Principled Understanding of Generalization for Generative Transformer Models in Arithmetic Reasoning Tasks

Xingcheng Xu[1] Zibo Zhao[2,3] Haipeng Zhang[2,*] Yanqing Yang[4,*]

[1]Shanghai AI Laboratory [2]ShanghaiTech University [3]University of Hong Kong [4]Fudan University

ACL 2025: The 63rd Annual Meeting of the Association for Computational Linguistics

The Generalization Puzzle

Transformer models show puzzling inconsistencies in arithmetic generalization. Their ability to generalize to longer, unseen inputs (length generalization) varies dramatically across seemingly similar tasks.

Table 1. Length Generalization Mysteries from Literature

PE Type	Addition	Multiplication	Modular Op. $p = 100$	Modular Op. $p = 101$
APE	✗	✗	✓	✗
RPE	✓	✗	✓	✗

Our Goal: Provide a unified theory to explain these phenomena.

A Unified Theoretical Framework

We propose that generalization emerges from the alignment between three factors:

- **Task Properties**: Intrinsic structure like symmetries (*e.g.*, translation invariance).
- **Model Architecture**: Inductive biases from components like Positional Encodings (PE).
- **Training Data Distribution**: The specific function the model is trained to approximate.

Insight 1: Addition & Translation Invariance

The digit-wise addition algorithm is **translation-invariant**—the computation is identical for every position.

- **Relative PE (RPE)** captures this repeating structure, enabling successful upward generalization.
- **Absolute PE (APE)** learns position-specific rules and cannot generalize. It learns a truncated function:

$$\hat{f}(a, b) = (a \pmod{10^n}) + (b \pmod{10^n})$$

This causes upward generalization failure.

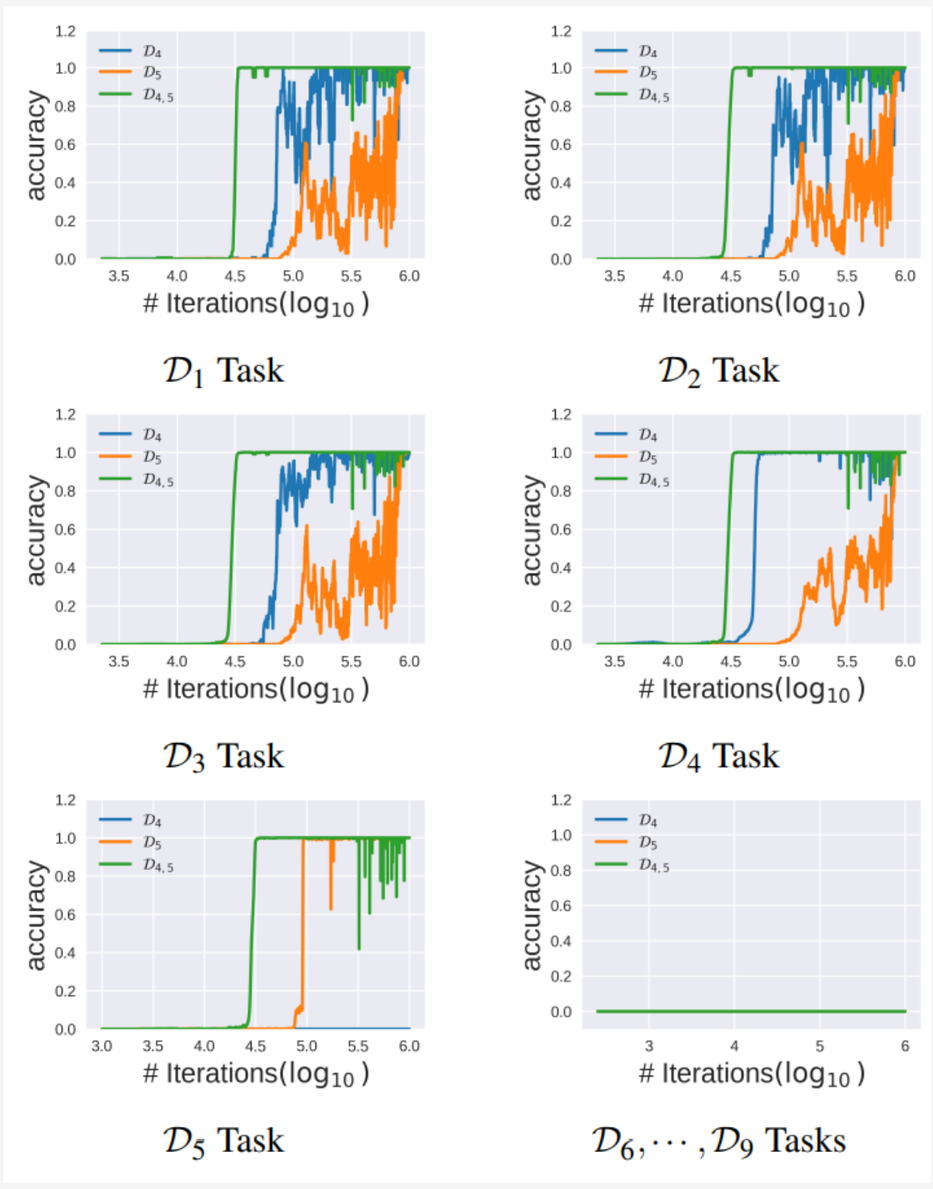

Figure 1. OOD Accuracy for Addition (APE). Models trained on $\mathcal{D}_4$ (red) or $\mathcal{D}_5$ (blue) fail on longer inputs.

Insight 2: Multiplication & Lack of Invariance

The multiplication algorithm is **not translation-invariant**. The calculation for output digit c_k involves a complex, non-local sum over many input digit pairs. This complex structure does not align with the inductive biases of either APE or RPE, leading to upward generalization failure.

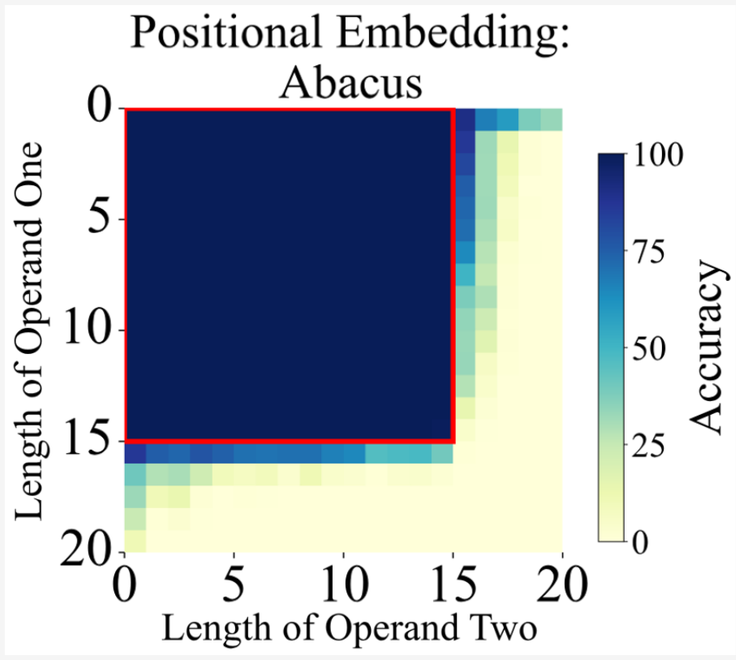

Figure 2. **Multiplication Failure (RPE)**. From McLeish et al. (2024), a model trained on up to 15 digits fails on longer inputs.

Insight 3: Modular Arithmetic & Base Alignment

The key is whether the modulus p aligns with the number base (10).

Case 1: p divides 10^k (e.g., $p = 100, 50$)

- **Property**: The result depends only on the last k digits.
- **Result**: The model learns to ignore higher-order digits, enabling perfect upward generalization.

Case 2: p does not divide 10^k (e.g., $p = 101, 51$)

- **Property**: All digits matter. Higher-order digits are crucial.
- **Result**: The APE-trained model learns a truncated function, causing upward generalization failure.

The "Smoking Gun": A Quantitative Prediction

For the hard case (modular addition, p does not divide 10^n), our framework yields a precise, falsifiable prediction for the upward OOD accuracy.

Theorem (Informal, Thm. 5)

For a model trained on n digits and tested on much longer inputs, the accuracy is approximately:

$$\text{Accuracy} \approx \frac{\gcd(p, 10^n)}{p}$$

Table 2. Experimental Test Accuracy (%) on $\widetilde{\mathcal{D}}_i$ vs. Theory

Modulus (p)	Experimental Test Accuracy (%) on $\widetilde{\mathcal{D}}_i$: $i = 4$ (ID)	$i = 5$	$i = 6$	$i = 7$	$i = 8$	$i = 9$	Theory $\gcd(p, 10^4)/p$
$p = 100$	100	100	100	100	100	100	100%
$p = 101$	100	0.0	1.2	0.9	1.1	1.0	**0.99%**
$p = 150$	100	33.2	33.6	32.3	33.0	33.7	**33.3%**
$p = 51$	99.3	0.3	1.8	1.9	1.9	1.6	**1.96%**

The experimental results show a stunning match with the theoretical predictions.

Conclusion

- We proposed a **unified theoretical framework** that resolves long-standing puzzles about arithmetic generalization in Transformers by aligning task structure, model biases, and data distribution.
- Our framework provides principled, quantitative, and experimentally validated explanations for OOD behavior.

Contact: Xingcheng Xu
Email: xingcheng.xu18@gmail.com

Paper: `https://arxiv.org/abs/2407.17963`
Code: `https://github.com/xingchengxu/ArithmeticLLM`