7B and 13B base and instruct models have also been trained with fill-in-the-middle (FIM) capability, allowing them to insert code into existing code, meaning they can support tasks like code completion right out of the box.

The three models address different serving and latency requirements. The 7B model, for example, can be served on a single GPU. The 34B model returns the best results and allows for better coding assistance, but the smaller 7B and 13B models are faster and more suitable for tasks that require low latency, like real-time code completion.

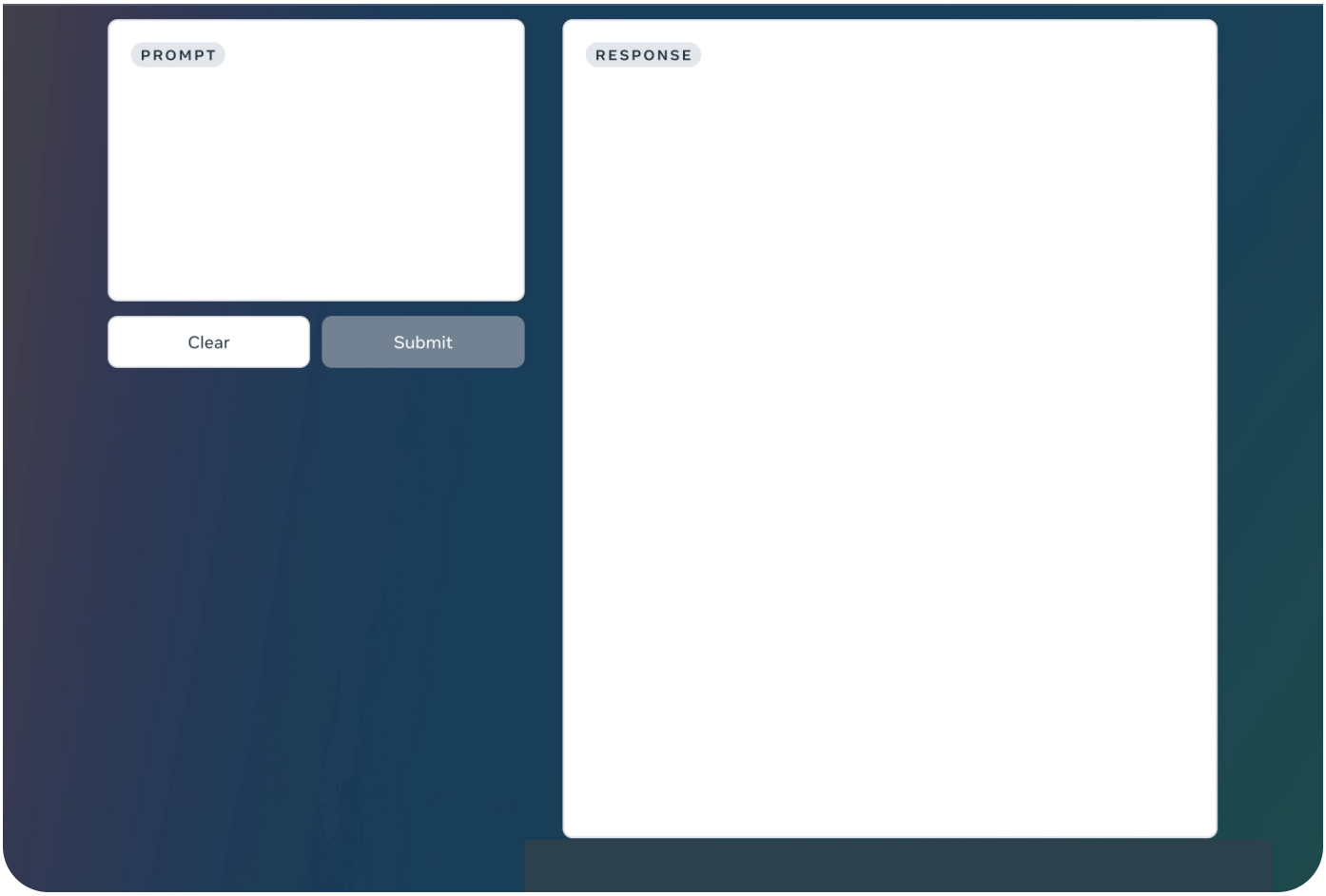
Meta



The Code Llama models provide stable generations with up to 100,000 tokens of context. All models are trained on sequences of 16,000 tokens and show improvements on inputs with up to 100,000 tokens.

Aside from being a prerequisite for generating longer programs, having longer input sequences unlocks exciting new use cases for a code LLM. For example, users can provide the model with more context from their codebase to make the generations more relevant. It also helps in debugging scenarios in larger codebases, where staying on top of all code related to a concrete issue can be challenging for developers. When developers are faced with debugging a large chunk of code they can pass the entire length of the code into the model.

Meta



Additionally, we have further fine-tuned two additional variations of Code Llama: Code Llama - Python and Code Llama - Instruct.

Code Llama - Python is a language-specialized variation of Code Llama, further fine-tuned on 100B tokens of Python code. Because Python is the most benchmarked language for code generation – and because Python and PyTorch play an important role in the AI community – we believe a specialized model provides additional utility.

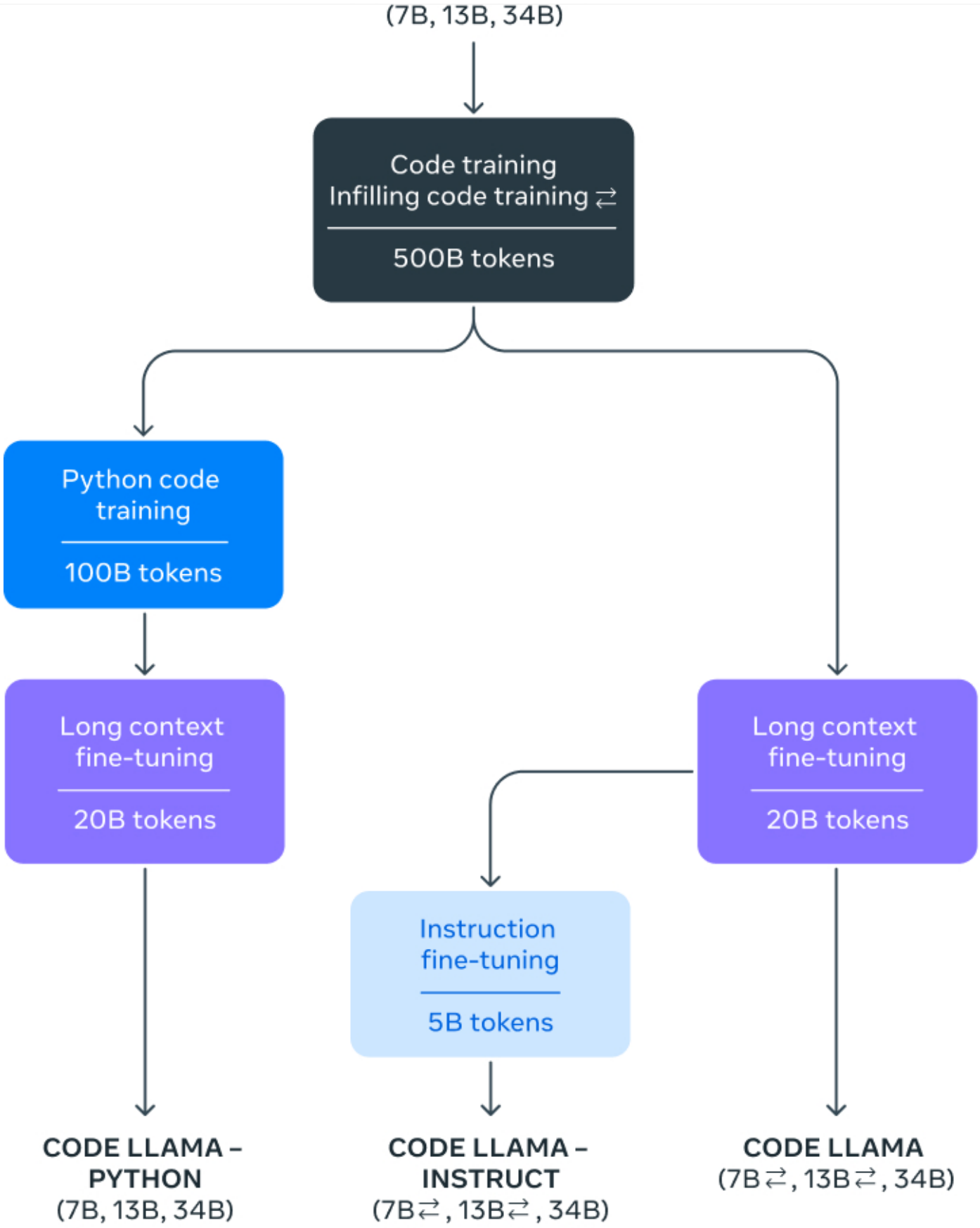
Code Llama - Instruct is an instruction fine-tuned and aligned variation of Code Llama. Instruction tuning continues the training process, but with a different objective. The model is fed a “natural language instruction” input and the expected output. This makes it better at understanding what humans expect out of their prompts. We recommend using Code Llama - Instruct variants whenever using Code Llama for code generation since Code Llama - Instruct has been fine-tuned to generate helpful and safe answers in natural language.

We do not recommend using Code Llama or Code Llama - Python to perform general natural language tasks since neither of these models are designed to follow natural

Meta

When using the Code Llama models, users must abide by our license and acceptable use policy.

Meta


(7B, 13B, 34B)
Code training
Infilling code training ⇄
500B tokens
Python code training
100B tokens
Long context fine-tuning
20B tokens
Long context fine-tuning
20B tokens
Instruction fine-tuning
5B tokens
CODE LLAMA – PYTHON
(7B, 13B, 34B)
CODE LLAMA – INSTRUCT
(7B⇄, 13B⇄, 34B)
CODE LLAMA
(7B⇄, 13B⇄, 34B)

Evaluating Code Llama's performance

To test Code Llama's performance against existing solutions, we used two popular coding benchmarks: HumanEval and Mostly Basic Python Programming (MBPP). HumanEval tests the model's ability to complete code based on docstrings and MBPP tests the model's ability to write code based on a description.

Our benchmark testing showed that Code Llama performed better than open-source, code-specific LLMs and outperformed Llama 2. Code Llama 34B, for example, scored 53.7% on HumanEval and 56.2% on MBPP, the highest compared with other state-of-the-art open solutions, and on par with ChatGPT.

Meta

Model		HumanEval (pass@1)	MBPP (pass@1)	Multilingual Human Eval (pass@1)
Codex		33.5	45.9	26.1
GPT 3.5		48.1	52.2	-
GPT 4		67.0	-	-
Palm-Coder		36.0	47.0	-
StarCoder Python		33.6	52.7	25.3
StarCoder (prompted)		40.8	49.5	-
Llama 2 (70B)		30.5	45.4	24.4
Code Llama	7B	33.5	41.4	26.3
	13B	36.0	47.0	30.6
	34B	48.8	55.0	36.4
Code Llama - Instruct	7B	34.8	44.4	25.8
	13B	42.7	49.4	32.0
	34B	41.5	57.0	36.1
Code Llama - Python	7B	38.4	47.6	27.5
	13B	43.3	49.0	31.5
	34B	53.7	56.2	35.1

As with all cutting edge technology, Code Llama comes with risks. Building AI models responsibly is crucial, and we undertook numerous safety measures before releasing Code Llama. As part of our red teaming efforts, we ran a quantitative evaluation of Code Llama's

ChatGPT's (GPT3.5 Turbo). Our results found that Code Llama answered with safer responses.

Details about our red teaming efforts from domain experts in responsible AI, offensive security engineering, malware development, and software engineering are available in our research paper.

Releasing Code Llama

Programmers are already using LLMs to assist in a variety of tasks, ranging from writing new software to debugging existing code. The goal is to make developer workflows more efficient, so they can focus on the most human centric aspects of their job, rather than repetitive tasks.

At Meta, we believe that AI models, but LLMs for coding in particular, benefit most from an open approach, both in terms of innovation and safety. Publicly available, code-specific models can facilitate the development of new technologies that improve peoples' lives. By releasing code models like Code Llama, the entire community can evaluate their capabilities, identify issues, and fix vulnerabilities.

Code Llama's training recipes are available on our Github repository.

Model weights are also available.

Responsible use

Our research paper discloses details of Code Llama's development as well as how we conducted our benchmarking tests. It also provides more information into the model's limitations, known challenges we encountered, mitigations we've taken, and future challenges we intend to investigate.

We've also updated our Responsible Use Guide and it includes guidance on developing downstream models responsibly, including:

- Defining content policies and mitigations.
- Preparing data.
- Fine-tuning the model.
- Evaluating and improving performance.

Meta

Developers should evaluate their models using code-specific evaluation benchmarks and perform safety studies on code-specific use cases such as generating malware, computer viruses, or malicious code. We also recommend leveraging safety datasets for automatic and human evaluations, and red teaming on adversarial prompts.

The future of generative AI for coding

Code Llama is designed to support software engineers in all sectors – including research, industry, open source projects, NGOs, and businesses. But there are still many more use cases to support than what our base and instruct models can serve.

We hope that Code Llama will inspire others to leverage Llama 2 to create new innovative tools for research and commercial products.

Try Code Llama today

Code Llama GitHub repository

Download the Code Llama Model

Read the research paper

Code Llama: Open foundation models for code

Share:

Meta

Our latest updates delivered to your inbox

Subscribe to our newsletter to keep up with Meta AI news, events, research breakthroughs, and more.

Join us in the pursuit of what's possible with AI.

See all open positions

Related Posts

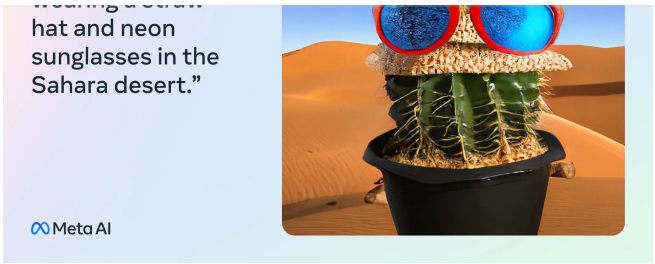
Research

Meta and Microsoft Introduce the Next Generation of Llama

July 18, 2023

Read post

Meta



Research

Introducing CM3leon, a more efficient, state-of-the-art generative model for text and images

July 14, 2023

Read post

Large Language Model

Community-driven AI innovation comes alive with Llama 2

July 28, 2023

Read post

Search AI content

Meta

Latest Work

Our Actions

Newsletter

Privacy Policy
Terms
Cookies