ResearchCodeBench: Benchmarking LLMs on Implementing Novel ML Research Code

Tianyu Hua, Harper Hua, Violet Xiang, Benjamin Klieger, Sang T. Truong, Weixin Liang, Fan-Yun Sun, Nick Haber

LLMs can write code… but can they implement **novel ML research ideas**?

Why Benchmark Research Code Generation?

Traditional Code Benchmarks

- Test general coding or bug fixing
- Focus on implementing / reproducing existing code
- Fail to test understanding of new scientific ideas

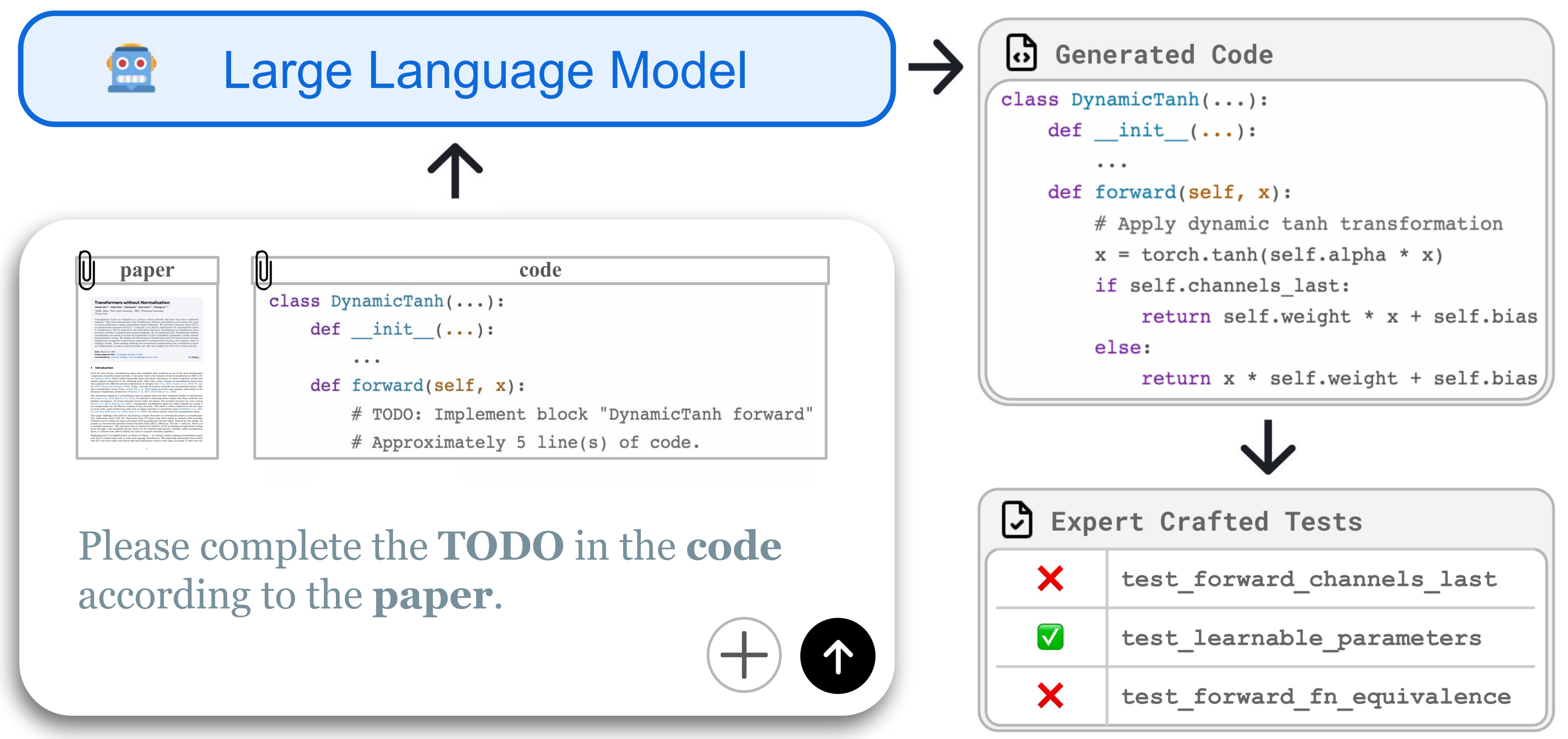
Research Code Generation

- Must implement **novel** ideas
- Often lies outside training distribution
- Requires understanding scientific intent

We need better evaluations for code-gen systems meant to assist scientific discovery.

ResearchCodeBench

`212 challenges from 20 recent ML papers (2024–25) with executable tests`



```
class DynamicTanh(...):
    def __init__(...):
        ...
    def forward(self, x):
        # TODO: Implement block "DynamicTanh forward"
        # Approximately 5 line(s) of code.
```

Please complete the **TODO** in the **code** according to the **paper**.

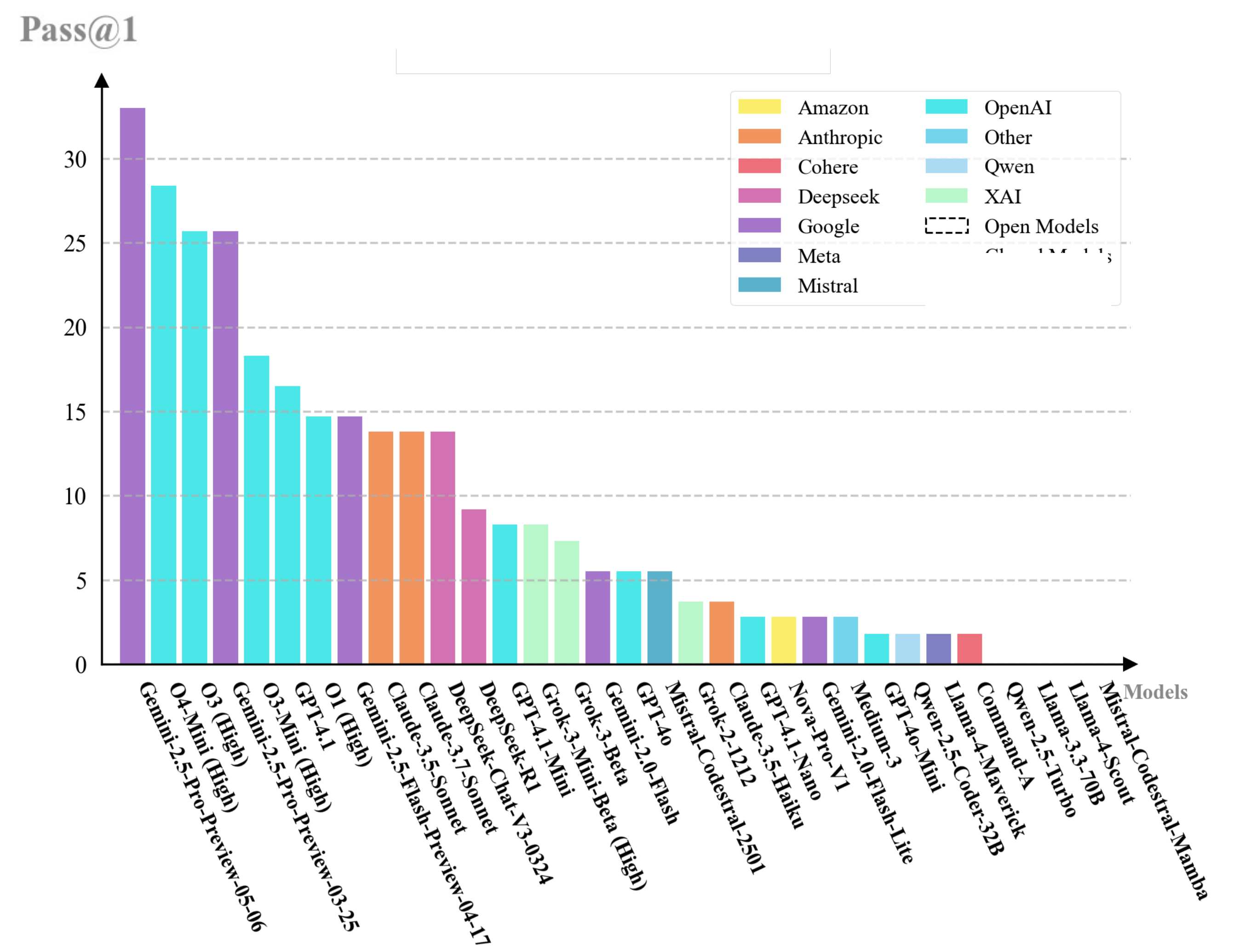
```
class DynamicTanh(...):
    def __init__(...):
        ...
    def forward(self, x):
        # Apply dynamic tanh transformation
        x = torch.tanh(self.alpha * x)
        if self.channels_last:
            return self.weight * x + self.bias
        else:
            return x * self.weight + self.bias
```

✗	test_forward_channels_last
✓	test_learnable_parameters
✗	test_forward_fn_equivalence

How it works?

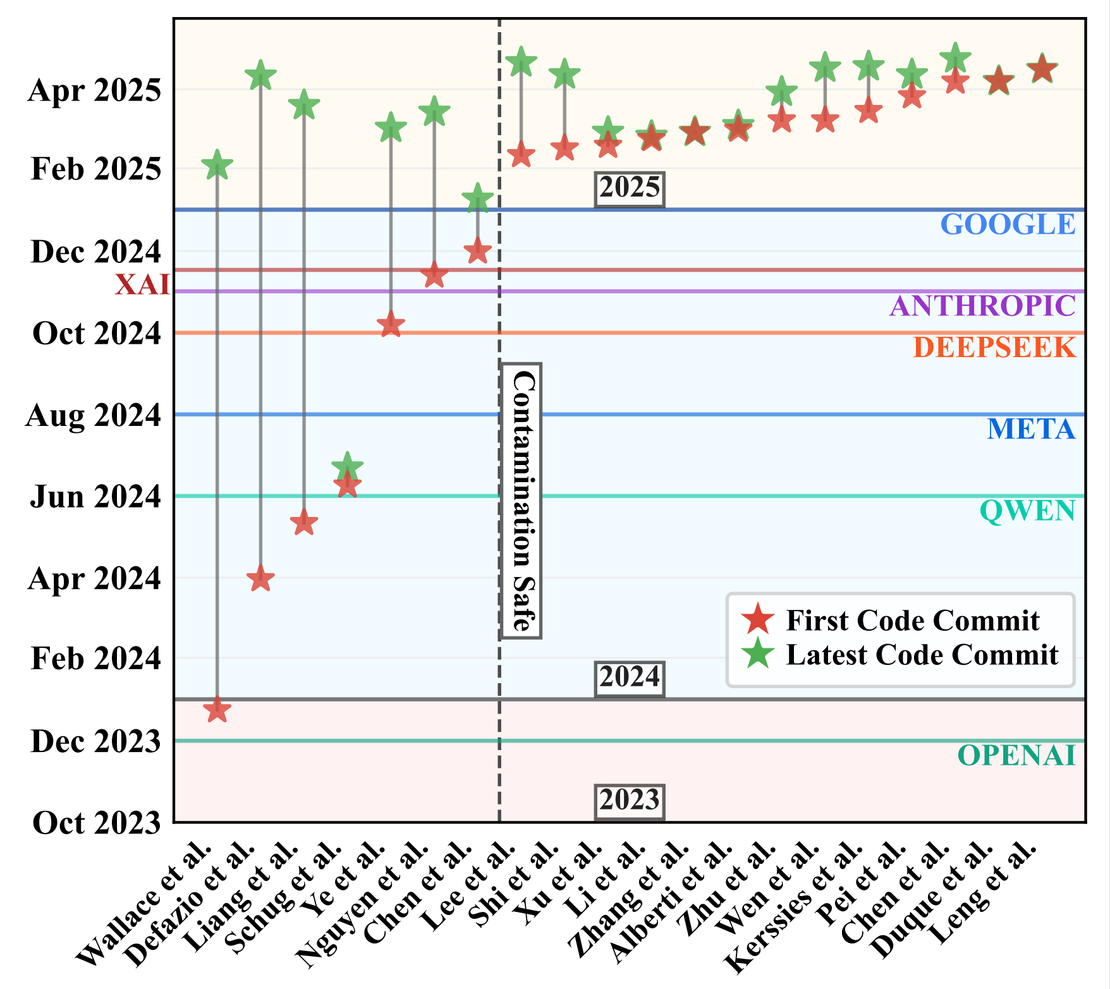
- LLM receives a single prompt containing the research paper and masked context code
- Model fills in the core missing contribution
- Completion is validated via expert-crafted tests
- Tests use the original paper implementation as reference

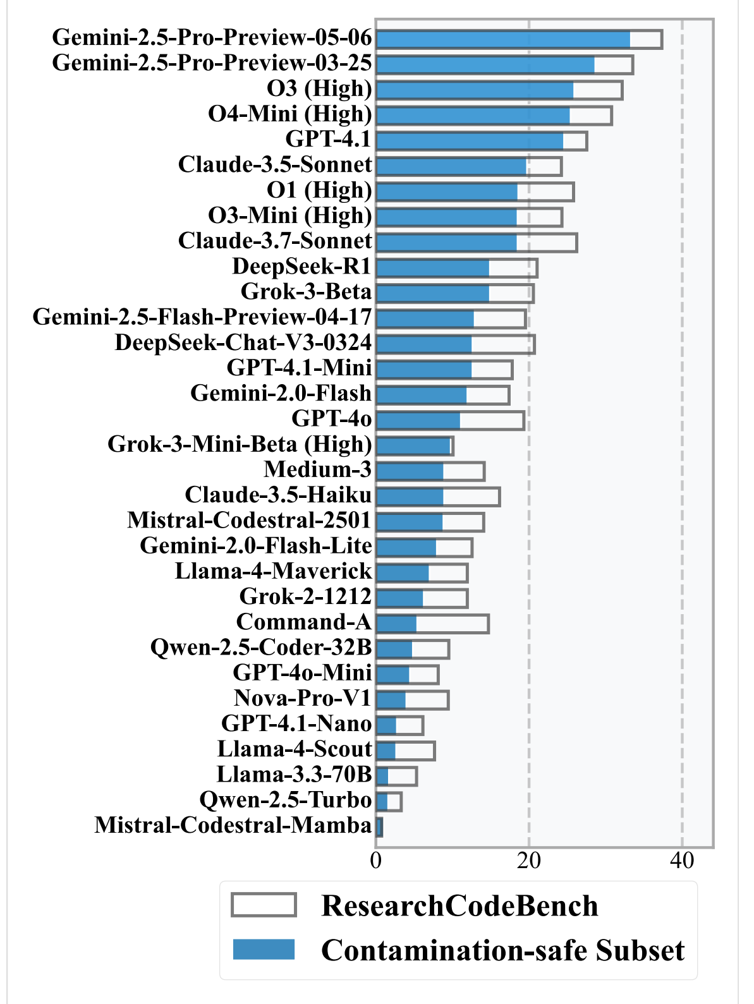
ResearchCodeBench-HARD: Deep Algorithmic Reasoning Remains Unsolved!



- Best model achieves only **33% Pass@1**
- Many tasks remain unsolved across all model families
- HARD tasks stress deep algorithmic reasoning

Newer Papers Are Harder for LLMs!



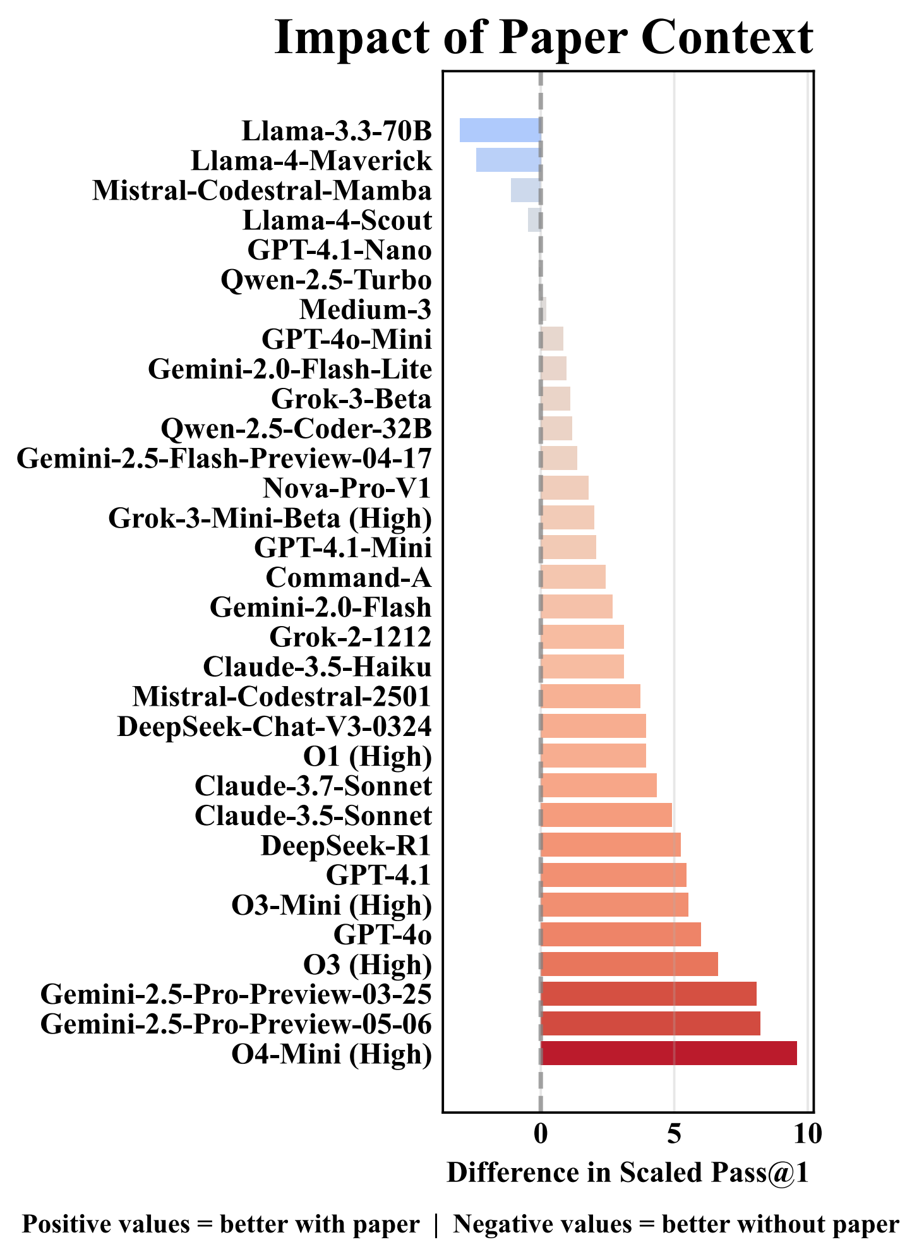


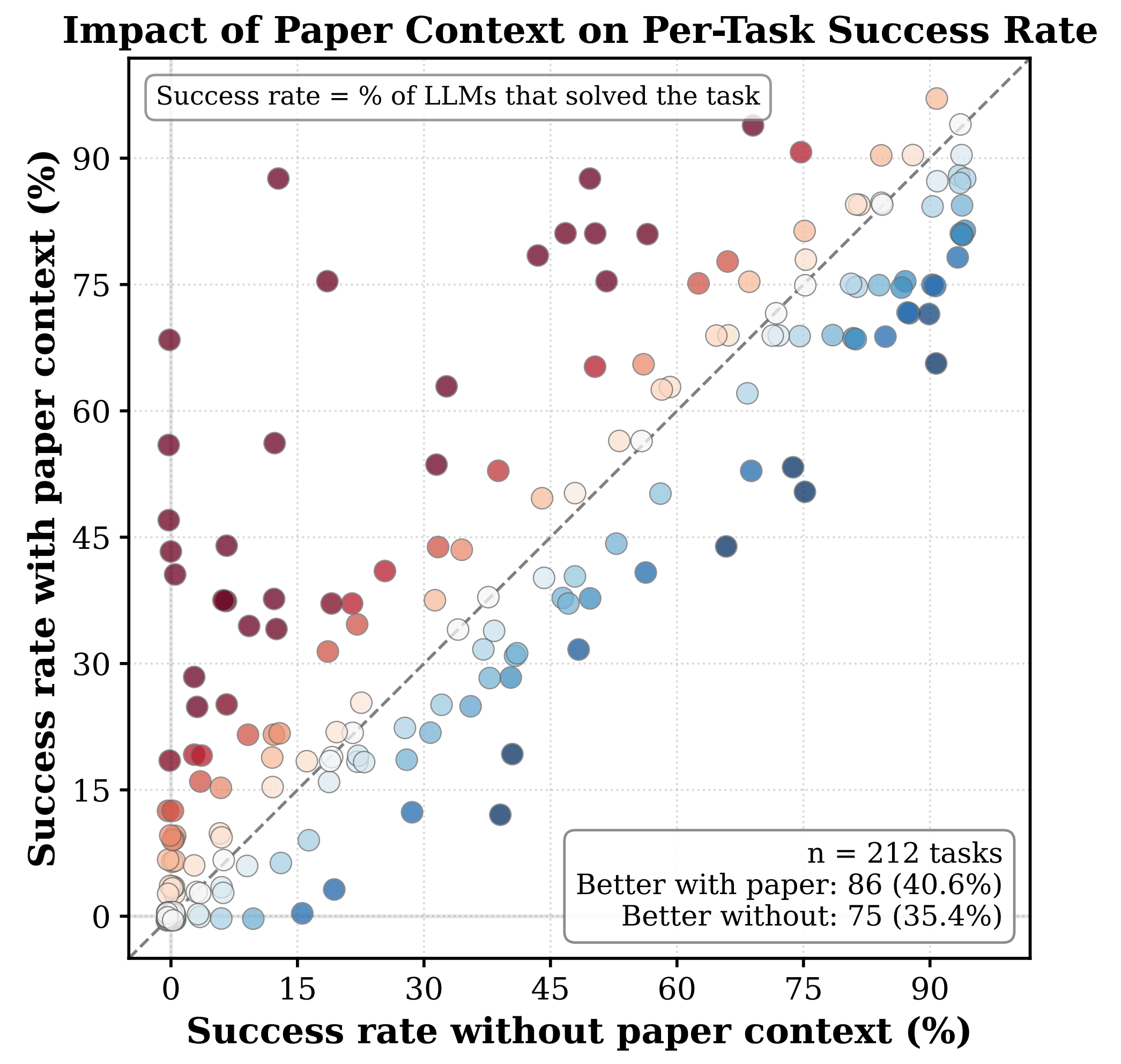
Research question: *How does knowledge cutoff affect performance?*

Findings:

- **13 of 20** papers were published *after* the knowledge cutoff of all models.
- Performance drops sharply on these **post-cutoff papers**, indicating models cannot rely on memorized code.
- Suggests performance depends strongly on whether the paper predates model training.

Stronger Models Make Better Use of the Paper!

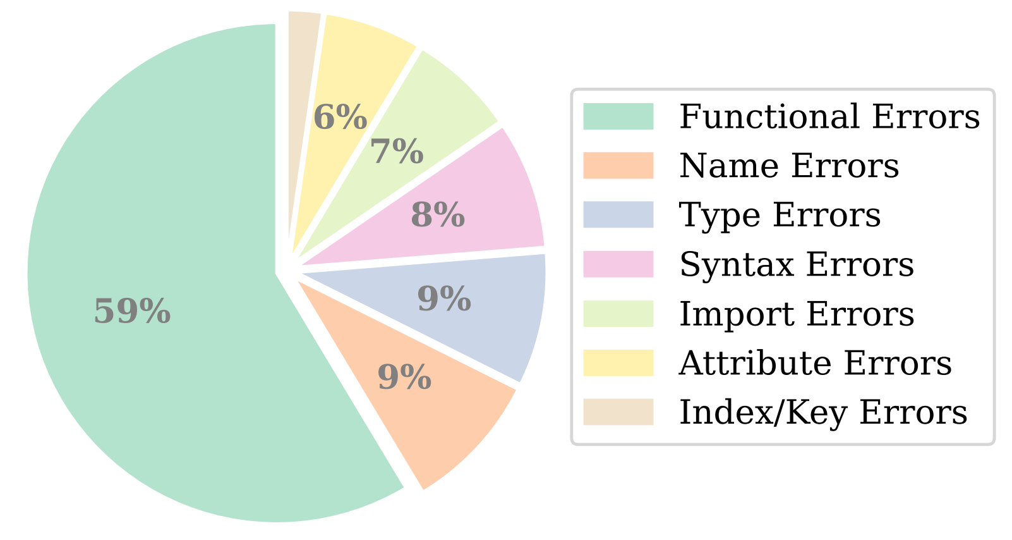




Ablation: Compare performance w vs. wo providing the research paper excerpt.

Finding: Larger and more capable models gain more from the paper context, showing they rely on scientific descriptions rather than memorized patterns.

Error Analysis



- Models correctly handle basic Python syntax
- Primary challenge: aligning code with intended algorithms
- Future directions: advancing LLM scientific reasoning and algorithmic understanding

Future Work: Toward Live, Automated Research Benchmarks

- Automating item creation from new papers could turn ResearchCodeBench into a continuously updating “livebench.”
- This would evaluate LLMs on research ideas as they emerge, not months or years later.
- Enables scalable testing of scientific reasoning, not memorization.