8. Frontier Topics

The Frontier of Code Intelligence

What is **the ultimate goal** of Code Intelligence?

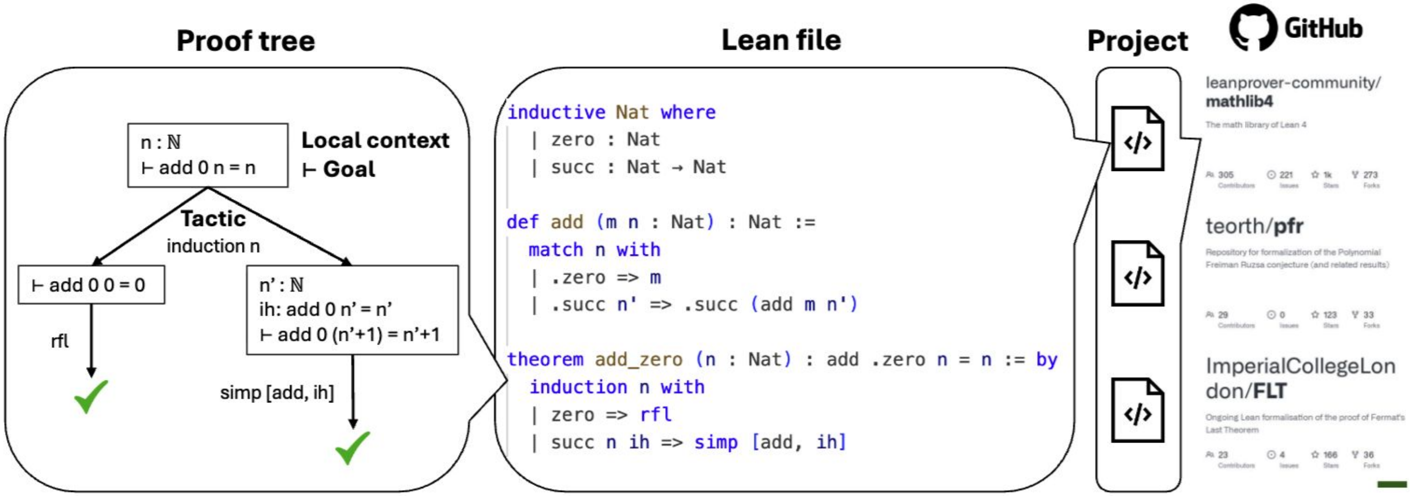
Write correct code for software? Not really...

Code is the **tool & interface** that connects to the real world.

Think Beyond Writing Code.

Code Language Models for FORMAL MATH

- Formal languages like Lean can be used to write not only conventional programs but also mathematical definitions, theorems, and proof.

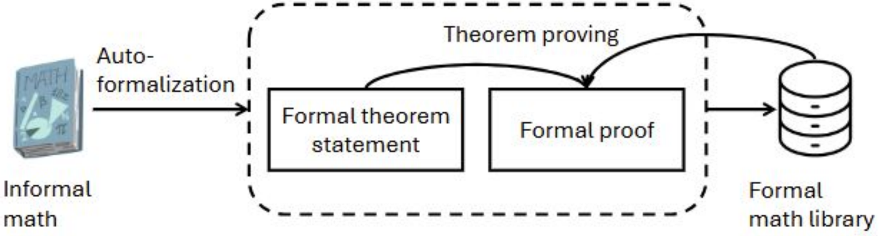


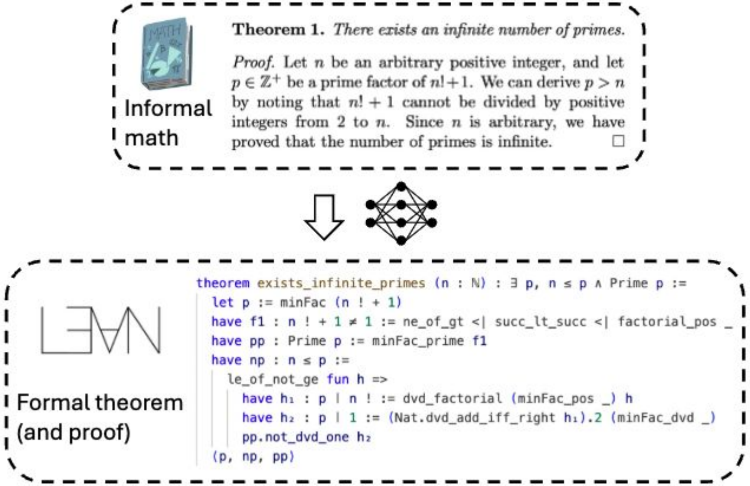
De Moura, L., Kong, S., Avigad, J., Van Doorn, F., & von Raumer, J. (2015, July). The Lean theorem prover (system description). In International Conference on Automated Deduction (pp. 378-388). Cham: Springer International Publishing.

Code Language Models for FORMAL MATH

- Code LMs can help automating proofs.

Given informal mathematics, *autoformalization* automatically translates it into formal theorems and proofs, and then *theorem proving* generates formal proofs.

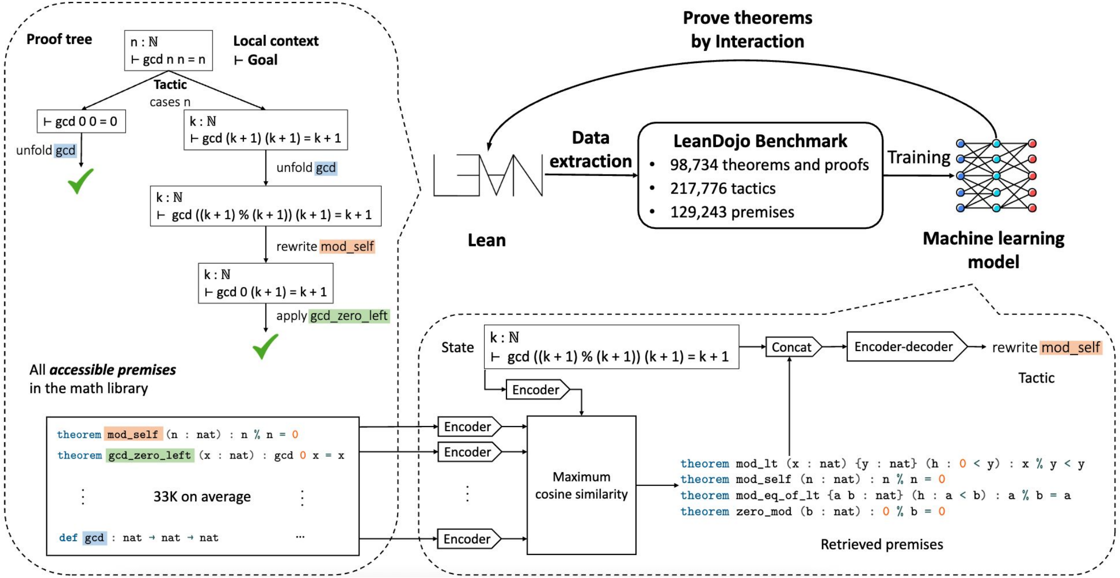




Yang, K., Poesia, G., He, J., Li, W., Lauter, K. E., Chaudhuri, S., & Song, D. (2025). Position: Formal Mathematical Reasoning—A New Frontier in AI. In Forty-second International Conference on Machine Learning Position Paper Track.

Code Language Models for FORMAL MATH

- ➢ LeanDojo extracts data from Lean, enables interaction with the proof environment programmatically, and uses an LM-based prover to augment with retrieval for selecting premises from a vast math library.



Yang, K., Swope, A., Gu, A., Chalamala, R., Song, P., Yu, S., ... & Anandkumar, A. (2023). Leandojo: Theorem proving with retrieval-augmented language models. Advances in Neural Information Processing Systems, 36, 21573-21612.

Code Language Models for FORMAL MATH

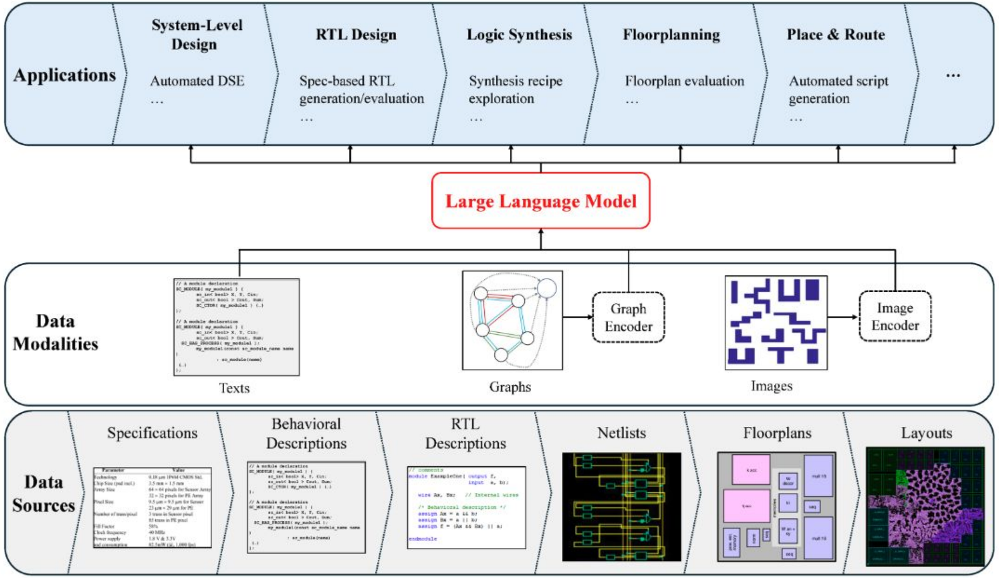
➢ Verified Code Generation: jointly generating code, specifications, and proofs of code-specification alignment

```
1  -- Description of the coding problem in natural language
2  -- Remove an element from a given array of integers at a specified index. The resulting array should
3  -- contain all the original elements except for the one at the given index. Elements before the
4  -- removed element remain unchanged, and elements after it are shifted one position to the left.
5
6  -- Code implementation
7  def removeElement (s : Array Int) (k : Nat) (h_precond : removeElement_pre s k) : Array Int :=
8    s.eraseIdx! k
9
10 -- Pre-condition
11 def removeElement_pre (s : Array Int) (k : Nat) : Prop :=
12   k < s.size -- the index must be smaller than the array size
13
14 -- Post-condition
15 def removeElement_post (s : Array Int) (k : Nat) (result: Array Int) (h_precond : removeElement_pre s k) : Prop :=
16   result.size = s.size - 1 ∧ -- Only one element is removed
17   (∀ i, i < k → result[i]! = s[i]!) ∧ -- The elements before index k remain unchanged
18   (∀ i, i < result.size → i ≥ k → result[i]! = s[i + 1]!) -- The elements after index k are shifted by one position
19
20 -- Proof
21 theorem removeElement_spec (s: Array Int) (k: Nat) (h_precond : removeElement_pre s k) :
22   removeElement_post s k (removeElement s k h_precond) h_precond := by sorry -- The proof is omitted for brevity
23
24 -- Test cases
25 (s : #[1, 2, 3, 4, 5]) (k : 2) (result : #[1, 2, 4, 5]) -- Positive test with valid inputs and output
26 (s : #[1, 2, 3, 4, 5]) (k : 5) -- Negative test: inputs violate the pre-condition at Line 12
27 (s : #[1, 2, 3, 4, 5]) (k : 2) (result : #[1, 2, 4]) -- Negative test: output violates the post-condition at Line 16
28 (s : #[1, 2, 3, 4, 5]) (k : 2) (result : #[2, 2, 4, 5]) -- Negative test: output violates the post-condition at Line 17
29 (s : #[1, 2, 3, 4, 5]) (k : 2) (result : #[1, 2, 4, 4]) -- Negative test: output violates the post-condition at Line 18
```

Ye, Z., Yan, Z., He, J., Kasriel, T., Yang, K., & Song, D. (2025). VERINA: Benchmarking Verifiable Code Generation. arXiv preprint arXiv:2505.23135.

Code Language Models for HARDWARE

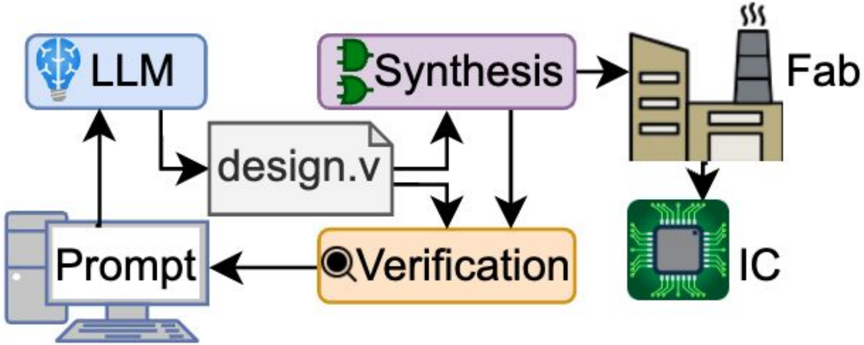
➢ LMs revolutionize electronic design automation with the code generation capabilities.



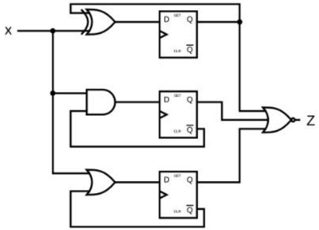
Pan, J., Zhou, G., Chang, C. C., Jacobson, I., Hu, J., & Chen, Y. (2025). A survey of research in large language models for electronic design automation. ACM Transactions on Design Automation of Electronic Systems, 30(3), 1–21.

Code Language Models for HARDWARE

➢ LMs can help fabricate the chips, with a focus on hardware language (e.g., Verilog) generation.



Given the finite state machine circuit described below, assume that the D flip-flops are initially reset to zero before the machine begins.
Build this circuit in Verilog.
Input x goes to three different two-input gates: a XOR, an AND, and a OR gate. Each of the three gates is connected to the input of a D flip-flop and then the flip-flop outputs all go to a three-input XNOR, whose output is Z. The second input of the XOR is its corresponding flip-flop's output, the second input of the AND is its corresponding flip-flop's complemented output, and finally the second input of the OR is its corresponding flip-flop's complementary output.

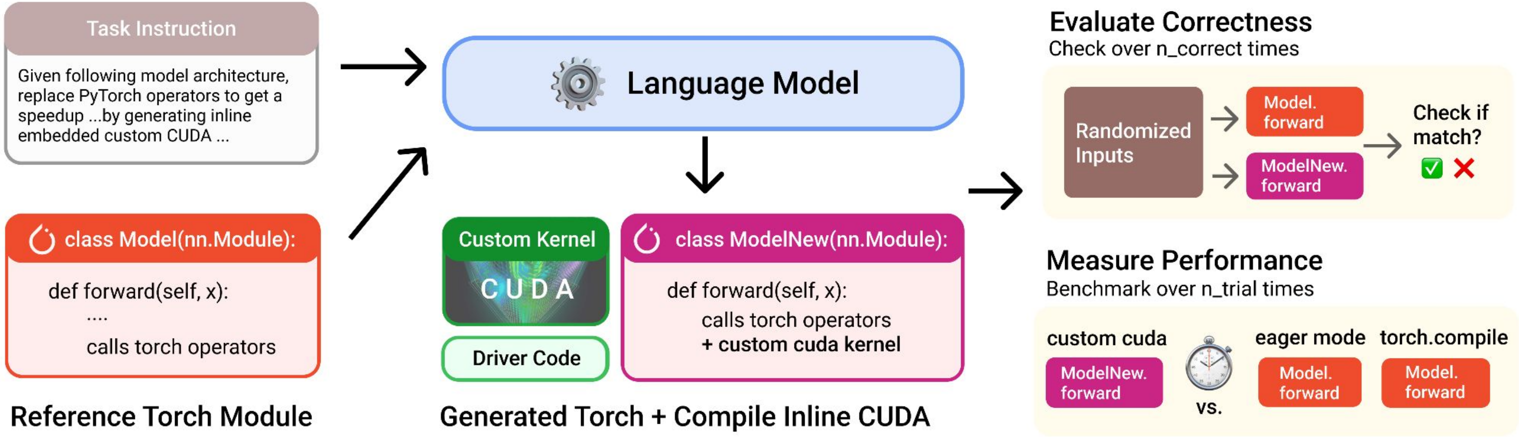


Chang, K., Wang, Y., Ren, H., Wang, M., Liang, S., Han, Y., ... & Li, X. (2023). Chipgpt: How far are we from natural language hardware design. arXiv preprint arXiv:2305.14019.

Liu, M., Pinckney, N., Khailany, B., & Ren, H. (2023, October). Verilogeval: Evaluating large language models for verilog code generation. In 2023 IEEE/ACM International Conference on Computer Aided Design (ICCAD) (pp. 1-8). IEEE.

Code Language Models for HARDWARE

➢ LMs can generate performant GPU Kernels, mimicking the AI engineer's workflow.



Ouyang, A., Guo, S., Arora, S., Zhang, A. L., Hu, W., Ré, C., & Mirhoseini, A. (2025). Kernelbench: Can llms write efficient gpu kernels?. arXiv preprint arXiv:2502.10517.

Code Language Models for HARDWARE

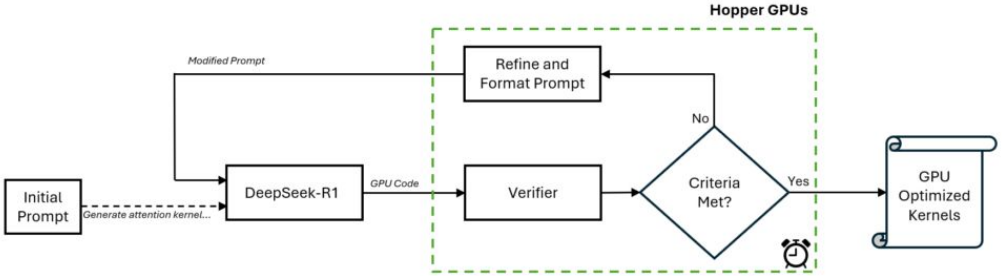
- KernelBench is a collection of 250 PyTorch neural network operations that researchers think systems should be able to automatically write optimized kernels for.

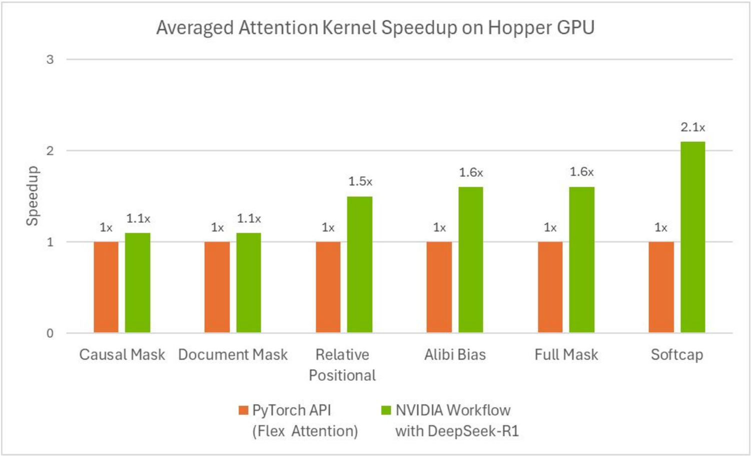
Level	# problems	Description	Realism	# Kernels per problem	Expert Time Estimate
1	90	Single PyTorch operations, eg CrossEntropyLoss	Realistic, memorizable	1	15 min - 4 hours
2	80	Sequences of 3-6 PyTorch operations, eg Linear->MaxPool3d->ReLU	Unrealistic, novel	1-3	30 min - 10 hours
3	37	Whole architectures from 2010s, eg AlexNet, GRU	Realistic, memorizable	10+	8-100+ hours
5	14	Frontier of open source capabilities and complexity in 2024	Realistic	10+	40-500+ hours

Ouyang, A., Guo, S., Arora, S., Zhang, A. L., Hu, W., Ré, C., & Mirhoseini, A. (2025). Kernelbench: Can llms write efficient gpu kernels?. arXiv preprint arXiv:2502.10517.

Code Language Models for HARDWARE

➢ NVIDIA engineers created a new workflow that includes a special verifier along with the DeepSeek-R1 model during inference in a closed-loop fashion for a predetermined duration.



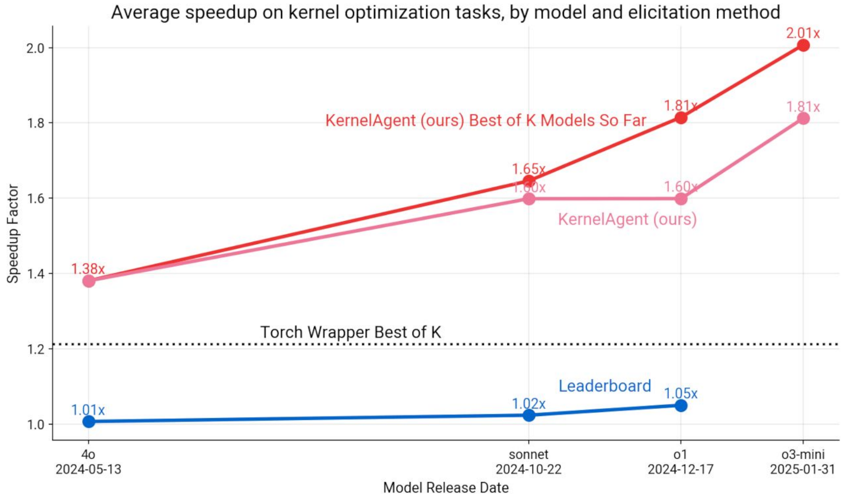


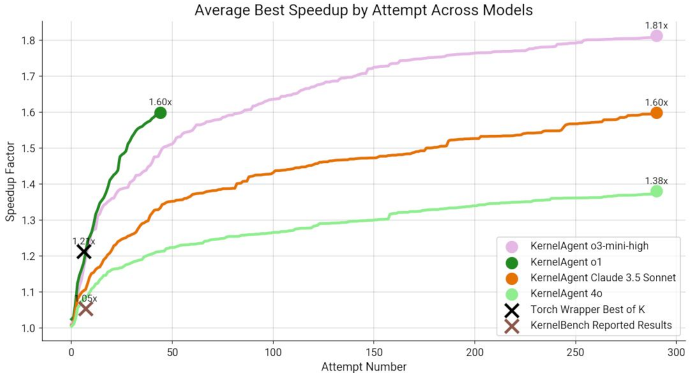
https://developer.nvidia.com/blog/automating-gpu-kernel-generation-with-deepseek-r1-and-inference-time-scaling/

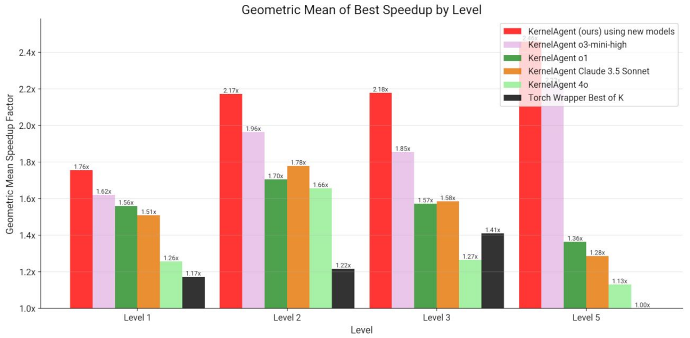
Code Language Models for HARDWARE

➢ METR researchers created "KernelAgent" to solve KernelBench tasks, achieving a speedup of 1.81x. Model written kernels could fill the underserved niche of accelerating machine learning projects that

dollars of compute.







https://metr.org/blog/2025-02-14-measuring-automated-kernel-engineering/

Code Language Models for Cybersecurity

"Fun" with computer security! For learning and hobbyists:

Forensics: Find a secret message in a filesystem

Tools: filesystem and network tools, grep, xd, etc.

Cryptography: Decrypt a message

Tools: SageMath, etc.

Binary exploitation (pwn): Exploit memory vulnerabilities

Tools: Debuggers, etc.

Reverse engineering: Compiling and disassembling binaries, identifying vulnerabilities

Web: Injection attacks, cross-site scripting attacks

Code Language Models for Cybersecurity: CTF

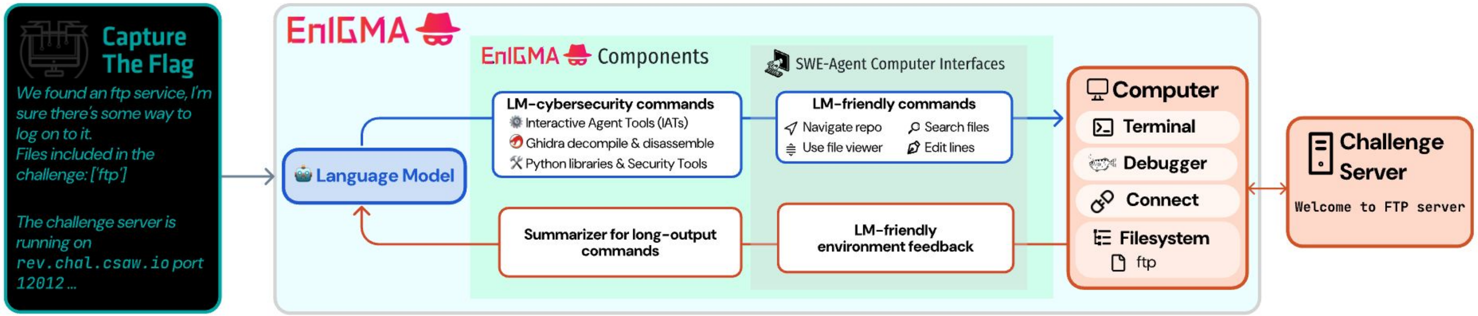
- InterCode-CTF (2023)
  - 100 challenges from PicoCTF (high-school level)
- NYU CTF Bench (2024)
  - 200 challenges from CSAW CTF (university level)
- Cybench (2025)
  - 40 challenges from various CTF competitions (professional level)

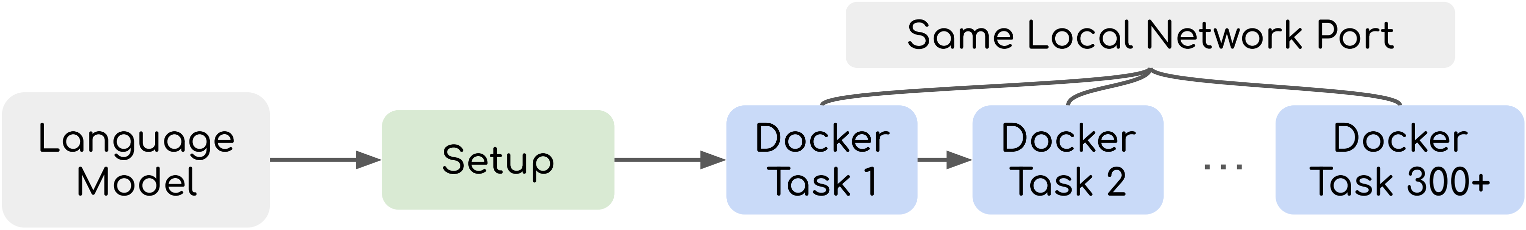
Easy

Hard

Code Language Models for Cybersecurity: Scaffolding

➢ EnIGMA interacts with the computer through an environment that is built on top of SWE agent and extends it to cybersecurity.

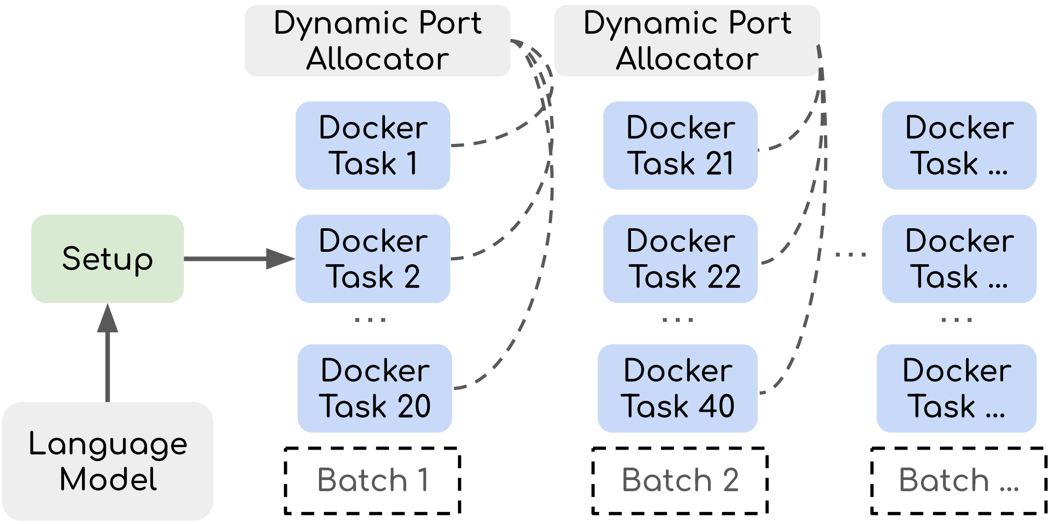




Abramovich, T., Udeshi, M., Shao, M., Lieret, K., Xi, H., Milner, K., ... & Press, O. EnIGMA: Interactive Tools Substantially Assist LM Agents in Finding Security Vulnerabilities. In Forty-second International Conference on Machine Learning.

Code Language Models for Cybersecurity: Scaffolding

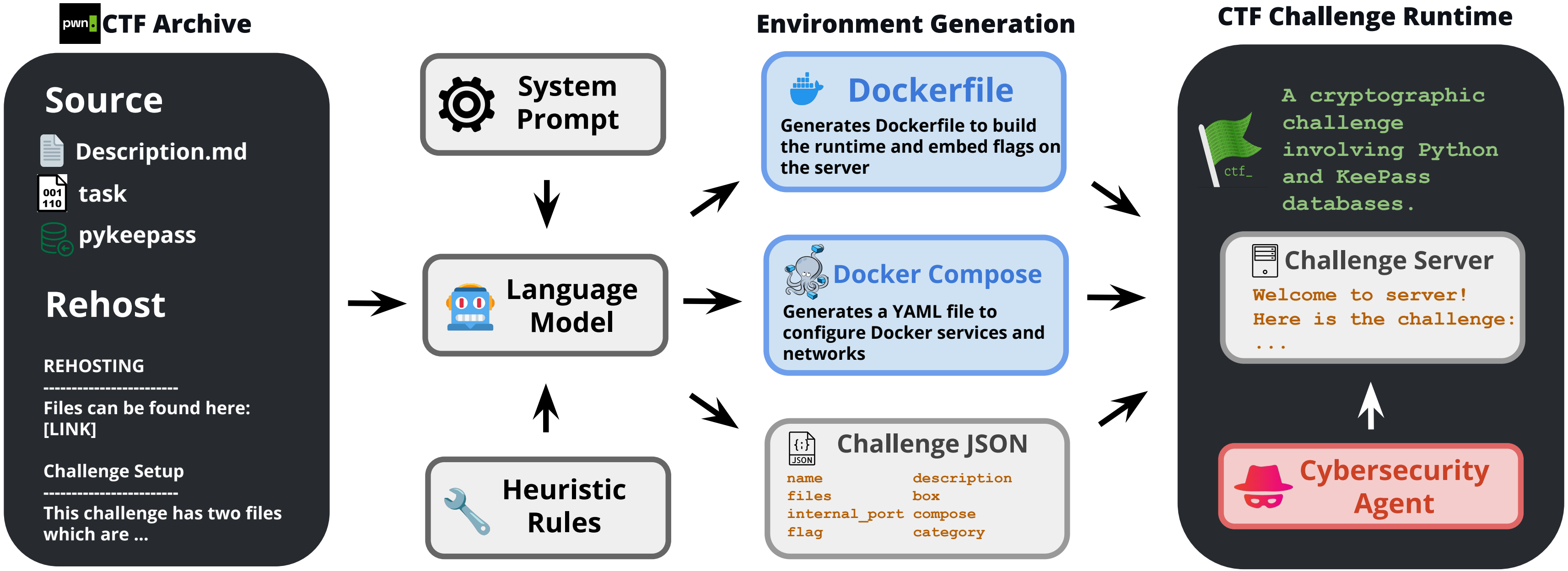
➢ EnIGMA+ reduces evaluation time from <u>days</u> to <u>hours</u> with dynamic port allocation.



Zhuo, T. Y., Wang, D., Ding, H., Kumar, V., & Wang, Z. (2025). Cyber-Zero: training cybersecurity agents without runtime. arXiv preprint arXiv:2508.00910.

Code Language Models for Cybersecurity: CTF-Forge

- CTF-Forge can automatically build 600+ CTF environments in 2 mins instead of weeks of expert configuration.

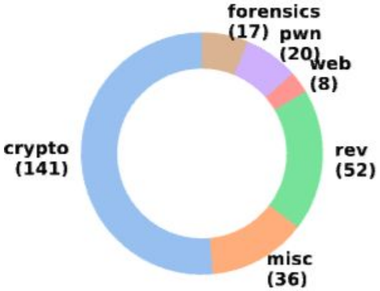


Zhuo, T. Y., Wang, D., Ding, H., Kumar, V., & Wang, Z. (2025). Training Language Model Agents to Find Vulnerabilities with CTF-Dojo. arXiv preprint arXiv:2508.18370.

Code Language Models for Cybersecurity: CTF-Dojo

➢ CTF-Dojo is the first collection of runtime environments to train cybersecurity agents.

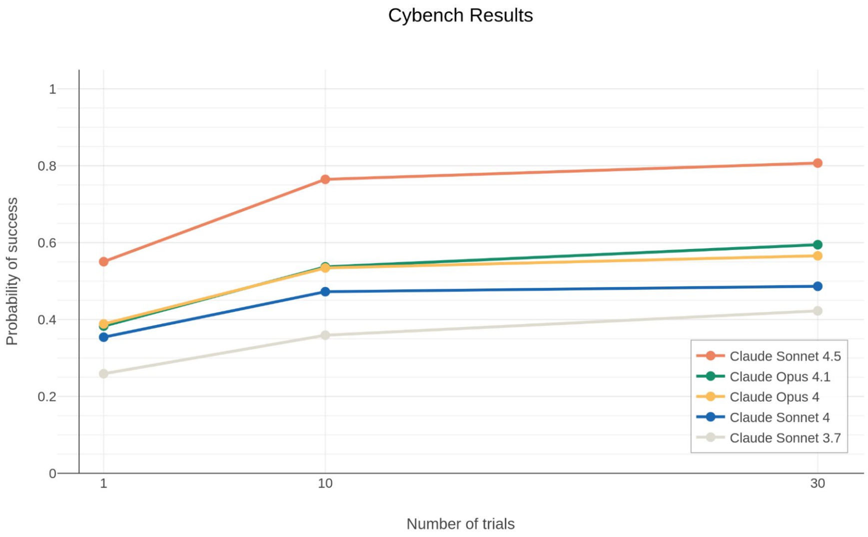
Benchmark	Level	# Competition	# Crypto	# Forensics	# Pwn	# Rev	# Web	# Misc	# Total
Training									
CTF-DOJO	Multi-Level	50	228	38	163	123	21	85	658
Evaluation									
InterCode-CTF	High School	1	16	13	2	27	2	31	91
NYU CTF Bench	University	1	53	15	38	51	19	24	192
Cybench	Professional	4	16	4	2	6	8	4	40



Zhuo, T. Y., Wang, D., Ding, H., Kumar, V., & Wang, Z. (2025). Training Language Model Agents to Find Vulnerabilities with CTF-Dojo. arXiv preprint arXiv:2508.18370.

Code Language Models for Cybersecurity: Road Ahead

➢ Language model agents get better on CTF.

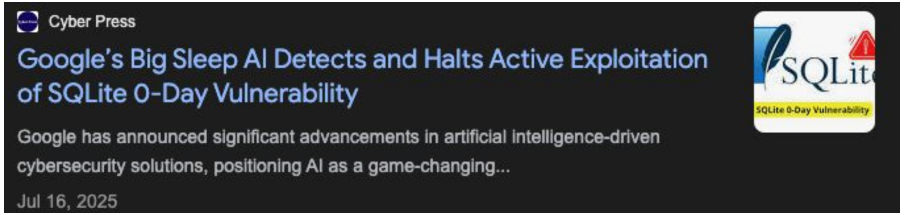


https://red.anthropic.com/2025/ai-for-cyber-defenders/

Code Language Models for Cybersecurity: Road Ahead

➢ Language model agents get better and better on CTF.

➢ Researchers recently have been applying language model agents to find vulnerabilities in the real-world software, but with limited scale.


Cyber Press

Google's Big Sleep AI Detects and Halts Active Exploitation of SQLite 0-Day Vulnerability

Google has announced significant advancements in artificial intelligence-driven cybersecurity solutions, positioning AI as a game-changing...

Jul 16, 2025

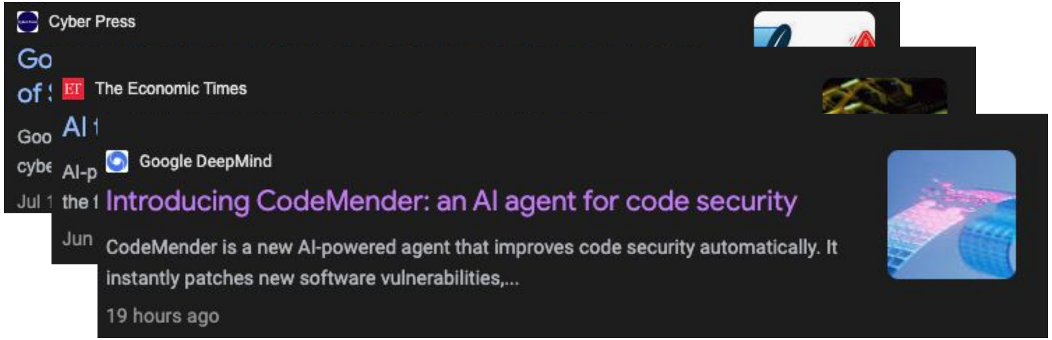
Code Language Models for Cybersecurity: Road Ahead

- ➢ Language model agents get better and better on CTF.
- ➢ Researchers recently have been applying language model agents to find vulnerabilities in the real-world software, but with limited scale.

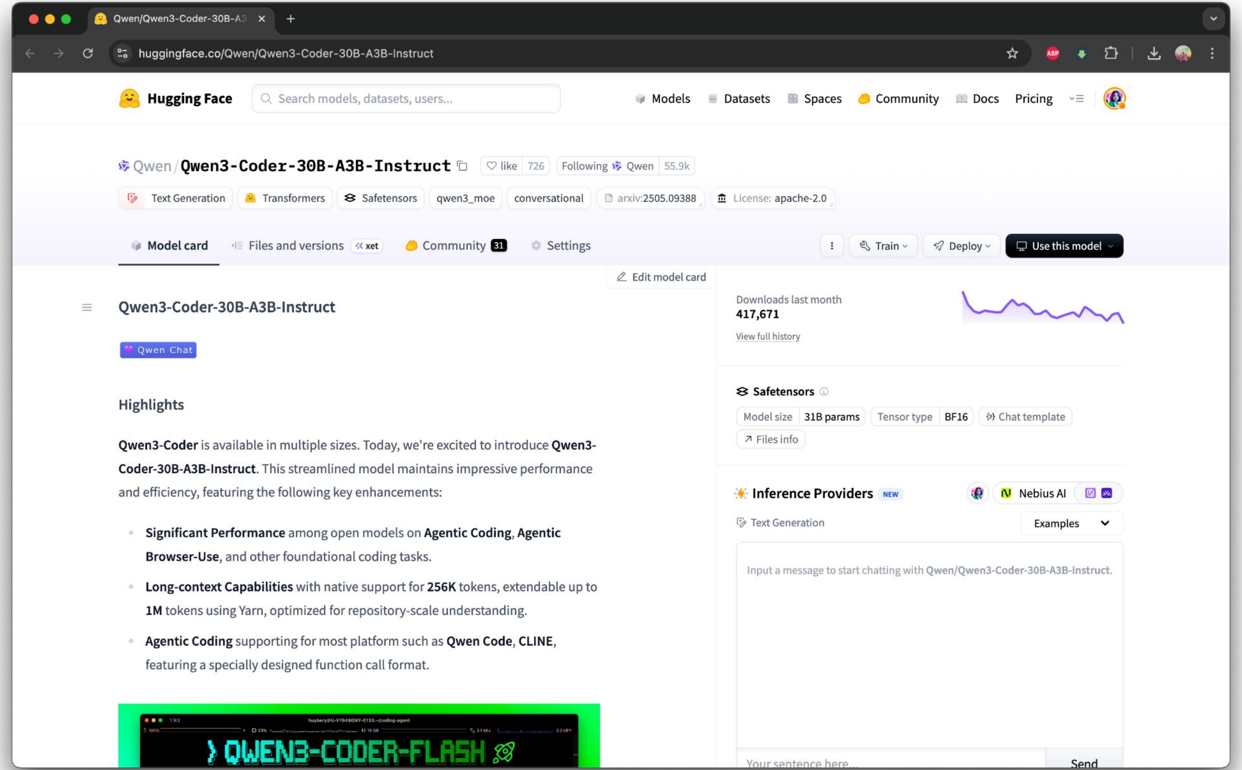
Code Language Models for Cybersecurity: Road Ahead

➢ Language model agents get better and better on CTF.

➢ Researchers recently have been applying language model agents to find vulnerabilities in the real-world software, but with limited scale.

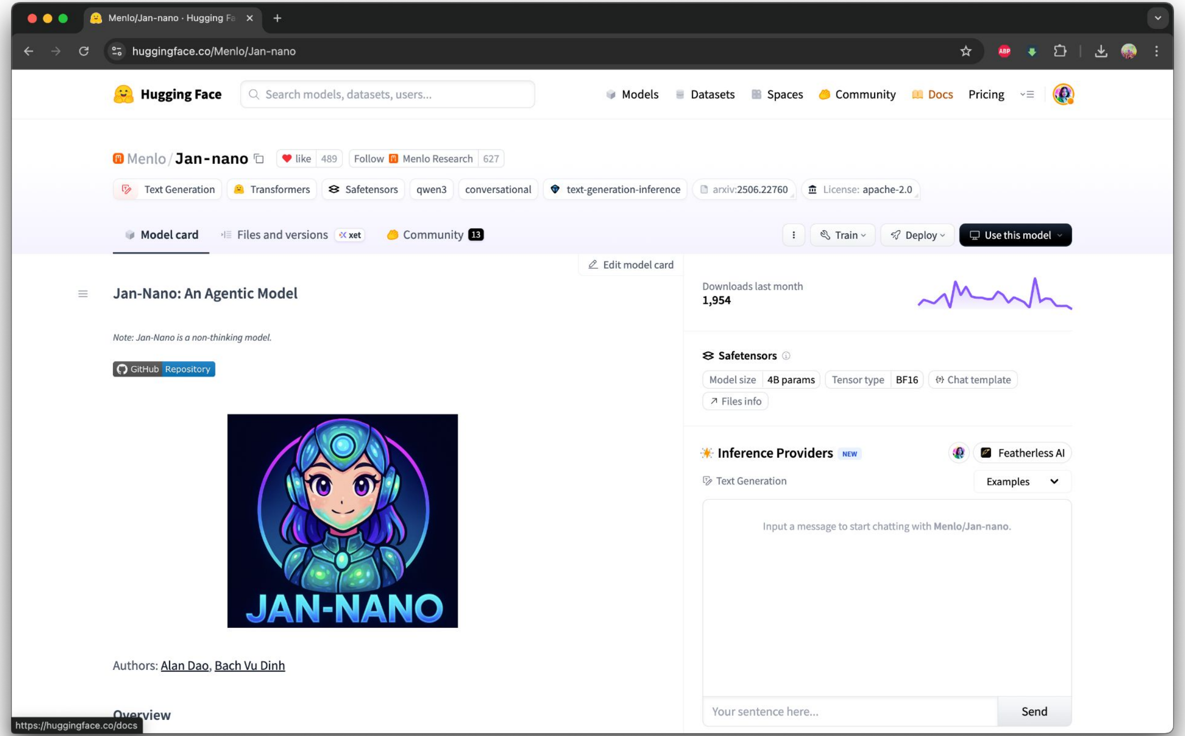


Small 🥺 Agentic Models for Code



https://huggingface.co/Qwen/Qwen3-Coder-30B-A3B-Instruct

Small 🥺 Agentic Models for Code



https://huggingface.co/Menlo/Jan-nano