MAI-Code-1-Flash model card

Model Summary

Developer	Microsoft Corporation Authorized Representative: Microsoft Ireland Operations Limited (MIOL) 70 Sir John Rogerson's Quay, Dublin 2, D02 R296, Ireland
Description	MAI-Code-1-Flash is a text-to-text coding model built for fast, efficient assistance in everyday developer workflows. The model is optimized to deliver high-quality coding help with low latency and low serving cost, making it well-suited for tasks such as agentic coding in real repositories, repository question answering, refactoring, and tool-using developer scenarios.
Model architecture	The model uses a transformer architecture with self-attention using sparse Mixture-of-Experts layers.
Parameters	137B
Inputs	Text
Context length	256K tokens
Outputs	Text
Public data summary	https://microsoft.ai/pdf/MAI-Code-1-Flash-Data-Card.PDF
Training dates	March 2026 - May 2026
Release date	June 2, 2026
Release date in the EU	June 2, 2026
License	Various product and service terms where the model is deployed, such as those for Visual Studio Code.
Model dependencies	MAI-Thinking-1 (Coming soon to the EU)

Additional related assets	N/A
Acceptable use policy	As described in the License section above. Subsequent integrations of MAI-Code-1-Flash may be subject to different terms of service and policies.

Key capabilities

About this model

MAI-Code-1-Flash is a text-to-text coding model built by Microsoft for fast, efficient assistance in everyday developer workflows. The model is optimized to deliver high-quality coding help with low latency and low serving cost, making it well-suited for tasks such as agentic coding in real repositories, repository question answering, refactoring, and tool-using developer scenarios inside GitHub Copilot in Visual Studio Code. Rollout to GitHub Copilot CLI is planned for a later phase.

Key model capabilities

- Agentic coding in real developer environments, trained directly with the GitHub Copilot harness.
- Adaptive solution-length control, stays concise for simple requests and spends more reasoning budget on complex tasks.
- Strong instruction-following across single-turn and multi-turn scenarios.
- Competitive reasoning across math, science, and visual coding tasks.

Key use cases

MAI-Code-1-Flash is a coding-focused generative model intended for interactive developer assistance and agentic coding tasks inside GitHub Copilot in Visual Studio Code. Primary use cases include code generation and completion, repository question answering, refactoring, telemetry-grounded coding tasks, and agentic tool use. GitHub Copilot CLI support is planned for a later rollout. The relevant terms of service and acceptable use policy for GitHub Copilot Business and Enterprise users can be found here, and the terms of service and acceptable use policy for Copilot Free, Pro, and Pro+ users can be found here.

Pricing

To be finalized

Technical specs

Training cut-off date

December 2025

Supported languages

English

Optimizing model performance

MAI-Code-1-Flash was trained directly with the GitHub Copilot harness used in production, so offline improvements translate into real-world developer quality. It also uses adaptive solution length control, staying concise about simpler requests and spending more reasoning budget on harder problems.

Training disclosure

Training, testing and validation

MAI-Code-1-Flash was trained end to end by Microsoft on carefully curated data using Microsoft infrastructure and data pipelines. The development pipeline spans pretraining, midtraining, supervised fine-tuning, and reinforcement learning, starting from MAI-Thinking-1's mid-training checkpoint.

A lightweight supervised fine-tuning stage on curated instruction-following and agentic task data was applied on top of that mid-training checkpoint to establish reliable instruction- and format-following behavior. An additional "mid2" training phase used approximately 2 million diverse synthetic agentic tasks, organized into two progressive stages from simpler to more complex scenarios. A final large-scale reinforcement learning stage was conducted on diverse agentic tasks spanning more than 150,000 environments to strengthen per-token intelligence and end-to-end task quality. Synthetic data techniques — including prompt rewriting, rubrics synthesization, process supervision, and repo-level synthesization — were used throughout to maintain learnability as the model improved.

For testing and validation, MAI-Code-1-Flash was evaluated using both public benchmarks and internal benchmarks built from real developer scenarios. All evaluations were run with the GitHub Copilot production harness used to serve users, covering software engineering tasks, repository question answering, refactoring, and telemetry-grounded tasks adapted from real GitHub Copilot usage, so offline measurements reflect real-world model behavior.

Distribution

Distribution channels

Available only in GitHub Copilot in Visual Studio Code at launch, starting with a fraction of individual users. No additional setup is required; GitHub Copilot may route tasks to MAI-Code-1-Flash through the Auto picker, or the model may be available directly in the model picker as rollout progresses. Any future release formats, such as API release, would be accompanied by an update to the relevant documentation.

Responsible AI considerations

Safety techniques

Safety is addressed across the full training pipeline. During pre-training, harmful content is filtered out or demoted in the data mixture to reduce exposure during base model learning. In later training phases — including supervised fine-tuning and reinforcement learning — alignment techniques are applied to shape model behavior, reinforce safe and helpful responses, and discourage harmful, unsafe, or otherwise undesirable outputs.

Safety evaluations

This model was evaluated for safety using a combination of cybersecurity benchmarks and secure coding assessments, including CyberBench, CyberSecEval, and SecRepo, to assess its ability to withstand real-world security threats, avoid introducing vulnerabilities, and align with secure coding standards. Additional safety evaluations were conducted through a structured release process using production model APIs with safety classifiers and filters applied. These layered methodologies help ensure the model meets rigorous safety and security requirements before deployment.

Known limitations

As with any large language model, AI-generated text and code from MAI-Code-1-Flash may be inaccurate, incomplete, or otherwise incorrect. Developers should review, test, and validate outputs before relying on them in production or other consequential contexts.

Acceptable use

Acceptable use policy

MAI-Code-1-Flash is intended for commercial use within GitHub Copilot and its supported clients (currently Visual Studio Code, with GitHub Copilot CLI planned for a later rollout). Use of the model is governed by the applicable GitHub Copilot terms of service and

Microsoft's Acceptable Use Policy, including restrictions on generating harmful, unlawful, or otherwise prohibited content.

Quality and performance evaluations

MAI-Code-1-Flash was evaluated across a broad set of benchmarks covering agentic coding, reasoning, instruction following, and tool use.

Coding:

	MAI-Code-1-Flash		Claude Haiku 4.5	
Benchmark	Pass rate	Avg token usage (K)	Pass rate[1]	Avg token usage (K)
SWE-Bench Verified Agentic coding	71.6	10.8	66.6	27.3
SWE-Bench Pro Diverse agentic coding	51.2	28.0	35.2	29.8
SWE-Bench Multilingual Multilingual coding	65.5	15.3	62.7	17.2
Terminal Bench 2 Agentic terminal coding	54.8	21.6	41.6	25

1 Numbers from internal benchmark system with production harness

Core reasoning capabilities in math, science, visual generation coding:

	MAI-Code-1-Flash		**Claude Haiku 4.5**	
	Acc	Avg token usage (K)	Acc	Avg token usage (K)
AIME 2026 **Competitive math**	92.5	23.6	83.3	30.3
AMO Bench **Olympiad math**	40	56	16	46.2
Frontier Math **Tier 1-3**	6.3	31.2	2.8	38.7
HLE **Academic reasoning, text**	18	26.3	9.5	22.2
GPQA Diamond **Biology, chemistry, physics**	84.6	9.6	73.2	14.6
Frontier Science **Scientific reasoning**	58.2	20.5	42.3	24.1
Artifacts Bench **Visual coding**	36.4	12.0	36.6	23.6

Instruction following and agentic tool use:

		MAI-Code-1-Flash	Claude Haiku 4.5
IF Bench Precise instruction following	average	75.0	46.1

Advanced IF Rubric-based IF	average	71.4	56.9
Robust IF Bench Internal diverse hard IF	average	61.2	45
τ^2 -Bench Agentic tool use	telecom	71.7	54.7

Benchmarking methodology

SWE-Bench Verified, SWE-Bench Pro, and SWE-Bench Multilingual were evaluated in the same VS Code-based harness used to evaluate production GitHub Copilot workflows. Pass rates are measured end to end, with repository context, tool calls, and verification included, rather than in a stripped-down benchmark environment. Solution length is reported as the average number of tokens used per completed task; shorter responses are better when pass rates are comparable. All compared models were evaluated in the same harness with the same settings to keep the comparison consistent.

Public data summary

Pre-training used a data mixture combining diverse, high-quality sources — including web text, code, books, and technical documents — carefully deduplicated and filtered to maximize knowledge coverage while reducing memorization of low-quality content. Post-training used synthetic and open-source coding datasets spanning SWE task completion (SWE-Bench variants, repository sampling, and bug-bounty data), codebase question answering (repository- and file-level QA), and instruction following for developer workflows (debug, explain, build, and improve). To read more about the data used to train MAI-Code-1-Flash, please see the public data summary here.