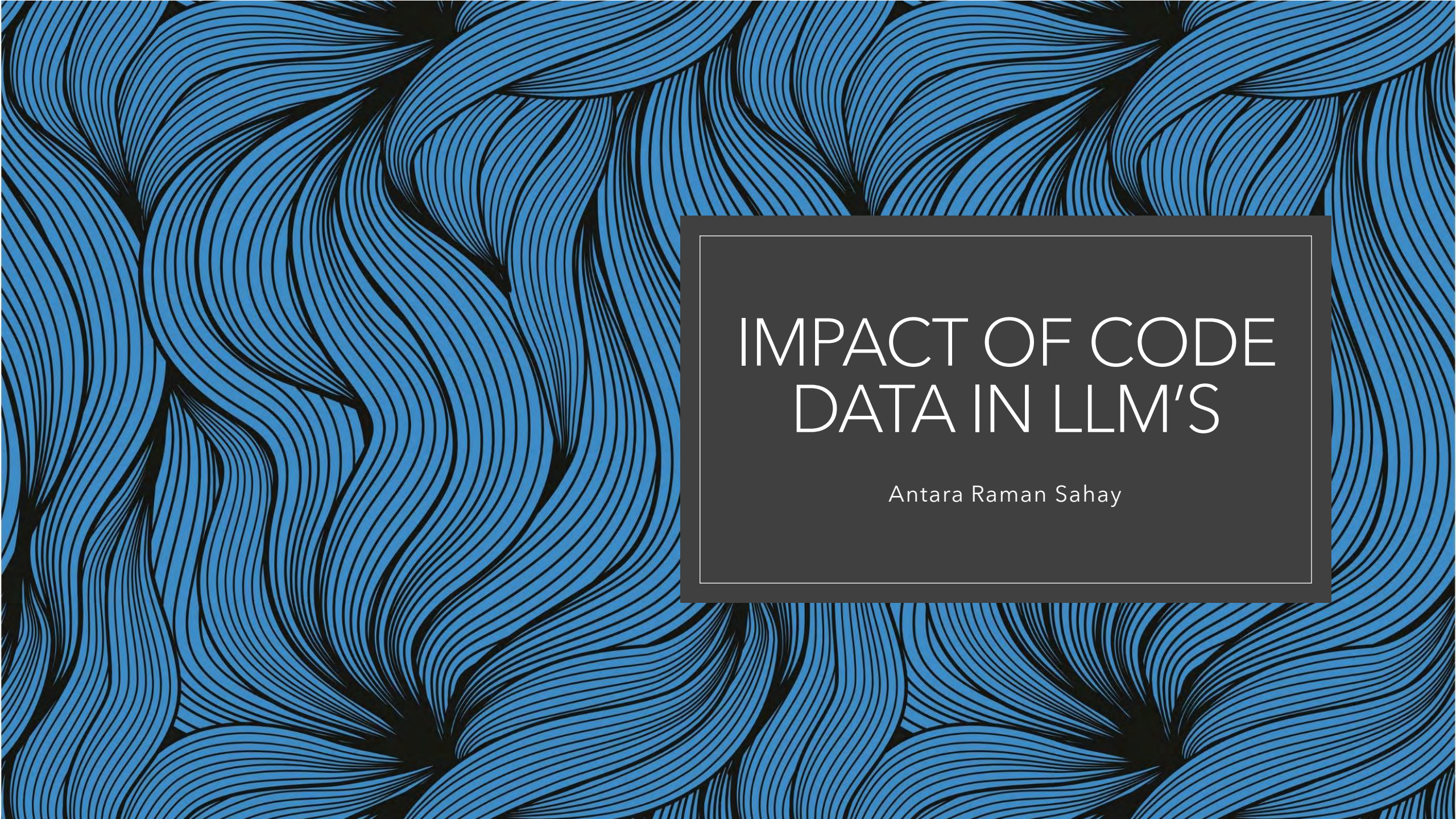
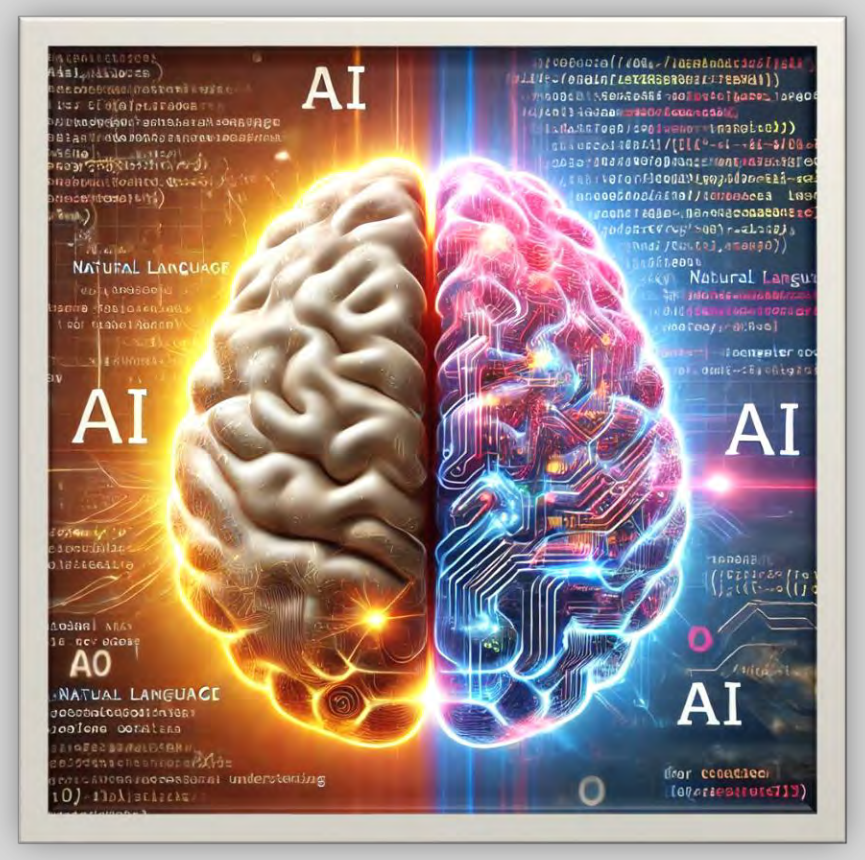


IMPACT OF CODE DATA IN LLM'S

Antara Raman Sahay

To Code or Not To Code!?

Text Data



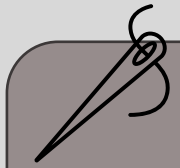
Code Data

Research Objectives

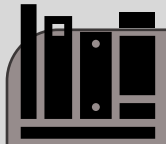
- Assess the effect of code data across three task categories:
  - Natural Language Reasoning
  - World Knowledge
  - Code Performance
- Test code-heavy, balanced, and text-only models.

Phases of training LLM

Pre-Training



Fine-Tuning



Cont.
Pre-Training

Cooldown

Evaluation
& Fine-tuning

Experimental setup

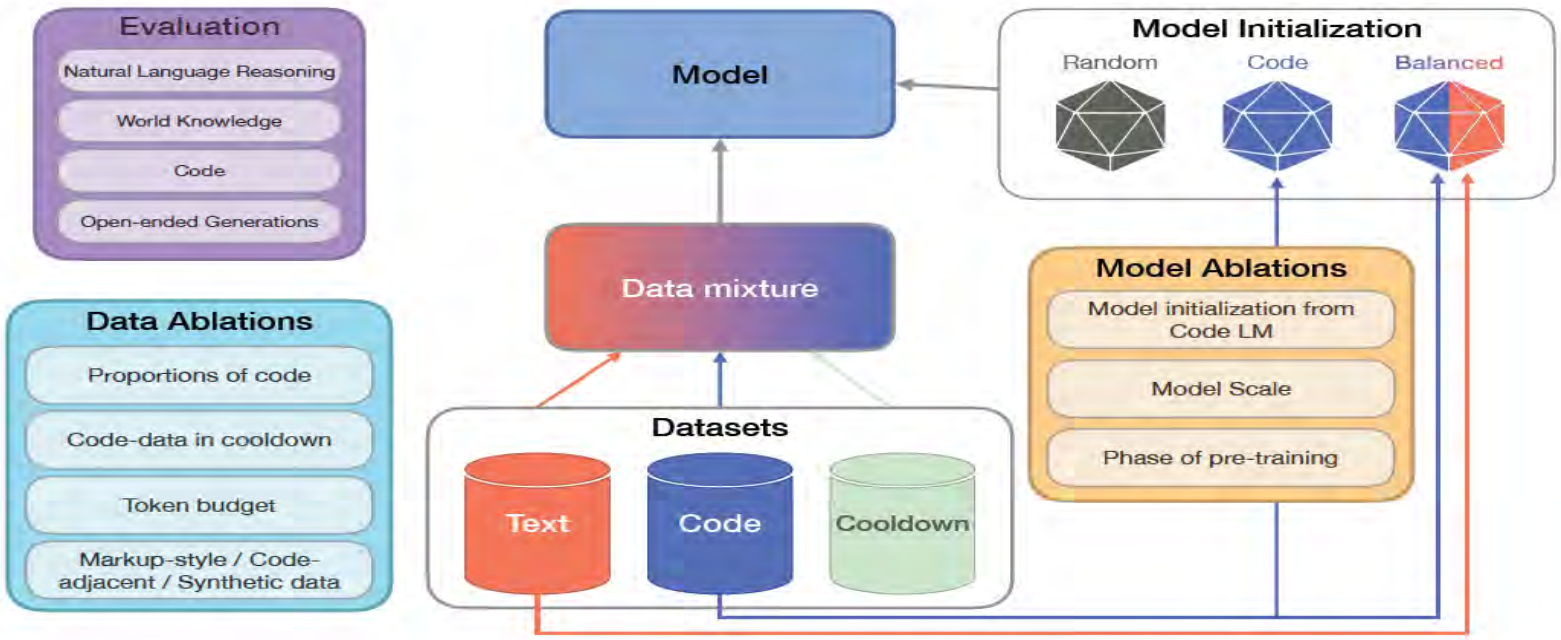
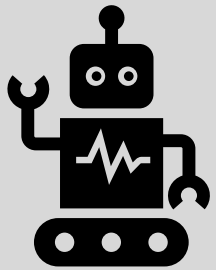
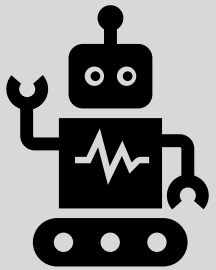


Figure 1: **Overview of our experimental framework**: We exhaustively evaluate the impact of code by varying: 1) the proportion of code in pre-training, 2) code quality and properties, 3) model initialization, 4) model scale, and 5) stage of training at which code is introduced. We evaluate the resulting model on a wide-ranging set of tasks, including natural language reasoning, world knowledge, code, and open-ended generations.

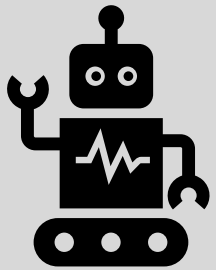
IMPACT OF INITIALIZATION USING CODE PRE-TRAINED MODELS



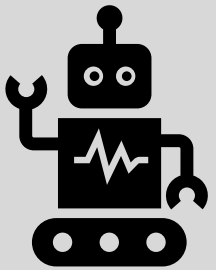
Text-LM



Balanced-LM



Balanced > Text



Code > Text

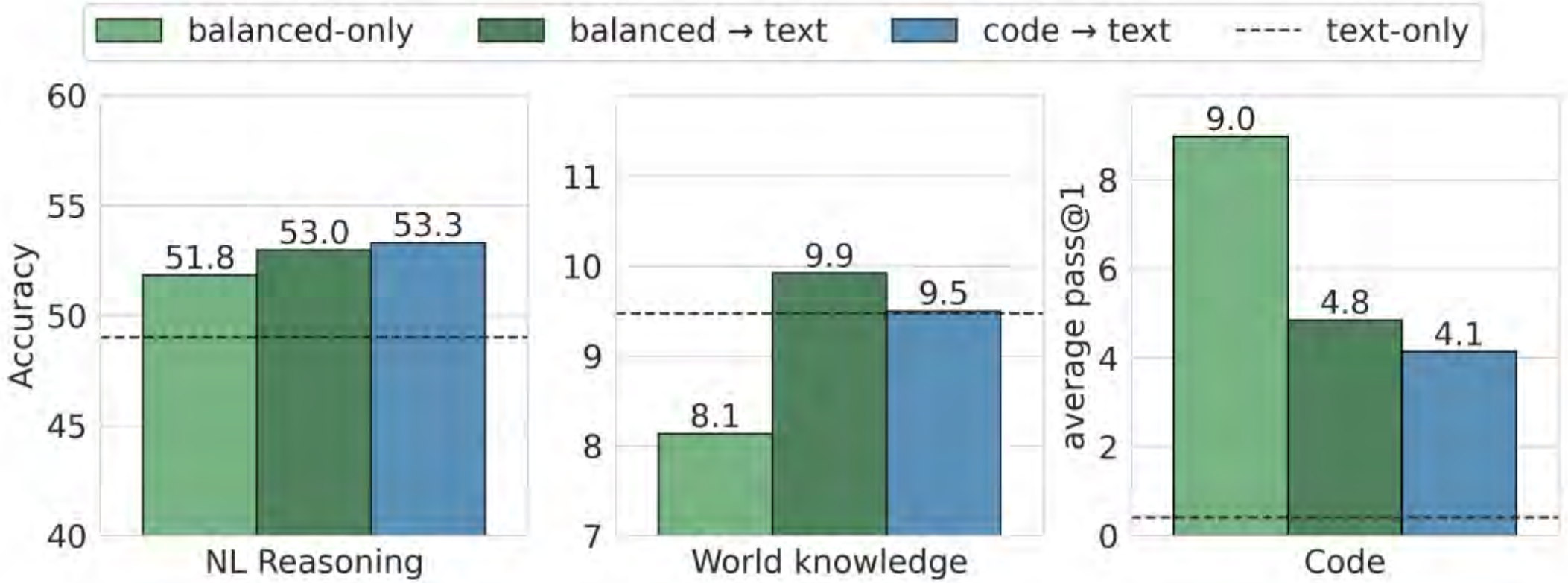
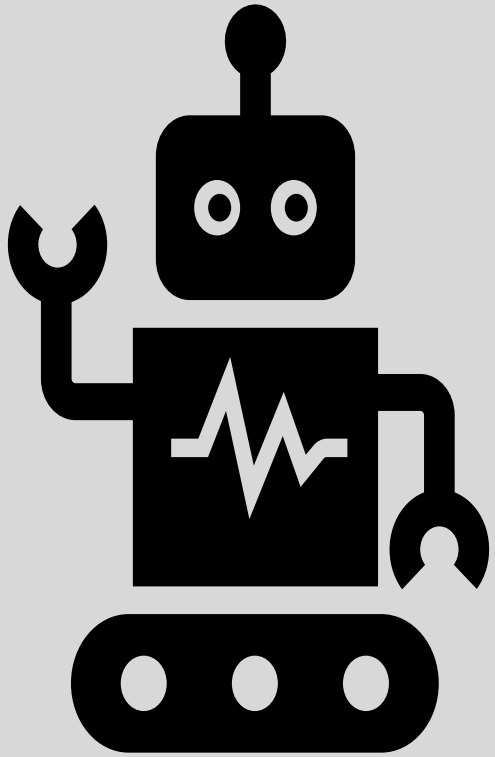


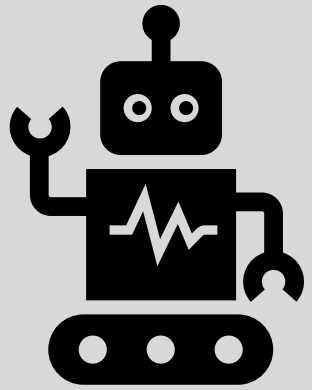
Figure 2: **Impact of initialization using code pre-trained models**: Initializing model training with code pre-trained models improves reasoning and code generation compared to text-only models, where the improvement is the most when continued pre-training with high percentage text (Balanced→Text, Code→Text). Note that these variants are designed to isolate the role of initialization, so do not include cooldown.

IMPACT OF SCALE



2.8B

>



470M

IMPACT OF CODE DATA PROPORTION

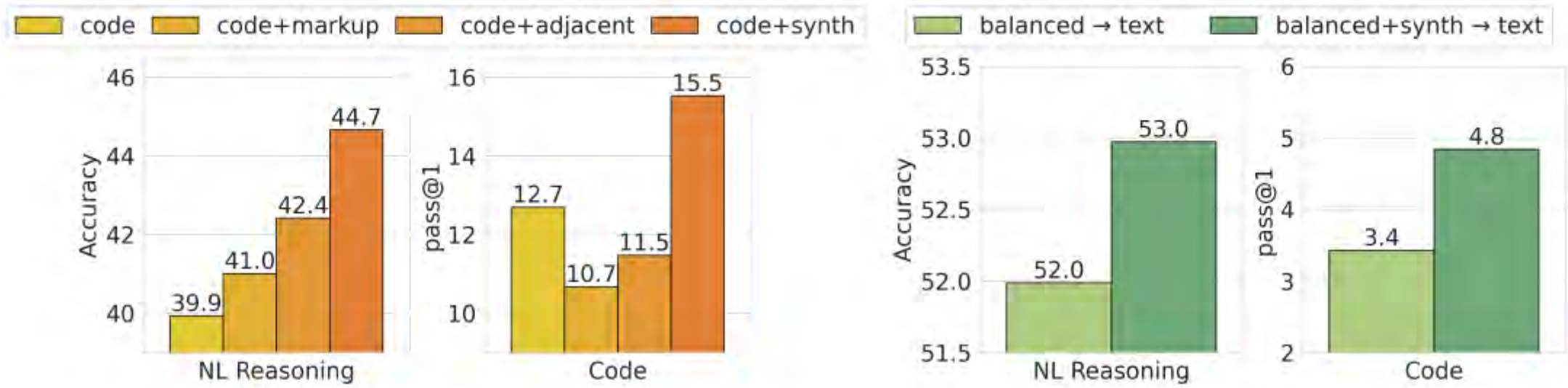
Section Findings

- The optimal share of code to optimize for the highest average performance across World Knowledge and NL Reasoning benchmarks is 25% code. Average performance starts to decay at 75% code with notable drops in World knowledge of up to 86.1% at the highest ratios of code.
- Not including any code hurts NL reasoning performance, degrading by 3.4% relative to pre-training with 25% code.
- Code performance benchmarks improve almost linearly with an increasing proportion of code data. Increasing code from 25% to 100% during pre-training boost code performance by 2.6x.

IMPACT OF CODE QUALITY

4 Types of CODE Model

- Web-based Code (Baseline): Natural code from online repositories.
- Synthetic Code: High-quality, machine-generated code.
- Code-Adjacent: GitHub issues, Jupyter Notebooks, Stack Exchange.
- Markup Languages: HTML, CSS – not code, but relate



(a) Code-only Pre-training

(b) Continual Pre-training

Figure 5: **Impact of using different properties of code data**: **(a)** As the most impactful code data source, synthetically generated high-quality code improves NL reasoning and code performance for code pre-training. **(b)** These improvements with synthetically generated high-quality code data also transfer the continual pre-training setting.

IMPACT OF CODE IN COOLDOWN

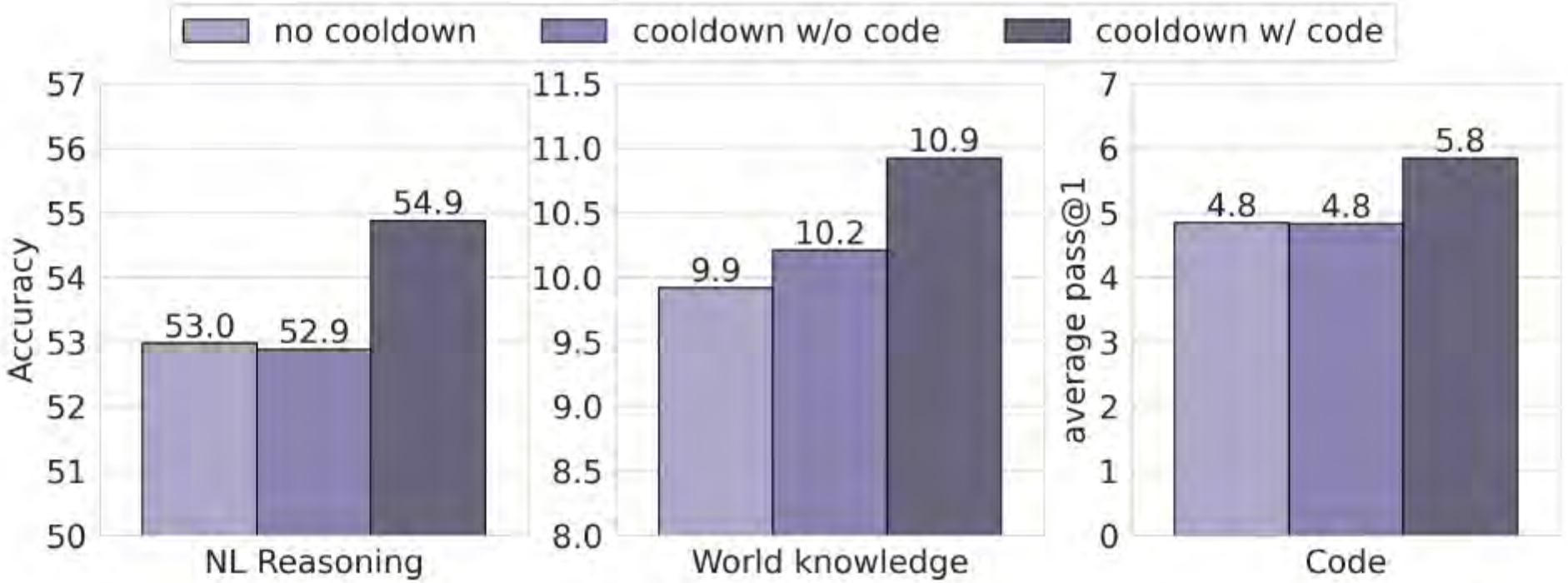


Figure 6: **Impact of code data in pre-training cooldown**: Including code data in the cooldown phase improves downstream relative to cooldown with no code. Both cooldown variants improve upon no cooldown across all tasks.

KEY FINDINGS

Results – Natural Language Reasoning

- Adding 25% code data boosted NL reasoning by 8.2%.
- Cooldown with code further improved reasoning performance by 3.6%.
- Text-only models performed well, but balanced models were the sweet spot.

Results – World Knowledge

- Code in pre-training provided a 10.1% boost in world knowledge tasks.
- Markup and code-adjacent data also had a positive, but smaller effect.
- Cooldown with code was crucial for world knowledge tasks.

Results – Code Performance

- Code-heavy models outperformed text-heavy models by 12x in code tasks.
- Synthetic code data was particularly impactful, with a 44.9% boost.
- Balanced models gave a strong overall performance but lagged code-only models in coding tasks.

Best Recipe for Code Performance

- For code benchmarks, code-only models were the clear winners.
- Balanced→text models were strong performers in NL reasoning but lagged in code.
- Synthetic code data was a key differentiator in boosting code performance.

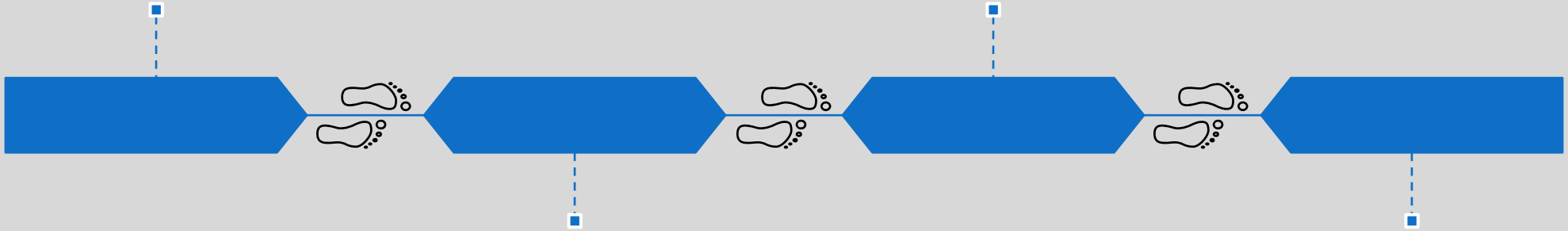
Key Recommendations for Pre-Training with Code

- Include a balanced mix of code and text data from the start.
- Use synthetic code data to improve both code and NL tasks.
- Prioritize the inclusion of code in the cooldown phase to maximize performance gains

Future Research Areas

Scale synthetic code generation

Task Specific Fine-Tuning



Explore Training models

Advanced cooldown

Final Takeaways

- Code data significantly improves AI models across all tasks, not just code-specific tasks.
- Balanced models with both text and code are best for general tasks, while code-heavy models dominate coding benchmarks.
- The cooldown phase, particularly with code, is critical for optimal model performance.

QUESTIONS ?

Thank-You!