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 Lab [2]ShanghaiTech Univ. [3]Univ. of Hong Kong [4]Fudan Univ.

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

The Generalization Puzzle in Arithmetic Reasoning

- Transformer models excel at many tasks, but their generalization capabilities, even in simple arithmetic, are not fully understood.

- Empirical results from prior work reveal puzzling discrepancies.

- **Our Goal:** Provide a unified theoretical framework to explain these mysteries.

Table 1: Length Generalization Mysteries from Literature

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

Why do these inconsistencies exist?

This Paper: A Unified Theoretical Framework

- We argue that generalization behavior is determined by the interplay of three key factors:
  1. **Task Properties**: *e.g.*, translation invariance in addition.
  2. **Model Architecture**: *e.g.*, Absolute vs. Relative Positional Embeddings (APE vs. RPE).
  3. **Training Data Distribution**: What the model actually sees during training.

This Paper: A Unified Theoretical Framework

- We argue that generalization behavior is determined by the interplay of three key factors:
  1. **Task Properties**: *e.g.*, translation invariance in addition.
  2. **Model Architecture**: *e.g.*, Absolute vs. Relative Positional Embeddings (APE vs. RPE).
  3. **Training Data Distribution**: What the model actually sees during training.

- To analyze this, we define two types of Out-of-Distribution (OOD) generalization for a model trained on n-digit numbers:
  - **Downward OOD Generalization**: Testing on shorter numbers ($< n$ digits).
  - **Upward OOD Generalization**: Testing on longer numbers ($> n$ digits). This is the key challenge.

Insight 1: Addition and Translation Invariance

Task Property: Digit-wise addition is (largely) translation-invariant. The algorithm to compute $c_i = (a_i + b_i + \text{carry}) \pmod{10}$ is the same for any position i.

- **RPE** models this invariance. It learns the relative computation, enabling successful upward OOD generalization.
- **APE** learns position-specific functions. It cannot generalize to unseen positions $(n+1, n+2, \ldots)$.
  - The model learns the function: $\hat{f}(a, b) = (a \pmod{10^n}) + (b \pmod{10^n})$.
  - This leads to failure in upward OOD generalization.

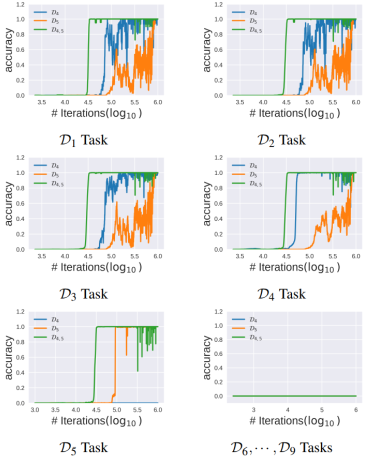

Figure 1: **OOD Test Accuracy (APE)**.

Insight 2: Multiplication and Lack of Invariance

Task Property: Multiplication is not translation-invariant.

- The calculation for digit c_k depends on a sum over all pairs of input digits (a_i, b_j) where $i + j = k + 1$.
- This creates complex, non-local dependencies that grow with the position k.

Result: The algorithm is too complex for the inductive biases of standard positional encodings.

- Both **APE** and **RPE** fail to generalize upward.

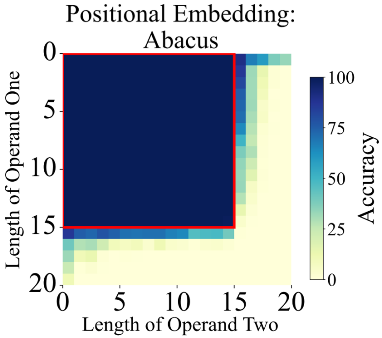

Figure 2: **Multiplication Failure (RPE)**. From McLeish et al. (2024), a model trained on up to 15 digits fails on longer inputs.

Insight 3: Modular Arithmetic and Base Alignment

Explaining the “mod 100 vs. mod 101” Puzzle

Key Insight

The model’s ability to generalize depends on whether the modulus p aligns with the number base (10).

Insight 3: Modular Arithmetic and Base Alignment

Case 1: p divides 10^k (e.g., $p = 100, 50, 200$)

- **Task Property**: The result only depends on the last k digits.
- $(a + b) \pmod{100} \equiv ((a \pmod{100}) + (b \pmod{100})) \pmod{100}$
- **Implication**: Higher-order digits are irrelevant. The model learns to ignore them, allowing perfect upward generalization, even with APE.

Case 2: p does not divide 10^k (e.g., $p = 101, 51, 151$)

- **Task Property**: The result depends on *all* digits. Higher-order digits matter.
- $(a + b) \pmod{101} \not\equiv ((a \pmod{100}) + (b \pmod{100})) \pmod{101}$
- **Implication**: The model (with APE, trained on n digits) learns the truncated function $\hat{f}^p(a, b) = ((a \pmod{10^n}) + (b \pmod{10^n})) \pmod{p}$, leading to upward generalization failure.

Quantitative Prediction: A "Smoking Gun" Result

For the hard case (modular addition where p does not divide 10^n), our theory makes a sharp, quantitative prediction for the accuracy on longer digits:

Theorem (Informal, Thm. 5 from paper)

The test accuracy for a model trained on n digits and tested on much longer digits ($n_{test} \gg n$) is approximately:

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

Modulus	**Experimental Test Accuracy (%) on $\widetilde{\mathcal{D}}_i$** $i = 4$ (ID)	$i = 5$	$i = 6$	$i = 7$	$i = 8$	$i = 9$	**Theory** $\gcd(p, 10^4)/p$
$p = 100$ (divides 10^4)	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 perfectly match the theoretical predictions!

Conclusion

- We proposed a **unified theoretical framework** that resolves long-standing puzzles about arithmetic generalization in Transformers.

- We showed that generalization is not magic, but emerges from the alignment between:
  - **Task structure** (e.g., symmetries like translation invariance)
  - **Model inductive biases** (e.g., relative positional encodings)
  - **Training data distribution** (which defines the function being learned)

- Our framework provides principled, quantitative, and experimentally validated explanations for OOD behavior.

- **Implications**: This work is a step towards more reliable and aligned AI, providing insights for data-efficient training and a deeper understanding of what neural networks learn.

Thank You

Paper and Code available at:

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