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

Anonymous ACL submission

Abstract

Transformer-based models excel in various tasks but their generalization capabilities, especially in arithmetic reasoning, remain incompletely understood. Arithmetic tasks provide a controlled framework to explore these capabilities, yet performance anomalies persist, such as inconsistent effectiveness in multiplication and erratic generalization in modular addition (e.g., modulo 100 vs. 101). This paper develops a unified theoretical framework for understanding the generalization behaviors of transformers in arithmetic tasks, focusing on length generalization. Through detailed analysis of addition, multiplication, and modular operations, we reveal that translation invariance in addition aligns with relative positional encoding for robust generalization, while base mismatch in modular operations disrupts this alignment. Experiments across GPT-family models validate our framework, confirming its ability to predict generalization behaviors. Our work highlights the importance of task structure and training data distribution for achieving data-efficient and structure-aware training, providing a systematic approach to understanding of length generalization in transformers.

1 Introduction

Since the introduction of Transformer (Vaswani et al., 2017), Transformer-based models including large language models (LLMs) and large multimodal models (LMMs) have experienced a rapid rise, excel in a wide range of tasks, such as natural language processing, coding, mathematical reasoning, and vision understanding (Bubeck et al., 2023; Lu et al., 2024). However, the generalization capabilities of these transformer based foundation models are not yet fully understood in areas such as natural language understanding (Bender et al., 2021) and mathematical reasoning (Anil et al., 2022; Jelassi et al., 2023).

The generalization capabilities are often link to models' capability to generalize beyond their training data (out-of-distribution (OOD) generalization) in NLP tasks, which is much complex and challenging. LLMs perform exceptionally well on some generalization tasks while produce factual errors or misinformation on others e.g., (Bender et al., 2021; Lu et al., 2024). Studies therefore try to figure out why these differences exist between generalization tasks (Briakou et al., 2023), what LLMs are actually learning on failed ones (Xu et al., 2024), and how they manage to generalize on successful tasks (Jelassi et al., 2023; McLeish et al., 2024).

Given the complexity of next-token prediction across diverse corpora and models' opacity, mathematical tasks (e.g., n-digit addition / multiplication / modular operations) serve as interpretable probes for generalization analysis.

However, mysterious discrepancies in models' generalization capability still exist – (1) certain tasks (e.g., addition) succeed in unseen generalization with certain positional encodings (e.g., relative) but not other tasks (e.g., multiplication), and (2) there is a significant generalization difference between very close moduli in modular operations (e.g., modulo 100 and 101). Specifically, previous studies have observed that when training models with absolute positional embeddings (APE) on n-digit operations (e.g., addition), where both input operands are no longer than n-digit in length such as $1234+5678$ for $n=4$, the models successfully generalize on unseen n-digit inputs such as $4321+8765$ (termed in-distribution (ID) generalization). However, they fail on longer unseen cases such as $91234+15678$ (termed OOD generalization) as shown by Anil et al. (2022), Jelassi et al. (2023), Lee et al. (2023), and Xu et al. (2024). Besides, models with relative positional embeddings (RPE) can generalize to longer unseen inputs for addition tasks but struggle with multiplication tasks, according to Jelassi et al. (2023) and McLeish et al. (2024). Additionally, models trained on modular

operations with specific moduli such as 100 can perfectly generalize to any longer unseen inputs with either absolute or relative positional embeddings. However, they fail to generalize to longer unseen inputs for other very close moduli such as 101, as noted by Jelassi et al. (2023). These OOD generalization mysteries are cataloged in Table 1.

	Addition	Multiplication	Modular Operations	
			$p = 100$	$p = 101$
APE	✗	✗	✓	✗
RPE	✓	✗	✓	✗

Table 1: Length Generalization of Transformers with APE and RPE on Arithmetic Tasks

As we can summarize, these previous efforts address generalization issues in specific tasks, modifying components of individual models, such as altering positional encodings (Jelassi et al., 2023; McLeish et al., 2024) or attention mechanisms (Dubois et al., 2019). Their failure in figuring out the underneath mechanism calls for a reflective examination – we believe the field has overlooked the differences in task properties (e.g., addition v.s. multiplication, modulo 10^2 v.s. modulo $10^2 + 1$) that may drive the difference in generalization property among tasks. The perspective of mechanistic interpretability (Hernandez et al., 2022; Liu et al., 2022) offers an angle in this direction. This data-driven and experimentally-based analytical approach has helped identify and interpret phenomena such as "grokking" (Liu et al., 2022) and analyze the impact of repeated data on the performance of LLMs (Hernandez et al., 2022).

In this paper, we present a unified theoretical framework integrating language modeling principles, universal approximation capabilities, and task-specific property analysis across diverse arithmetic tasks. Our model assumes that generalization behaviors depend on task properties once the model converges on the training data. For example, digital addition is translation invariant with a large probability, yielding consistent results despite digit shifts, aligning with RPE's preservation of positional relationships, unlike multiplication. This leads to well generalization of addition with a large probability to unseen longer domains under RPE but not for multiplication. The modulo (e.g. 100, 101) discrepancy stems from base alignment: modulo 100 matches base 10, discarding higher digits $11234 + 15678 \equiv 1234 + 5678 \equiv 34 + 78 \pmod{100}$, whereas modulo 101 requires them.

We then perform more extensive generalization analyzes assuming that transformer models are trained in n-digit operations with at least one operand having a length of n such as $1234 + 567$ for $n = 4$. This differs from the literature where the length of both operands is no longer than n. We categorize generalization into two types: *downward OOD generalization* and *upward OOD generalization*. Downward OOD generalization[1] involves generalizing to downward domains, such as $120 + 235$ or $11 + 32$, while upward OOD generalization involves generalizing to upward domains, such as $12035 + 235$ or $123456 + 323456$. The core conclusions of our theoretical analysis are as follows: (1) For addition, under APE, Transformer models can generalize to the downward (downward) OOD domain, but not to the upward (upward) OOD domain. However, under RPE, the models can generalize to both downward and upward OOD domains, benefiting from the translation invariance of digit addition. (2) For multiplication, even RPE has limited effectiveness in the upward OOD domain due to the lack of translation invariance property. (3) For modular operations, if the modulus p divides 10^n, models can generalize to both downward and upward OOD domains regardless of the positional encoding, due to the compatibility with base 10 such that the information at higher-digit positions of the operands do not affect the result. When the modulus p does not divide 10^n, models can only generalize to the downward OOD domain. For upward OOD domains, we have derived a theoretical accuracy formula based on the information loss and identification of the model's final learned function.

The challenge in understand the generalization capacity of LLM has significant implications for LLM training, alignment, and application (Ji et al., 2023). Our analysis highlights the importance of training data distribution. If the data excluded from the training dataset does not affect the desired ground truth support set, such as when the downward OOD domain is excluded during training, the model can still learn to generalize to the excluded downward OOD domain. However, if a significant

[1]As a note, the downward OOD domain generalization is not trivial. If a model is trained on a smaller domain with a significant gap from the desired training dataset, such as training on n-digit addition with both operands having a length of n and the highest digits of both operands being greater than, for example, 5, the model fails to generalize to the downward OOD domain.

amount of data is omitted, or a large number of training samples are mapped to the same answer, as shown in our counterexample above, the downward OOD domain generalization fails. Therefore, when our goal is to align the model to generalize certain OOD domains as expected, precise analysis of the task nature and careful control of the training data are necessary.

To validate our theoretical framework, we experiment on various generative language models, including models of various sizes (Karpathy, 2023), and our tasks involving n-digit addition, multiplication, and modular operations. We further perform robustness analysis across different model scales, dataset sizes, and training data schemes.

Our main contributions are as follows:

1. Establishing a unified theoretical framework for understanding OOD generalization of Transformers: Our framework is the first to address task differences in transformer models' generalization ability. Comprehensive experimental evidences validate our theoretical predictions[2].

2. Clarifying the downward and upward OOD generalization and their requirement of task and training data. We introduce the concepts of downward and upward generalization, which more clearly delineates the differences between generalization to downward and upward domains.

2 Related Work

Generalization of Transformers and LLMs on Arithmetic. Numerous studies have examined the performance of Transformer-based language models in tasks involving arithmetic operations and mathematical reasoning. Brown et al. (2020), Bubeck et al. (2023) and Lu et al. (2024) investigated various LLMs, such as GPT-3, GPT-4, and Gemini, in performing basic arithmetic and mathematical reasoning. Nogueira et al. (2021) explored the limitations of Transformers in learning arithmetic, highlighting the significant influence of surface representation on model accuracy and the need for improved tokenization and positional encoding strategies. Subsequent research such as Qian et al. (2022), Anil et al. (2022), Jelassi et al. (2023), Lee et al. (2023), Xu et al. (2024), McLeish et al. (2024) and Duan et al. (2024). Abbe et al. (2023) examined generalization on unseen logical functions. While previous studies have mainly focused on evaluating or improving generalization capabilities, our work develops a unified theoretical framework to analyze OOD generalization behaviors in Transformer models trained on arithmetic operations, bridging the gap between empirical observations and theoretical understanding.

Mechanistic Interpretability and General Understanding. Many studies have focused on understanding and interpreting the working dynamics of neural networks and Transformer models (Zhang et al., 2021; Hernandez et al., 2022; Elhage et al., 2022; Bills et al., 2023; Templeton, 2024). From the perspective of universal approximation, Yun et al. (2019) and Alberti et al. (2023) demonstrated that Transformer models equipped with trainable positional encodings can act as universal approximators for continuous functions in a compact domain under the L^p norm or the supremum norm.

From a mechanistic viewpoint, Hernandez et al. (2022) investigated the impact of repeated data on the performance of LLMs, highlighting significant performance degradation when a small fraction of data is repeated multiple times. Liu et al. (2022) addressed the phenomenon of delayed generalization or "grokking" using addition and modular addition tasks, and Zhong et al. (2023) utilized modular addition to mechanistically explain algorithm discovery in neural networks.

Our work contributes to this growing field of mechanistic interpretability by providing a macroscopic explanation specifically for Transformer models. We systematically identify systematic biases and understand model behaviors in arithmetic reasoning scenarios.

3 Theoretical Analysis on Generalization for Arithmetic Reasoning

We first review the Transformer model and the universal approximation theorem, and then conduct theoretical analyses of the downward and upward OOD generalization capabilities of the Transformer in solving tasks related to addition, modular addition, multiplication, and modular multiplication.

3.1 Preliminaries on Transformer and Universal Approximation

A Transformer model (Vaswani et al., 2017) predicts the next token based on the preceding tokens within the input sequence. Its output is subsequently used as input for the next prediction. For a target token x_i at position i in the sequence, the

[2]Our opensource our code at https://anonymous.4open.science/r/ArithmeticLLM-034D under the MIT license

model generates a probability distribution over the vocabulary of the next potential tokens. To be precise, let $x = x_1 x_2 \ldots x_T \in \mathcal{V}^T$ denote the input sequence of tokens. The probability of observing this sequence with respect to a Transformer model is given as follows:

$$\mathbf{P}_\theta(x) = \prod_{i=1}^{T} \mathbf{P}_\theta(x_i|x_1, x_2, ..., x_{i-1}) = \prod_{i=1}^{T} \mathbf{P}_\theta(x_i|x_{<i}).$$

The conditional probability $\mathbf{P}_\theta(x_i|x_{<i})$ is computed using the softmax function applied to the last hidden state.

Universal approximation theorem for Transformer models: Transformer models have the capacity to universally approximate any arbitrary continuous sequence-to-sequence function within a compact domain. Yun et al. (2019) and Alberti et al. (2023) have shown that, when equipped with trainable positional encodings, Transformers can serve as universal approximators for continuous functions in a compact domain under the L^p norm or the supremum norm. These characterizations highlight the representation power of fixed-width Transformer networks, despite the intrinsic parameter sharing and permutation equivariance.

3.2 Theoretical Analysis on Addition

Consider two natural numbers $a = \sum_{i=1}^{n} a_i \times 10^{i-1} = (a_1, a_2, \cdots, a_n)$ and $b = \sum_{i=1}^{n} b_i \times 10^{i-1} = (b_1, b_2, \cdots, b_n)$. The addition of these n-digit numbers, denoted as $f(a,b) = a + b$, is expressed by $c = \sum_{i=1}^{n+1} c_i \times 10^{i-1} = (c_1, c_2, \cdots, c_n, c_{n+1})$.

Let the dataset $\mathcal{D}_n := \{(a,b) \in \mathbb{N}^2 : a_n \vee b_n \geq 1, a_i = b_i \equiv 0, \forall i > n\}$. For notation simplicity, assume $(0,0) \in \mathcal{D}_1$. Here, $a_n \vee b_n = \max\{a_n, b_n\}$. Note that $\mathcal{D}_n \cap \mathcal{D}_m = \emptyset$ for $n \neq m$ and $\mathbb{N}^2 = \bigcup_{n=1}^{\infty} \mathcal{D}_n$. Denote the downward (downward) domain $\mathcal{D}_{<n} := \bigcup_{m=1}^{n-1} \mathcal{D}_m$ and the upward domain $\mathcal{D}_{>n} := \bigcup_{m=n+1}^{\infty} \mathcal{D}_m$.

Theorem 1. ***(Informal)*** *Assume a Transformer model with absolute positional embedding (APE) is trained on a multi-digit addition dataset for the operands $(a,b) \in \mathcal{D}_n$ ($n \geq 2$) with infinite training computation, then the learned model can perfectly generalize for the downward OOD domain $\mathcal{D}_{<n}$, but fail for the upward OOD domain $\mathcal{D}_{>n}$.*

Proof Sketch. Assume a Transformer model is trained on this dataset $\mathcal{D}_n$ using absolute positional embeddings (APE). The model is trained to approximate the function that computes the sum digit by digit, with carries propagated as follows:

$$c_i = \zeta(a_i + b_i + c_{i-1}^{\chi}),$$

where c_{i-1}^{χ} is the carry from the previous position, and ζ is a function taking the units of the input.

Case I: Downward OOD Domain ($\mathcal{D}_{<n}$)

For positions $i \leq n$, the model can generalize well to the downward OOD domain $\mathcal{D}_{<n}$ by *universal approximation theorem for Transformer models*. Since the model has seen all possible carry combinations during training, it can correctly predict the digit sums at positions $i = 1, 2, \ldots, n$. For position $i = n+1$, the model predicts the carry $c_{n+1} = c_n^{\chi} \in \{0,1\}$ for all pairs where $a_n \vee b_n \geq 1$, and when both $a_n = b_n = 0$, the model learns $c_{n+1} = 0$. For positions $i > n+1$, the model predicts zero, since the input digits a_i and b_i are zero beyond the n-th position. Thus, the model perfectly generalizes to $\mathcal{D}_{<n}$.

Case II: Upward OOD Domain ($\mathcal{D}_{>n}$)

For positions $i \leq n$, the model behaves similarly to the downward OOD case. However, when $i = n+1$, the model is unable to predict the correct sum. The probability distribution learned by the model at this position only supports values in $\{0,1\}$, but for the model to correctly predict the carry, the support must include $\{0,1,\ldots,9\}$. Since the model has never seen pairs where both a_{n+1} and b_{n+1} are non-zero, it cannot generalize correctly to the upward OOD domain. Beyond position $n+1$, the model will predict zeros, as $a_i = b_i = 0$ for all $i > n$. Thus, the model fails to generalize to $\mathcal{D}_{>n}$. □

Based on the analysis above, we can immediately draw the following conclusion, which provides an explanation for the findings by Xu et al. (2024).

Corollary 2. ***(Informal)*** *The learned Transformer model with APE approximates the function $\hat{f}(a,b) = (a \mod 10^n) + (b \mod 10^n)$. The OOD generalization error is zero for the downward OOD domain $\mathcal{D}_{<n}$, but not less than 10^n for every point in the upward OOD domain $\mathcal{D}_{>n}$.*

We are curious about the conditions under which a Transformer model can learn to perform addition operations. With APE, the model successfully generalizes downward, but fails to generalize upward. What would be the conclusion under RPE? Through theoretical and experimental analysis, we have arrived at the following conclusions.

Theorem 3. ***(Informal)*** *Assume a Transformer model with relative/abacus positional embedding (RPE) is trained on a multi-digit addition dataset*

for the operands $(a,b) \in \mathcal{D}_n$ ($n \geq 2$) with infinite training computation, then the learned model can perfectly generalize for the downward OOD domain $\mathcal{D}_{<n}$ and generalize well for the upward OOD domain $\mathcal{D}_{>n}$, with a probability of failure in the upward domain being less than $1/10^{n-1}$.

Proof Sketch. A Transformer model with relative positional embeddings (RPE) has a key property of *translation invariance*. This means the model's predictions at any position i depend only on the relative distances between positions, not their absolute locations.

Special Case: Translation Invariance

Translation invariance can be expressed as:

$$\mathbf{P}_\theta(c_i \mid a_{\leq i}, b_{\leq i}) = \mathbf{P}_\theta(c_i \mid a_{i-1}, a_i, b_{i-1}, b_i),$$

ensuring that the carry at each position is determined by the preceding digits a_{i-1}, b_{i-1}, and not their absolute positions. Thus, the sum at position i is:

$$c_i = \zeta(a_i + b_i + c_{i-1}^{\chi}),$$

where $c_{i-1}^{\chi} = \chi(a_{i-1} + b_{i-1})$, as long as $a_{i-1} + b_{i-1} \neq 9$.

General Case: Extended Translation Invariance

For longer sequences, the prediction for c_i depends on the relative positions $a_{i-n+1}, \cdots, a_i, b_{i-n+1}, \cdots, b_i$. The translation invariance fails when carry propagation extends past the n-th digit, which happens if $a_{i-k} + b_{i-k} = 9$ for all $k = 1, \ldots, n-1$. The probability of this failure is small, less than $1/10^{n-1}$. Thus, the model effectively handles longer sequences by mapping them to shorter ones with similar relative distances, with the failure probability in the upward domain being less than $1/10^{n-1}$. □

3.3 Theoretical Analysis on Modular Addition

Consider the function for modular addition with a modulus p, expressed as $f(a,b) = (a+b) \mod p$, which will be the focus of our analysis in the following section. Subsequently, we will also represent modular addition using the notation $\overline{c}^p = \overline{a+b}^p$. For simplicity, we will omit the superscript p when it is clear from the context.

Scenarios on Divisibility of 10's Power by Modulus

Theorem 4. ***(Informal)*** *Assume a Transformer model with either absolute or relative/abacus positional embedding is trained on a multi-digit modular addition dataset with a modulus p that divides 10^m for the operands $(a,b) \in \mathcal{D}_n$ ($n \geq 2$ and $m \leq n$) with infinite training computation, then the learned model can perfectly generalize both for the downward OOD domain $\mathcal{D}_{<n}$ and the upward OOD domain $\mathcal{D}_{>n}$.*

Scenarios on Non-Divisibility of 10's Power by Modulus

Theorem 5. ***(Informal)*** *(1) Assuming a Transformer model equipped with absolute positional embeddings is trained on a multi-digit modular addition dataset $\mathcal{D}_n$ ($n \geq 2$) where the modulus p neither divides 10^n nor exceeds 10^n, and provided that infinite training computation is allocated, then the resulting trained model is capable of perfect generalization to the downward OOD domain $\mathcal{D}_{<n}$, while encountering difficulties in generalizing to the upward OOD domain $\mathcal{D}_{>n}$. (2) The function that the model has learned is $\hat{f}^p(a,b) = \overline{\overline{a}^{10^n} + \overline{b}^{10^n}}^p$. (3) Furthermore, the test accuracy on $\widetilde{\mathcal{D}}_{n_{test}}$ ($n_{test} > n$) is given by $\mathrm{Acc}(p,n,n_{test}) \approx \frac{\gcd(p,10^n)}{p}$ if $n_{test} \geq n + \log_{10}(p'/2+1)$, otherwise $\mathrm{Acc}(p,n,n_{test}) = 0$, where $\gcd(p,10^n)$ represents the greatest common divisor of p and 10^n, and $p' = p/\gcd(p,10^n)$.*

3.4 Theoretical Analysis on Multiplication

Theorem 6. ***(Informal)*** *(1) Assuming a Transformer model equipped with absolute positional embeddings is trained on a multi-digit multiplication dataset $\mathcal{D}_n$ ($n \geq 2$), and provided that infinite training computation is allocated, then the resulting trained model is capable of perfect generalization to the downward OOD domain $\mathcal{D}_{<n}$, while it cannot generalize to the upward OOD domain $\mathcal{D}_{>n}$. (2) The function that the model has learned is $\hat{f}(a,b) = \overline{a}^{10^n} \times \overline{b}^{10^n}$.*

3.5 Theoretical Analysis on Modular Multiplication

Theorem 7. ***(Informal)*** *(1) Assume that a Transformer model with absolute or relative/abacus positional embedding is trained on a multidigit modular multiplication dataset with a modulus p that divides 10^m for operands $(a,b) \in \mathcal{D}_n$ ($n \geq 2$ and $m \leq n$) with infinite training computation, then the learned model can perfectly generalize both for the downward OOD domain $\mathcal{D}_{<n}$ and the upward OOD domain $\mathcal{D}_{>n}$. (2) If the modulus p neither divides 10^n nor exceeds 10^n, and provided that infinite training computation is allo-*

cated, then the resulting trained model is capable of perfect generalization to the downward OOD domain $\mathcal{D}_{<n}$, while encountering difficulties in generalizing to the upward OOD domain $\mathcal{D}_{>n}$. The function that the model with APE has learned is $\hat{f}^p(a,b) = \overline{\bar{a}^{10^n} \times \bar{b}^{10^n}}^p$.

4 Experiments

In this section, we describe our experiment design with result outcome validating the prediction make using our theoretical framework. We also conducted additional experiment providing detailed investigation into the mechanism as well as checking for robustness of our result which are provided in Appendix.A.

4.1 Experimental Design

Model Description: In line with most LLMs, we utilize a decoder-only architecture consisting of multiple layers and multi-head attentions. Our models are trained from scratch with varying model scale[3]. Detailed configuration of training and architecture are provided in Table3 (see Appendix).

Data Description: We employ 4 four primary arithmetic operations with different symmetric property as well as difficulty in term of how much a digit can have impact in term of upward/downward generalization, which are described here:

- **Addition:** $c = a + b$
- **Modular addition:** $c \equiv a + b \pmod{p}$
- **Multiplication:** $c = a \times b$
- **Modular multiplication:** $c \equiv a \times b \pmod{p}$

We randomly generate datasets for each arithmetic task. Following (Lee et al., 2023; Xu et al., 2024) we organize our training data as a sequence of operand pairs in natural order, with the results of the operations in reversed order with character-level tokenization[4], which has been shown to be more effective for learning in next-token prediction models in arithmetic tasks[5].

[3]The models architecture are in respect NanoGPT, MicroGPT, and MiniGPT (Karpathy, 2023)

[4]After the tokenization, ";", "[bos]", and "[eos]", a "line break" token are added to the beginning and the end of each line of data, resulting in a vocabulary size of 16. When the context window exceeds the required size for n-digit arithmetic operations, we pad zeros before the numbers "a", "b", and "c".

[5]For example, consider an n-digit addition $a+b=c$, represented in standard format as "$a_n \cdots a_2 a_1 + b_n \cdots b_2 b_1 = c_{n+1} \cdots c_2 c_1$". By reversing the order of the output "c", we obtain the reversed data format "$a_n \cdots a_2 a_1 + b_n \cdots b_2 b_1 = c_1 \cdots c_n c_{n+1}$".

We control the length of arithmetic operations n and randomly generate datasets from $\mathcal{D}_n$ for different lengths n. These datasets for each arithmetic task are categorized into three distinct subsets: a training set, an in-distribution (ID) test set, and additional out-of-distribution (OOD) test sets which we further break down by the upper bound digit for upward generalization, sampled from m-digit operations with $m \neq n$. The case where $m < n$ is referred to as the downward (downward) OOD domain, and the case where $m > n$ is termed the upward (upward) OOD domain. We also construct numerous combination sets of samples from different domains $\mathcal{D}_n$, such as $\mathcal{D}_{n-1,n}$, to be used as training and ID test datasets. In the demonstrative example, the OOD test sets are sampled from $\mathcal{D}_m$ with $m \neq n-1$ and n. The test accuracy is measured using maximum probability sampling.

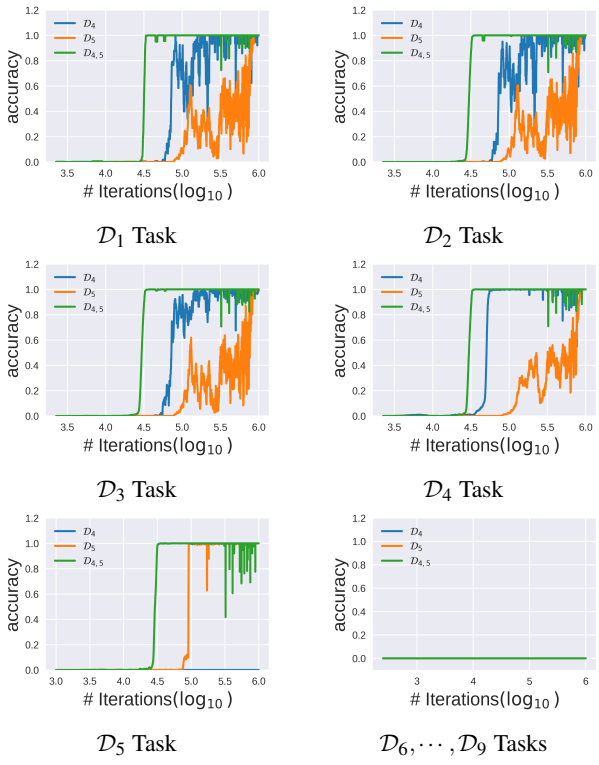


Figure 1: Test Accuracy of Transformer Models with APE for Different Multi-digit Addition Tasks

Note: This figure presents results from three experiments using different training datasets with the MiniGPT model and a learned APE. The labels $\mathcal{D}_4$, $\mathcal{D}_5$, and $\mathcal{D}_{4,5}$ indicate training on random samples from $\mathcal{D}_4$, $\mathcal{D}_5$, and a combined subset of both, respectively. Each subfigure shows test accuracy across different domains $\mathcal{D}_i$ during training.

Modulus	Test Accuracy (%) w.r.t. the Ground Truth on the Domain $\widetilde{\mathcal{D}}_i$ 1	2	3	4	5	6	7	8	9	Theory $1/p'$
$p=50$	100	100	100	100	99.3	92.0	93.1	95.2	91.4	100
$p=51$	100	98.5	99.9	99.3	0.3	1.8	1.9	1.9	1.6	1.96
$p=100$	100	100	100	100	100	100	100	100	100	100
$p=101$	100	100	100	100	0.0	1.2	0.9	1.1	1.0	0.99
$p=150$	100	100	100	100	33.2	33.6	32.3	33.0	33.7	33.3
$p=151$	100	99.9	99.9	100	0.0	0.6	0.7	0.7	0.6	0.66
$p=200$	100	100	100	100	99.8	98.9	93.7	94.1	93.5	100
$p=201$	100	100	99.9	99.9	0.0	0.0	0.5	0.4	0.5	0.50

Table 2: Modular Addition: Test Accuracy w.r.t. the Ground Truth $f^p(a,b) = \overline{a+b}^p$ on $\widetilde{\mathcal{D}}_i$

4.2 Experiments on Addition

In this subsection, we trained multiple models on different datasets (e.g. $\mathcal{D}_4$, $\mathcal{D}_5$, $\mathcal{D}_{4,5}$) and tracked the changes in their accuracy. Additionally, we demonstrated how the models learn each digit during the training process.

4.2.1 Generalization for Different Digit Tasks

In Figure 1, we present the results of three different experiments using distinct training datasets (i.e., $\mathcal{D}_4$, $\mathcal{D}_5$, $\mathcal{D}_{4,5}$). For all experiments, we employ the MiniGPT model equipped with a learned APE. Each subfigure illustrates the test accuracy on different test domains $\mathcal{D}_i$ for these models throughout the training process. Figure 1 verifies our Theorem 1. It demonstrates that models incorporating APE are unable to generalize to longer digits than those they are trained on but can succeed with lower digits. Additionally, the model trained on $\mathcal{D}_5$ has a much more challenging training process compared to the model trained on $\mathcal{D}_4$, while the model trained on $\mathcal{D}_{4,5}$ experiences the easiest and smoothest training process among the three models. The reason, as explained in Theorem 1, is that for $\mathcal{D}_{4,5}$, the model learns addition tasks on lower digits directly from the training data. In contrast, $\mathcal{D}_4$ and $\mathcal{D}_5$ require OOD generalization for the edge positions.

More results can be found in Table 4 and Table 5. We test the final trained model on datasets with varying digit lengths. While the models do not learn the addition of higher digits, they successfully learn the operation $\hat{f}(a,b) = \overline{a}^{10^n} + \overline{b}^{10^n}$, supporting our Corollary 2.

We also conduct extensive experiments using various training datasets, model scales, and data scales. The results of these experiments are robust, and presented in Appendix.

4.2.2 Learning Dynamics for Each Digit Position

The models and training datasets are identical to those described in Figure 1. We have assembled a comprehensive test dataset that contains a random sample from $\mathcal{D}_1$ to $\mathcal{D}_9$. Our objective is to demonstrate how these Transformer models equipped with APE learn each digit at every position throughout the training phase. The digit-wise test accuracy is defined as the accuracy of the prediction for each position in the result c.

The plots in Figure 4 (see Appendix) visually represent whether these models are capable of accurately predicting the digits c_i at all positions. These graphs effectively illustrate the learning dynamics for each token in the context of addition tasks. The models exhibit high accuracy for the first four or five digits, with accuracy approaching 1.0 as training progresses, for datasets $\mathcal{D}_4$, or $\mathcal{D}_5$, and $\mathcal{D}_{4,5}$, respectively. However, accuracy sharply declines for the 5th or 6th digits and remains near zero for the 7th, 8th, and 9th digits. These findings illustrate that while the models can effectively learn and predict lower-position digits, they struggle significantly with higher-position digits. This aligns with the theorem that Transformer models with APE can generalize well for downward OOD domains but fail for upward OOD domains.

4.2.3 Generalization Under Relative/Abacus Positional Embeddings

McLeish et al. (2024) conducted experiments using a 16-layer Transformer (decoder only) model with abacus positional embedding, trained on a random sample from $\mathcal{D}_{\leq 20}$. It can generalize on 100-digit addition problems (see Figure 7 in Appendix.[6] Ad-

[6]Code to reproduce the results can be found on GitHub:

ditionally, Jelassi et al. (2023) demonstrated that relative positional embeddings enable length generalization in addition tasks. In their work, models such as Transformer and Universal Transformer (encoder only) trained to add 5-digit numbers could generalize to 20-digit operands.

These results provide empirical evidence validating our Theorem 3 for upward OOD generalization. The findings are clear, and we will not replicate the procedures here. Instead, we reference these studies in the present context.

4.3 Experiments on Modular Addition

The results in Table 2 validate Theorem 4, which states that Transformer models with absolute positional embeddings trained on multi-digit modular addition datasets exhibit distinct generalization capabilities based on the modulus p. For moduli such as $p = 50, 100, 200$ that divide 10^n, the models achieve perfect test accuracy across all digit domains, demonstrating their ability to generalize flawlessly to both downward and upward OOD domains. In contrast, for moduli such as $p = 51, 101, 150, 151, 201$ that do not divide 10^n, the models maintain high accuracy for lower digit domains but show significant performance degradation for higher digit positions[7].

The OOD test accuracy in Table 2 for high-order digits can be completely expected using Theorem 5, which states that the test accuracy on $\widetilde{\mathcal{D}}_{n_{\text{test}}}$ ($n_{\text{test}} > n$) is given by $\text{Acc}(p, n, n_{\text{test}}) \approx 1/p'$ if $n_{\text{test}} \geq n + \log_{10}(p'/2+1)$, otherwise $\text{Acc}(p, n, n_{\text{test}}) = 0$. These observations align well with the theoretical expectations outlined in Theorem 4 and Theorem 5, also explaining the experimental results found in the literature (see, e.g., Jelassi et al. (2023)) in handling modular addition tasks with different moduli.

Furthermore, the results in Table 6 (see Appendix) support Theorem 5, indicating that Transformer models with absolute positional embeddings trained on multi-digit modular addition datasets learns the function $\hat{f}^p(a,b) = \overline{a^{10^n} + b^{10^n}}^p$ for any modulus p. These findings fully align with the theoretical predictions.

https://github.com/mcleish7/arithmetic

[7] The task of performing addition modulo 150 requires an extended training duration in our experiment. To facilitate this, we prime the training process with samples that have downward additions.

4.4 Experiments on Multiplication and Modular Multiplication

We also conducted extensive experimental analyses for multiplication and modular multiplication tasks, examining the performance and generalization capabilities of Transformer models. These experiments are designed to test various configurations, including different positional encodings, model size and training data schemes. Detailed results and additional analyses are available in Appendix.The experimental outcomes consistently support our theoretical framework, demonstrating the robustness of our approach and providing further insights into the behavior of Transformer models in arithmetic reasoning tasks.

5 Discussion

Our study sheds light on the *mechanistic interpretability* of Transformer models. Understanding the learning mechanisms is crucial for ensuring the meaningfulness of learned representations.

Additionally, our work identifies challenges associated with different training data schemes, such as concatenation training without padding [8] and line-by-line padding training [9]. These approaches can significantly impact model performance and generalization. Further understanding on these problems is essential for refining training strategies to improve model robustness and generalization.

6 Conclusion

In this paper, we developed a unified theoretical framework to explain OOD generalization in Transformer models trained on arithmetic operations, categorizing generalization into downward OOD (downward domains) and upward OOD (upward domains). Our analysis highlights the interactions among task properties, training data coverage, and model characteristics. Experiments with NanoGPT, MicroGPT, and MiniGPT validate our predictions, highlighting the framework's robustness. This work clarifies generalization mechanisms and provides insights for efficient model training and AI alignment. Future research should extend this framework to more complex tasks and factors influencing OOD generalization.

[8] e.g. "$123+45=168;267+1=268;$" as input.

[9] e.g. "$123+45=168$;[pad][pad][pad]" as input.

7 Limitation

This paper presents a unified theoretical framework for understanding generalization in transformers applied to arithmetic tasks. However, there are notable limitations to our analysis. Firstly, our focus on length generalization may overlook other critical aspects of out-of-distribution (OOD) generalization, as the representations learned for different tasks can exhibit varying relationships with length.

We selected arithmetic tasks for this study due to their clarity in distinguishing between downward and upward OOD generalization, as well as our ability to control the training data distribution effectively. Nonetheless, our framework's predictions are predicated on the assumption that the model has converged on the training data, which may not always hold true in practice, particularly given that many large language models (LLMs) remain undertrained.

Additionally, while our findings provide insights into generalization behaviors, they may not fully encompass the complexities involved in more intricate mathematical reasoning or other types of sequence-to-sequence tasks. Future work should explore these broader contexts to enhance our understanding of transformer generalization.

References

Emmanuel Abbe, Samy Bengio, Aryo Lotfi, and Kevin Rizk. 2023. Generalization on the unseen, logic reasoning and degree curriculum. *International Conference on Machine Learning*.

Silas Alberti, Niclas Dern, Laura Thesing, and Gitta Kutyniok. 2023. Sumformer: Universal approximation for efficient transformers. In *Topological, Algebraic and Geometric Learning Workshops 2023*, pages 72–86. PMLR.

Cem Anil, Yuhuai Wu, Anders Andreassen, Aitor Lewkowycz, Vedant Misra, Vinay Ramasesh, Ambrose Slone, Guy Gur-Ari, Ethan Dyer, and Behnam Neyshabur. 2022. Exploring length generalization in large language models. *Advances in Neural Information Processing Systems*, 35:38546–38556.

Emily M Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. 2021. On the dangers of stochastic parrots: Can language models be too big? In *Proceedings of the 2021 ACM conference on fairness, accountability, and transparency*, pages 610–623.

Steven Bills, Nick Cammarata, Dan Mossing, Henk Tillman, Leo Gao, Gabriel Goh, Ilya Sutskever, Jan Leike, Jeff Wu, and William Saunders. 2023. Language models can explain neurons in language models. *URL https://openaipublic. blob. core. windows. net/neuron-explainer/paper/index. html.(Date accessed: 14.05. 2023)*, 2.

Eleftheria Briakou, Colin Cherry, and George Foster. 2023. Searching for needles in a haystack: On the role of incidental bilingualism in palm's translation capability. *arXiv preprint arXiv:2305.10266*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, et al. 2023. Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv preprint arXiv:2303.12712*.

Shaoxiong Duan, Yining Shi, and Wei Xu. 2024. From interpolation to extrapolation: Complete length generalization for arithmetic transformers. *arXiv preprint arXiv:2310.11984*.

Yann Dubois, Gautier Dagan, Dieuwke Hupkes, and Elia Bruni. 2019. Location attention for extrapolation to longer sequences. *arXiv preprint arXiv:1911.03872*.

Nelson Elhage, Tristan Hume, Catherine Olsson, Nicholas Schiefer, Tom Henighan, Shauna Kravec, Zac Hatfield-Dodds, Robert Lasenby, Dawn Drain, Carol Chen, et al. 2022. Toy models of superposition. *arXiv preprint arXiv:2209.10652*.

Danny Hernandez, Tom Brown, Tom Conerly, Nova DasSarma, Dawn Drain, Sheer El-Showk, Nelson Elhage, Zac Hatfield-Dodds, Tom Henighan, Tristan Hume, et al. 2022. Scaling laws and interpretability of learning from repeated data. *arXiv preprint arXiv:2205.10487*.

Samy Jelassi, Stéphane d'Ascoli, Carles Domingo-Enrich, Yuhuai Wu, Yuanzhi Li, and François Charton. 2023. Length generalization in arithmetic transformers. *arXiv preprint arXiv:2306.15400*.

Jiaming Ji, Tianyi Qiu, Boyuan Chen, Borong Zhang, Hantao Lou, Kaile Wang, Yawen Duan, Zhonghao He, Jiayi Zhou, Zhaowei Zhang, et al. 2023. Ai alignment: A comprehensive survey. *arXiv preprint arXiv:2310.19852*.

Andrej Karpathy. 2023. The simplest, fastest repository for training/finetuning medium-sized gpts: nanogpt. *GitHub https://github.com/karpathy/nanoGPT*.

Nayoung Lee, Kartik Sreenivasan, Jason D Lee, Kangwook Lee, and Dimitris Papailiopoulos. 2023. Teaching arithmetic to small transformers. *arXiv preprint arXiv:2307.03381*.

Ziming Liu, Ouail Kitouni, Niklas S Nolte, Eric Michaud, Max Tegmark, and Mike Williams. 2022. Towards understanding grokking: An effective theory of representation learning. *Advances in Neural Information Processing Systems*, 35:34651–34663.

Chaochao Lu, Chen Qian, Guodong Zheng, Hongxing Fan, Hongzhi Gao, Jie Zhang, Jing Shao, Jingyi Deng, Jinlan Fu, Kexin Huang, et al. 2024. From gpt-4 to gemini and beyond: Assessing the landscape of mllms on generalizability, trustworthiness and causality through four modalities. *arXiv preprint arXiv:2401.15071*.

Sean McLeish, Arpit Bansal, Alex Stein, Neel Jain, John Kirchenbauer, Brian R Bartoldson, Bhavya Kailkhura, Abhinav Bhatele, Jonas Geiping, Avi Schwarzschild, et al. 2024. Transformers can do arithmetic with the right embeddings. *arXiv preprint arXiv:2405.17399*.

Rodrigo Nogueira, Zhiying Jiang, and Jimmy Lin. 2021. Investigating the limitations of transformers with simple arithmetic tasks. *arXiv preprint arXiv:2102.13019*.

Jing Qian, Hong Wang, Zekun Li, Shiyang Li, and Xifeng Yan. 2022. Limitations of language models in arithmetic and symbolic induction. *arXiv preprint arXiv:2208.05051*.

Adly Templeton. 2024. *Scaling monosemanticity: Extracting interpretable features from claude 3 sonnet*. Anthropic.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Xingcheng Xu, Zihao Pan, Haipeng Zhang, and Yanqing Yang. 2024. It ain't that bad: Understanding the mysterious performance drop in ood generalization for generative transformer models. *The 33rd International Joint Conference on Artificial Intelligence (IJCAI-24)*, pages 6578–6586.

Chulhee Yun, Srinadh Bhojanapalli, Ankit Singh Rawat, Sashank J Reddi, and Sanjiv Kumar. 2019. Are transformers universal approximators of sequence-to-sequence functions? *arXiv preprint arXiv:1912.10077*.

Yu Zhang, Peter Tiňo, Aleš Leonardis, and Ke Tang. 2021. A survey on neural network interpretability. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 5(5):726–742.

Ziqian Zhong, Ziming Liu, Max Tegmark, and Jacob Andreas. 2023. The clock and the pizza: Two stories in mechanistic explanation of neural networks. *arXiv preprint arXiv:2306.17844*.

A Appendix on Transformer

A Transformer model (Vaswani et al., 2017) predicts the next token based on the preceding tokens within the input sequence. Its output is subsequently used as input for the next prediction. For a target token x_i at position i in the sequence, the model generates a probability distribution over the vocabulary of potential next tokens. To be precise, let $x = x_1 x_2 \dots x_T \in \mathcal{V}^T$ denote the input sequence of tokens. The probability of observing this sequence with respect to a Transformer model is given as follows:

$$\mathbf{P}_\theta(x) = \prod_{i=1}^{T} \mathbf{P}_\theta(x_i | x_1, x_2, ..., x_{i-1}) = \prod_{i=1}^{T} \mathbf{P}_\theta(x_i | x_{<i}).$$

The conditional probability $\mathbf{P}_\theta(x_i|x_{<i})$ is computed using the softmax function applied to the last hidden state. One way to design this model (see e.g. Karpathy (2023), Brown et al. (2020)) is as follows:

$$\begin{aligned} a^{\ell-1} &= h^{\ell-1} + \mathrm{MHA}_\ell(\mathrm{LN}^A_\ell(h^{\ell-1})) \\ h^{\ell} &= a^{\ell-1} + \mathrm{MLP}_\ell(\mathrm{LN}^F_\ell(a^{\ell-1})) \end{aligned}$$

for $\ell = 1, 2, \dots, L$, with the initial embedding $h^0 = e_{tok} + e_{pos}$, where e_{tok} represents the initial token embedding and e_{pos} represents the positional embedding. In the context of GPT-series LLMs, MHA_ℓ refers to the masked multi-head attention of the ℓ-th layer, MLP_ℓ is a multi-layer perception with one hidden layer, and LN represents layer normalization. Define f_ℓ such that $h^\ell = f_\ell(h^{\ell-1})$. Consequently, the final hidden state of this LLM is

$$h^L = f_L \circ \dots \circ f_2 \circ f_1(h^0) \in \mathbb{R}^{d_m \times T},$$

where d_m is the embedding dimension.

Let $X = \mathrm{LN}(h^L) = [X_1, X_2, \dots, X_T]$. The final output conditional probability matrix

$$\begin{aligned} \mathbf{P}_\theta &= \mathrm{softmax}(WX) \\ &= \left(\frac{\exp(WX_i)}{\sum_{j=1}^{N} \exp(WX_i)_j} \right)_{i=1,2,\cdots,T} \in [0,1]^{N_V \times T}, \end{aligned}$$

where $W \in \mathbb{R}^{N_V \times d_m}$ is a weight matrix. The i-th column of the matrix $\mathbf{P}_\theta$ represents the conditional probability $\mathbf{P}_\theta(\tilde{x}_i|x_{<i})$ for any $\tilde{x}_i \in \mathcal{V}$. By training on a large corpus of language texts, the LLMs provide the estimated probabilities.

B Proofs of Theorems

B.1 Proof of Theorem 1

Define the functions

$$\chi(x) := \lfloor x/10 \rfloor \text{ and } \zeta(x) := x \bmod 10, \text{ for } x \in \mathbb{N}.$$

Then $c_i = \zeta(a_i + b_i + c_{i-1}^{\chi}), \forall i$, and the carry $c_i^{\chi} = \chi(a_i + b_i + c_{i-1}^{\chi})$. For simplicity, assume $a_0 = b_0 = 0$.

We define three forms of approximation:

- *Strong form*: If $\mathbf{P}_\theta(\tilde{c} = c_i \mid a + b = c_{<i}) = 1$ for any $i \geq 1$. This means the model $\mathbf{P}_\theta(\cdot \mid a + b = c_{<i})$ can perfectly learn the function $c_i = \zeta(a_i + b_i + c_{i-1}^{\chi}), \forall i$.
- *Standard form*: If $c_i = \arg\max_{\tilde{c}} \mathbf{P}_\theta(\tilde{c} \mid a + b = c_{<i})$ for any $i \geq 1$. This means the model $\mathbf{P}_\theta(\cdot \mid a + b = c_{<i})$ can approximate the function $c_i = \zeta(a_i + b_i + c_{i-1}^{\chi}), \forall i$ with the highest probability.
- *Weak form*: If $\mathbf{P}_\theta(\tilde{c} = c_i \mid a + b = c_{<i}) > 0$ for any $i \geq 1$. This means the model $\mathbf{P}_\theta(\cdot \mid a + b = c_{<i})$ can approximate the function $c_i = \zeta(a_i + b_i + c_{i-1}^{\chi}), \forall i$ with a non-zero probability.

In the following, we will use the standard form to demonstrate out-of-distribution (OOD) generalization. When training a Transformer model on $\mathcal{D}_n$-addition using absolute positional embedding (APE), the learned model approximates the function at each position of c:

$$\mathbf{P}_\theta(c_i \mid a_{\leq i}, b_{\leq i}) \to c_i = \zeta(a_i + b_i + c_{i-1}^{\chi}).$$

Case I: Downward OOD Domain

Let us consider the Downward OOD domain $\mathcal{D}_{<n}$ case. If $i < n$, the model trained on a sample dataset in $\mathcal{D}_n$ can at least approximate the function c_i in the standard form. If $i = n$,

$$\mathbf{P}_\theta(c_n \mid a_{\leq n}, b_{\leq n}) \to c_n = \zeta(a_n + b_n + c_{n-1}^{\chi})$$

for every $a_n \vee b_n \geq 1$ except the case $a_n = b_n = 0$ simultaneously. If $i = n + 1$,

$$\mathbf{P}_\theta(c_{n+1} \mid a_{\leq n+1}, b_{\leq n+1}) \to c_{n+1} = c_n^{\chi} \in \{0, 1\}$$

for every pair (a_n, b_n) with $a_n \vee b_n \geq 1$ and $a_{n+1} = b_{n+1} = 0$. In the case where $a_n = b_n = 0$, the conditions for both $i = n$ and $i = n + 1$ necessitate OOD generalization. Since the model has been trained to approximate c_n accurately for $a_n \vee b_n \geq 1$, it has learned the function for the carry-over mechanism properly. When $a_n = b_n = 0$, the digit c_n purely depends on the carry from the previous position. For $i = n + 1$, the carry c_n^{χ} is correctly learned such that it maps $\{0, 1\}$ depending on whether there was a carry from the n-th digit. With $a_n = b_n = 0$, the model correctly sets $c_{n+1} = 0$. The training on $\mathcal{D}_n$ includes all possible carry scenarios and digit summations for $a_n, b_n \in \{0, \ldots, 9\}$. The zero cases are naturally included in the learned patterns[10]. For $i \geq n + 2$,

$$\mathbf{P}_\theta(c_i \mid a_{\leq i}, b_{\leq i}) \to c_i = \zeta(a_i + b_i + c_{i-1}^{\chi}) \equiv 0,$$

since $a_i = b_i \equiv 0$ for any $(a, b) \in \mathcal{D}_n$ with $i \geq n + 1$. Thus, the model $\mathbf{P}_\theta$ can approximate the function of c at every position for the downward OOD domain $\mathcal{D}_{<n}$.

Case II: Upward OOD Domain

Consider the Upward OOD domain $\mathcal{D}_{>n}$ case. If $i \leq n$, the analysis remains the same as above. The learned model $\mathbf{P}_\theta$ can predict the correct numbers at these positions. However, when $i = n + 1$,

$$\mathbf{P}_\theta(c_{n+1} \mid a_{\leq n+1}, b_{\leq n+1}) \to c_{n+1} = c_n^{\chi} \in \{0, 1\}$$

for every pair (a_n, b_n) with $a_n \vee b_n \geq 1$ and $a_{n+1} = b_{n+1} = 0$. Note that for inference in the OOD domain $\mathcal{D}_{>n}$, the model needs to predict each sample with (a_{n+1}, b_{n+1}) at least for every $a_{n+1} \vee b_{n+1} \geq 1$. However, the support of probability measure learned by the model $\mathbf{P}_\theta$ is $\operatorname{supp} \mathbf{P}_\theta = \{0, 1\}$. For the model to predict c_{n+1} correctly even in the weak form, the support should be $\operatorname{supp} \mathbf{P}_\theta = \{0, 1, \cdots, 9\}$. This indicates that the model $\mathbf{P}_\theta$ cannot predict the number at position $n + 1$. Additionally, the learned probability $\mathbf{P}_\theta(c_{n+1} \mid a_{\leq n+1}, b_{\leq n+1})$ is actually independent of

[10] If the training dataset has significant gaps, such as when a model is trained on n-digit addition but only with $a_n, b_n \geq n_0$ (e.g., $a_n, b_n \geq 6$), it means the model never encounters pairs where both $a_n < 6$ and $b_n < 6$. While the digit-wise addition and carry mechanisms for positions 1 through $n - 1$ are learned correctly, since these positions involve a full range of digit pairs during training, the model fails to learn proper behavior for the n-th and $(n + 1)$-th positions. Specifically, for these positions, the model will not encounter any pairs where both digits are simultaneously less than 6. In this scenario, $\zeta(a_n + b_n) \in \{2, 3, \ldots, 8\}$ (missing the digits 0, 1, 9), and $c_n^{\chi} \equiv 1$ (missing the digit 0). Consequently, the training dataset lacks complete coverage of all possible carry scenarios and digit summations. This substantial gap negatively affects the model's ability to handle these edge situations. Thus, the final learned model cannot generalize to the OOD domain $\mathcal{D}_{<n}$. Specifically, you will observe that the $(n + 1)$-th position value $c_{n+1} \equiv 1$ for all samples in $\mathcal{D}_{<n}$.

(a_{n+1}, b_{n+1}). For $i \geq n+2$,

$$\mathbf{P}_\theta(c_i \mid a_{\leq i}, b_{\leq i}) \to c_i \equiv 0,$$

since $a_i = b_i \equiv 0$ for any $(a,b) \in \mathcal{D}_n$ with $i \geq n+1$. This means that the learned model maps all inputs to zeros for positions $i \geq n+2$. If the model could predict the numbers at positions $i \geq n+2$, the requirement even in the weak form is that at least $\{0,1\} \subset \operatorname{supp} \mathbf{P}_\theta(c_i \mid \cdots)$. This contradicts $\operatorname{supp} \mathbf{P}_\theta(c_i \mid \cdots) = \{0\}$. Combining the above analysis, we conclude that the learned model $\mathbf{P}_\theta$ cannot solve the problems in the OOD domain $\mathcal{D}_{>n}$ but instead outputs the result $(a \mod 10^n) + (b \mod 10^n)$ for every sample in $\mathcal{D}_{>n}$. □

B.2 Proof of Theorem 3.

We begin by noting the key property that under the assumption of relative positional embedding (RPE), the Transformer model possesses a form of *translation invariance*. This property implies that the prediction at any position i is invariant to the shift of the entire sequence, as long as the relative distances between positions remain unchanged.

Special Case:

The translation invariance property is mathematically expressed as:

$$\begin{aligned}\mathbf{P}_\theta(c_i \mid a_{\leq i}, b_{\leq i}) &= \mathbf{P}_\theta(c_i \mid a_{i-1}, a_i, b_{i-1}, b_i) \\ &= \mathbf{P}_\theta(c_{i+j} \mid a_{i+j-1}, a_{i+j}, b_{i+j-1}, b_{i+j}),\end{aligned}$$

for any $i, j \in \mathbb{N}$, provided that $a_{i-1} + b_{i-1} \neq 9$.

This translation invariance arises when the carry c^{χ}_{i-1} is determined by the previous digits a_{i-1} and b_{i-1}, and thus does not depend on any global position or the absolute positions of the digits in the sequence. In fact, we have:

$$c_i = \zeta(a_i + b_i + c^{\chi}_{i-1}),$$

where $c^{\chi}_{i-1} = \chi(a_{i-1} + b_{i-1})$, provided that $a_{i-1} + b_{i-1} \neq 9$.

General Case:

The failure of the above translation invariance property occurs when the carry c^{χ}_{i-1} is influenced by more digits beyond a_{i-1} and b_{i-1}. A generalized translation invariance property should be used, i.e.,

$$\begin{aligned}&\mathbf{P}_\theta(c_i \mid a_{\leq i}, b_{\leq i}) \\ &= \mathbf{P}_\theta(c_i \mid a_{i-n+1}, \cdots, a_i, b_{i-n+1}, \cdots, b_i) \\ &= \mathbf{P}_\theta(c_{i+j} \mid a_{i+j-n+1}, \cdots, a_{i+j}, b_{i+j-n+1}, \cdots, b_{i+j}).\end{aligned}$$

The failure for above formula happens when carry propagation extends beyond the maximum length n seen during training, i.e., when the carry is influenced by positions greater than n. The case only happens when $a_{i-k} + b_{i-k} = 9$ for all $k = 1, \cdots, n-1$.

The probability of this failure is quite small. Specifically, it is less than $1/10^{n-1}$, because the probability of the carry propagating beyond the maximum digit position n (in a dataset where all digits are restricted to the range 0-9) diminishes exponentially as the length of the sequence increases. This ensures that such failures are rare, especially for large n.

For the upward OOD domain $\mathcal{D}_{>n}$, the model faces the challenge of predicting the carry propagation for positions $i > n$. However, since the model and addition satisfies translation invariance, this ensures that the model can handle longer sequences by effectively "folding" them into smaller, equivalent-length sequences with the same relative distances between digits, with only a probability of failure in the upward domain being less than $1/10^{n-1}$. □

Remarks on APE and RPE: APE encodes positional information based on the absolute positions of tokens in a sequence. This approach can limit a model's ability to generalize to sequences of different lengths or to handle out-of-distribution scenarios effectively. In contrast, RPE captures translation-invariant positional dependencies by encoding the relative distances between tokens. This method allows the model to focus on the relationships between tokens regardless of their absolute positions, enhancing its ability to generalize across varying sequence lengths and to better understand contextual relationships. Consequently, RPE is more robust and adaptable in the addition context compared to APE. Our theoretical framework can explain the addition-based experimental findings reported in the following references: Jelassi et al. (2023), Xu et al. (2024), Duan et al. (2024), and McLeish et al. (2024).

B.3 Proof Sketch of Theorem 4.

We will initially focus on the scenario where $p = 10^m$, and subsequently explore the general case where p is a divisor of 10^m.

Case I: Let us revisit the equation for modular addition, which states that $\overline{c}^p = \overline{\overline{a+b}^p + \overline{b}^p}^p$. The above equation shows that for the case $p = 10^m$, the digits in positions higher than m in numbers a and b do not affect the result $\overline{c}^p$; only the digits in

positions m and lower have an impact. Furthermore, we have $\overline{c}^p = (\overline{c}_1^p, \overline{c}_2^p, \cdots, \overline{c}_m^p) = (c_1, c_2, \cdots, c_m)$, where $c = a+b$. A model trained on $\mathcal{D}_n$ is capable of approximating the digits at positions ranging from 1 to m. This can be expressed as:

$$\mathbf{P}_\theta(\overline{c}_i^p \mid a_{\le i}, b_{\le i}) \to \overline{c}_i^p = \zeta(a_i + b_i + c_{i-1}^\chi),$$

for $i = 1, \cdots, m$. All these functions are learned directly from the training data without the need for out-of-distribution (OOD) generalization if $m < n$, while $m = n$, only the n-th term $\overline{c}_n^p$ need OOD generalization. For $i > m$, the probability $\mathbf{P}_\theta(\overline{c}_i^p \mid \cdot) \equiv 0$. The aforementioned conclusions apply to both domains $\mathcal{D}_{<n}$ and $\mathcal{D}_{>n}$.

Case II: Consider the case where p is a divisor of 10^m. Since we have $\overline{c}^p = \overline{a+b}^p = \overline{a+b}^{\overline{10^m}^p}$, the result $\overline{c}^p$ is indeed not influenced by the digits in positions higher than m in numbers a and b. If let m be the minimum number which the m-th power of 10 can be divided by the modulus p, i.e. $m = \arg\min\{\tilde{m} : p \mid 10^{\tilde{m}}\}$, the model approximates the function at each position i:

$$\mathbf{P}_\theta(\overline{c}_i^p \mid a_{\le m}, b_{\le m}) \to \overline{c}_i^p = f_i^p(a_{\le m}, b_{\le m}),$$

for $i = 1, \cdots, m$, where f_i^p is the function for $\overline{c}_i^p$ at the position i. As an aside, it is worth noting that in the case described above, the function is more intricate than standard addition or modular addition with a modulus that divides a power of 10. These functions generally rely on the digits at all positions of the numbers a and b, from position 1 through m. All these functions can be learned directly from the training data without the need for OOD generalization when training on $\mathcal{D}_n$ ($n \ge m$) except the term $\overline{c}_n^p$. □

B.4 Proof Sketch of Theorem 5.

In this case, the model approximates the function for each position i as follows when training on $\mathcal{D}_n$:

$$\mathbf{P}_\theta(\overline{c}_i^p \mid a_{\le n}, b_{\le n}) \to \overline{c}_i^p = f_i^p(a_{\le n}, b_{\le n}),$$

for $i = 1, \cdots, n$, where f_i^p represents the function for $\overline{c}_i^p$ at position i. Generally, the function $f^p(a,b) = (a+b) - \lfloor (a+b)/p \rfloor p$. Each digit f_i^p depends on all positions of a and b. If the model is trained on $\mathcal{D}_n$, the aforementioned probabilities have been trained exclusively on scenarios where $a_n \vee b_n \ge 1$. The case where $a_n = b_n = 0$ requires OOD generalization for samples on the downward domain $\mathcal{D}_{<n}$. This can be addressed by aligning with the model trained on the domain containing $\mathcal{D}_{n-1,n}$. If the model is trained on the dataset $\mathcal{D}_{n-1,n}$, which includes the case where $a_n = b_n = 0$, it learns the relevant patterns directly from the training data without the need for OOD generalization on the domain $\mathcal{D}_{<n}$. However, the model typically struggles to generalize to the upward domain $\mathcal{D}_{>n}$. This is because the model is expected to approximate the functions $f^p(a,b) = \overline{a+b}^p$, which consider all digits of a and b. Since the model is trained on $\mathcal{D}_n$, it learns the function $\hat{f}^p(a,b) = \overline{\overline{a}^{10^n} + \overline{b}^{10^n}}^p$, which is independent of the positions $i > n$ of the numbers a and b.

OOD Test Accuracy Analysis for Longer Length. For the model's output to be correct, it must satisfy the condition $\overline{a+b}^p = \overline{\overline{a}^{10^n} + \overline{b}^{10^n}}^p$. This requirement also provides us with a method to estimate the OOD test accuracy on the upward domain $\mathcal{D}_{>n}$.

Let $H_n = \overline{a}^{10^n} + \overline{b}^{10^n}$, and $R_n = (a+b) - H_n$. The OOD generalization error is then

$$f^p(a,b) - \hat{f}^p(a,b) = R_n - (\lfloor (a+b)/p \rfloor - \lfloor H_n/p \rfloor)\, p.$$

Denote $\varepsilon_n^R := \frac{R_n}{p} - \lfloor \frac{R_n}{p} \rfloor \in [0,1)$ and $\varepsilon_n^H := \frac{H_n}{p} - \lfloor \frac{H_n}{p} \rfloor \in [0,1)$. Then

$$\begin{aligned} & f^p(a,b) - \hat{f}^p(a,b) \\ &= (R_n/p - \lfloor (R_n + H_n)/p \rfloor + \lfloor H_n/p \rfloor)p \\ &= (\varepsilon_n^R - \lfloor \varepsilon_n^R + \varepsilon_n^H \rfloor)p. \end{aligned}$$

That is,

$$\begin{aligned} & f^p(a,b) - \hat{f}^p(a,b) \\ &= \begin{cases} \varepsilon_n^R p \ge 0, & \text{if } \varepsilon_n^R + \varepsilon_n^H \in [0,1) \\ (\varepsilon_n^R - 1)p < 0, & \text{if } \varepsilon_n^R + \varepsilon_n^H \in [1,2) \end{cases}. \end{aligned}$$

For the special case where $\varepsilon_n^R = 0$ (i.e. R_n is divisible by p), we have $\hat{f}^p(a,b) = f^p(a,b)$. This implies that the OOD test accuracy for a finite OOD test dataset may be greater than 0.

The OOD test accuracy on the domain (denote as $\widetilde{\mathcal{D}}_{n_{test}}$ and $n_{test} > n$) in which the length of a, b are both n_{test} is $\mathrm{Acc}(p, n, n_{\text{test}}) = \frac{\#\{(a,b) \in \widetilde{\mathcal{D}}_{n_{test}} : \varepsilon_n^R = 0\}}{\#\widetilde{\mathcal{D}}_{n_{test}}}$. This can be calculated by counting the number of R_n divisible by p in this domain. The theoretical test accuracy on $\widetilde{\mathcal{D}}_{n_{\text{test}}}$ is given by $\mathrm{Acc}(p, n, n_{\text{test}}) \approx \frac{1}{p'}$ if $n_{\text{test}} \ge n + \log_{10}(p'/2 + 1)$, otherwise 0. The proof can be found in the following section on test accuracy analysis. □

Let's consider some examples. For $p = 151$ and $n = 4$, since $\gcd(151, 10^n) \equiv 1$, the test accuracy is $\mathrm{Acc}(151, 4, n_{\text{test}}) = \frac{1}{151} \approx 0.66\%$ if $n_{\text{test}} \geq 6$, but 0 when $n_{\text{test}} = 5$. For $p = 201$ and $n = 4$, the test accuracy is $\mathrm{Acc}(201, 4, n_{\text{test}}) = \frac{1}{201} \approx 0.5\%$ if $n_{\text{test}} \geq 7$, but 0 when $n_{\text{test}} = 5, 6$. Another example is $p = 150$ and $n = 4$, where the greatest common divisor is $\gcd(150, 10^4) = 50$ and $p' = 3$, resulting in a test accuracy of $\mathrm{Acc}(150, 4, n_{\text{test}}) = \frac{50}{150} \approx 33.3\%$ for all $n_{\text{test}} \geq 5$. In the extreme case where p is a divisor of 10^n, the test accuracy $\mathrm{Acc}(p, n, n_{\text{test}}) \equiv 100\%$. This aligns with the results for the scenarios on the divisibility of a power of 10 by the modulus. All these findings are confirmed by our experimental analysis (see Table 2 and Table 6).

B.5 Proof Sketch of Theorem 6.

Given two natural numbers a and b, each represented by n-digit sequences $(a_1, a_2, \ldots, a_n)$ and $(b_1, b_2, \ldots, b_n)$, respectively, the product ab is expressed as a $2n$-digit number $c = (c_1, c_2, \ldots, c_{2n})$.

To express each digit c_i of the product c in terms of the digits of a and b, we need to understand the multiplication task and how the digits interact. The product ab can be represented as:

$$\begin{aligned} ab &= \left(\sum_{i=1}^{n} a_i \cdot 10^{i-1}\right)\left(\sum_{j=1}^{n} b_j \cdot 10^{j-1}\right) \\ &= \sum_{i=1}^{n}\sum_{j=1}^{n} a_i b_j \cdot 10^{(i-1)+(j-1)}. \end{aligned}$$

This gives us a double sum where each term $a_i b_j$ contributes to a specific power of 10. To express the digit c_k (where $1 \leq k \leq 2n$) of the product, we need to collect all terms from the expansion that contribute to the 10^{k-1} place.

For c_k, we consider all pairs (i, j) such that $i + j - 2 = k - 1$, which simplifies to $i + j = k + 1$. Define that the raw sum c_k^R at the k-th position as follows:

$$c_k^R = \sum_{\substack{1 \leq i,j \leq n \\ i+j=k+1}} a_i b_j.$$

However, since this is a digital product and carries might affect higher places, the correct formulation needs to account for carries from previous steps. The process of digit-wise calculation and adjustment with carries are as follows:

1. Initialize carry $c_0^\chi = 0$.
2. Calculate the sum for each digit place:

$$S_i = c_i^R + c_{i-1}^\chi = \sum_{\substack{1 \leq i',j' \leq n \\ i'+j'=i+1}} a_{i'} b_{j'} + c_{i-1}^\chi,$$

where $a_{i'}$ and $b_{j'}$ are zeros if their indices are out of bounds.

3. Determine the digit and carry:

$$c_i = \zeta(S_i), \quad c_i^\chi = \chi(S_i).$$

Here, $\zeta(x) := x \bmod 10$ and $\chi(x) := \lfloor x/10 \rfloor$, for $x \in \mathbb{N}$. This recursive formula provides the digits of the product considering the carries correctly. Denote that $c_i = f_i(a_1, \cdots, a_{i \wedge n}, b_1, \cdots, b_{i \wedge n})$ for $i = 1, 2, \cdots, 2n$. A Transformer model $\mathbf{P}_\theta(c_i \mid a \times b = c_1 \cdots c_{i-1}) = \mathbf{P}_\theta(c_i \mid a_1, \cdots, a_{i \wedge n}, b_1, \cdots, b_{i \wedge n})$ will learn to approximate these functions f_i when given enough data and computation power.

Consider the longer length OOD domain $(a, b) \in \mathcal{D}_{>n}$. Let $\overline{a} = \overline{a}^{10^n}$ and $\overline{b} = \overline{b}^{10^n}$. The function learned by a Transformer model with absolute positional embeddings (APE) when trained with $(a, b) \in \mathcal{D}_{n-1,n}$ is then

$$\hat{f}(a, b) = \overline{a}^{10^n} \cdot \overline{b}^{10^n} = \overline{c} = (\overline{c}_1, \overline{c}_2, \cdots, \overline{c}_{2n}, 0, \cdots, 0)$$

with $\overline{c}_i = f_i(a_1, \cdots, a_{i \wedge n}, b_1, \cdots, b_{i \wedge n})$, $1 \leq i \leq 2n$, as all terms related to a_i, b_i for $i > n$ are discarded during the training process. If the true value of ab is c, then $\overline{c}_i = c_i$ for $1 \leq i \leq n$, but generally differs from c_i when $i > n$ since $\overline{c}_i$ neglects the contribution of higher terms (greater than n) of a and b.

Note that when a Transformer model is trained on domain $\mathcal{D}_n$, if $i < n$, the model learns the function $f_i(a_1, \cdots, a_{i \wedge n}, b_1, \cdots, b_{i \wedge n})$ directly from the training data. However, when $i \geq n$, the model learns the function $f_i(a_1, \cdots, a_n, b_1, \cdots, b_n)$ only for the case where $a_n \vee b_n \geq 1$. In the scenario where $a_n = b_n = 0$, the model requires OOD generalization. The training on $\mathcal{D}_n$ includes all possible carry scenarios and digit summations (here, we only need consider the units and tens digits of c_i^R and c_{i-1}^χ) for $a_n, b_n \in \{0, \ldots, 9\}$. The zero cases where $a_n = b_n = 0$ are naturally included in the learned patterns. □

B.6 Proof Sketch of Theorem 7.

The proof resembles the process for modular addition. Suppose $\overline{c}^p = \overline{ab}^p$. When p is a divisor of 10^m, we have $\overline{c}^p = \overline{\overline{ab}^{10^m p}}$. The value of $\overline{c}^p$ remains unaffected by the digits in positions beyond m in the numbers a and b. Now, let m be the smallest number such that the m-th power of 10 is divisible by the modulus p, i.e., $m = \arg\min\{\tilde{m} : p \mid 10^{\tilde{m}}\}$.

The model approximates the function for each position i as follows:

$$\mathbf{P}_\theta(\overline{c}_i^p \mid a_{\le m}, b_{\le m}) \to \overline{c}_i^p = f_i^p(a_{\le m}, b_{\le m}),$$

for $i = 1, \cdots, m$, where f_i^p represents the function for the i-th digit of $\overline{c}^p$. All these functions can be learned directly from the training data without the need for OOD generalization when training on $\mathcal{D}_n$ ($n \ge m$) except the term $\overline{c}_n^p$.

When p is not a divisor of 10^n and $p < 10^n$, the model approximates the function $\hat{f}^p(a,b) = \overline{a^{10^n} \times b^{10^n p}}$ at each position i.This is because the model has been trained on $\mathcal{D}_n$, which is agnostic to the digits in positions $i > n$ of the numbers a and b. □

C Remarks

Remarks on Theorem 1: The challenging aspect of model prediction in the downward OOD domain $\mathcal{D}_{<n}$ arises from the need to generalize the n-th and $(n+1)$-th positions in the result c when trained on $\mathcal{D}_n$. Specifically, these positions must be generalized to the scenario where $a_n = b_n = 0$. Through our experimental analysis, we confirmed that the positions n and $n+1$ are the last to be learned during the training process. An additional observation is that if the model is trained on the domain $\mathcal{D}_{n-1,n} := \mathcal{D}_{n-1} \cup \mathcal{D}_n$, the previously mentioned challenge is mitigated. This is because the case with $a_n = b_n = 0$ is already incorporated into the training dataset. Consequently, the positions n and $n+1$ do not require OOD generalization; instead, they are learned directly from the training data. We have also conducted experiments based on this training scheme and found that learning on the domain that includes $\mathcal{D}_{n-1,n}$ is significantly easier than learning on $\mathcal{D}_n$ alone.

Remark on Transformer models based on relative/abacus positional embedding: The standard addition benefits from the property of translation invariance, whereas modular addition or modular multiplication with a modulus p that does not divide 10^n lacks this property. Consequently, there is no apparent advantage to be gained from leveraging this characteristic.

D Difficulty for Learning Multiplication

Transition Invariance Property in Multiplication. The transition invariance property for multiplication refers to the idea that the position of digits in the multiplication process can be shifted or "transitioned" in a systematic way that still respects the overall structure of multiplication. In the context of digit-wise multiplication, each digit c_i should be adjusted by the previous carry. This process is transition invariant because each digit's place calculation transitions in a smooth and systematic way from one digit place to the next, maintaining the structure of the multiplication.

Transformers can utilize properties like transition invariance to learn multiplication using proper positional embeddings such as relative or abacus PE. In fact, the structured nature of multiplication, especially when broken down into steps that involve digit-by-digit operations and carry propagation, aligns well with the capabilities of Transformer models to capture sequential dependencies and patterns. However, the most challenging aspect is computing the raw sums c_i^R at each position. Each c_i^R results from a sum of specific pairs of digits from the input sequences a and b. For a given c_i^R, the valid pairs (i', j') must satisfy $i' + j' = i + 1$. Identifying these pairs involves that (1) ensuring $1 \le i', j' \le n$, i.e., the indices must be within the bounds of the sequences. (2) For each i, determining which pairs contribute to c_i^R involves iterating through potential values of i' and j' and checking if their sum equals $i+1$. Digit multiplication depends on the positional significance of digits. Misalignment in positions can lead to incorrect contributions to the product. Therefore, positional encoding and accurate handling of positional values are necessary to ensure correct multiplication results. There are also efficiency considerations. Multiplication of large numbers involves many such sums. For large n, directly computing c_i^R for each i involves nested loops or checks, leading to a time complexity of $O(n^2)$ in the worst case. This poses a great difficulty for computing the raw sum c_i^R.

This challenge can be understood through the following analysis. Suppose the model is provided with Chain-of-Thought (CoT) style intermediate steps of multiplication as part of the training data. The CoT-like training data format is:

$$a \times b \to (c^R, c^\chi) \to c.$$

In digit-wise format, this is:

$$\begin{aligned}&(a_1, \cdots, a_n) \times (b_1, \cdots, b_n)\\ \to\ &(c_1^R, c_1^\chi, \cdots, c_{2n-1}^R, c_{2n-1}^\chi)\\ \to\ &(c_1, \cdots, c_{2n}).\end{aligned}$$

The conditional probability equation is then given by:

$$\begin{aligned}&\mathbf{P}_\theta(c_i \mid a_1,\cdots,a_{i\wedge n},b_1,\cdots,b_{i\wedge n})\\&=\mathbf{P}^\chi_\theta(c^\chi_{i-1} \mid a_1,\cdots,a_{(i-1)\wedge n},b_1,\cdots,b_{(i-1)\wedge n})\\&\times\mathbf{P}^R_\theta(c^R_i \mid a_1,\cdots,a_{i\wedge n},b_1,\cdots,b_{i\wedge n})\\&\times\mathbf{P}_\theta(c_i \mid c^R_i,c^\chi_{i-1}),\end{aligned}$$

and

$$\begin{aligned}&\mathbf{P}^\chi_\theta(c^\chi_i \mid a_1,\cdots,a_{i\wedge n},b_1,\cdots,b_{i\wedge n})\\&=\mathbf{P}^\chi_\theta(c^\chi_{i-1} \mid a_1,\cdots,a_{(i-1)\wedge n},b_1,\cdots,b_{(i-1)\wedge n})\\&\times\mathbf{P}^R_\theta(c^R_i \mid a_1,\cdots,a_{i\wedge n},b_1,\cdots,b_{i\wedge n})\\&\times\mathbf{P}^\chi_\theta(c^\chi_i \mid c^R_i,c^\chi_{i-1}).\end{aligned}$$

For the carry at the i-th position, we then have that

$$\begin{aligned}&\mathbf{P}^\chi_\theta(c^\chi_i \mid a_1,\cdots,a_{i\wedge n},b_1,\cdots,b_{i\wedge n})\\&=\prod_{j=1}^{i}\mathbf{P}^R_\theta(c^R_j \mid a_1,\cdots,a_{j\wedge n},b_1,\cdots,b_{j\wedge n})\\&\times\mathbf{P}^\chi_\theta(c^\chi_j \mid c^R_j,c^\chi_{j-1}).\end{aligned}$$

Note that $\mathbf{P}_\theta(c_i \mid c^R_i,c^\chi_{i-1})$ and $\mathbf{P}^\chi_\theta(c^\chi_i \mid c^R_i,c^\chi_{i-1})$ exhibit transition invariance. This could be handled by relative or abacus positional embedding. The difficulty lies in the computation of the raw sums $\mathbf{P}^R_\theta(c^R_i \mid a_1,\cdots,a_{i\wedge n},b_1,\cdots,b_{i\wedge n})$ even when using relative or abacus positional embedding.

Experiments on Transformer models using relative or abacus positional embeddings to learn multiplication have been presented in the literature. Jelassi et al. (2023) and McLeish et al. (2024) show that addition can successfully generalize to OOD regions with higher numerical digits, but multiplication has largely not succeeded. Our analysis provides insights into the difficulties behind generalizing to higher numerical digits, which helps us understand the reasons for the failure in learning multiplication.

E Theoretical OOD Test Accuracy for Modular Arithmetic

E.1 Theoretical OOD Test Accuracy for Modular Addition Learning

To derive an accurate analytic formula (in Theorem 5) for the OOD test accuracy on $\widetilde{\mathcal{D}}_m$ with $m > n$ when a Transformer model is trained on the domain $\mathcal{D}_n$, we must carefully count the valid pairs $(a,b) \in \widetilde{\mathcal{D}}_m$ that satisfy $\overline{a+b}^p = \overline{\overline{a}^{10^n} + \overline{b}^{10^n}}^p$.

Let $a = A\cdot 10^n + a_0$ and $b = B\cdot 10^n + b_0$, where A,B range from 1 to $10^{m-n}-1$ and a_0,b_0 range from 0 to 10^n-1. We require $a+b \equiv (a \mod 10^n + b \mod 10^n) \pmod p$, which simplifies to that

$$(A+B)\cdot 10^n \equiv 0 \pmod p.$$

Let $p' = \frac{p}{\gcd(p,10^n)}$. We are then left with the condition $(A+B) \equiv 0 \pmod{p'}$.

The number of such pairs is determined by the frequency of multiples of p' in the valid range. The total number of pairs (A,B) is $(10^{m-n}-1)^2$. There are $(10^{m-n}-1)$ valid values for A. For each A, the number of valid B values is determined by the number of multiples of p' in the range. That is, for each A, the number of valid B values is about $(10^{m-n}-1)/p'$. The test accuracy is the ratio of valid pairs, i.e. the number of valid pairs divided by the total number of pairs.

Note that for $m \geq n + \log_{10}(p'/2+1)$, the range $1 \leq A,B < 10^{m-n}$ must include at least one complete cycle of p' to ensure some pairs (A,B) satisfy $A+B \equiv 0 \pmod{p'}$. This condition ensures that the number of digits in A and B is large enough to cover a full period of p'. Otherwise, there exists no pair (A,B) for which $A+B \equiv 0 \pmod{p'}$.

The ultimate formula is as follows:

$$\begin{aligned}\mathrm{Acc}(p,n,m) &= \frac{\text{Number of Valid Pairs}}{\text{Total Number of Pairs}}\\&\approx \frac{(10^{m-n}-1)\cdot\left(\frac{10^{m-n}-1}{p'}\right)}{(10^{m-n}-1)^2} = \frac{1}{p'}\end{aligned}$$

for $m \geq n + \log_{10}(p'/2+1)$, otherwise 0.

Given that $p' = \frac{p}{\gcd(p,10^n)}$, we have that

$$\begin{aligned}&\mathrm{Acc}(p,n,m)\\&\approx\begin{cases}\frac{\gcd(p,10^n)}{p}, & \text{if } m \geq n+\log_{10}(p'/2+1)\\0, & \text{otherwise}\end{cases}.\end{aligned}$$

E.2 Theoretical OOD Test Accuracy for Modular Multiplication Learning

To count the valid pairs $(a,b) \in \widetilde{\mathcal{D}}_m$ that satisfy $a\times b \equiv ((a \mod 10^n)\times(b \mod 10^n)) \pmod p$, denote a and b can be written as $a = A\cdot 10^n + a_0$ and $b = B\cdot 10^n + b_0$, where A,B are the upper $(m-n)$-digit parts and a_0,b_0 are the lower n-digit parts. A,B range from 1 to $10^{m-n}-1$ (since they are non-zero leading digits). a_0,b_0 range from 0 to 10^n-1. We need

$$(A\cdot 10^n + a_0)\times(B\cdot 10^n + b_0) \equiv (a_0 \times b_0) \pmod p.$$

This simplifies to that

$$A \cdot B \cdot 10^{2n} + (A \cdot b_0 + B \cdot a_0) \cdot 10^n \equiv 0 \pmod{p}.$$

This further simplifies to that

$$A \cdot B \cdot 10^n + A \cdot b_0 + B \cdot a_0 \equiv 0 \pmod{p'},$$

$$p' = \frac{p}{\gcd(p, 10^n)}.$$

The theoretical closed expression for this problem is challenging to derive, but the numerical solution can be computed through an algorithmic program for small-scale cases.

F Model and Training Hyperparameters

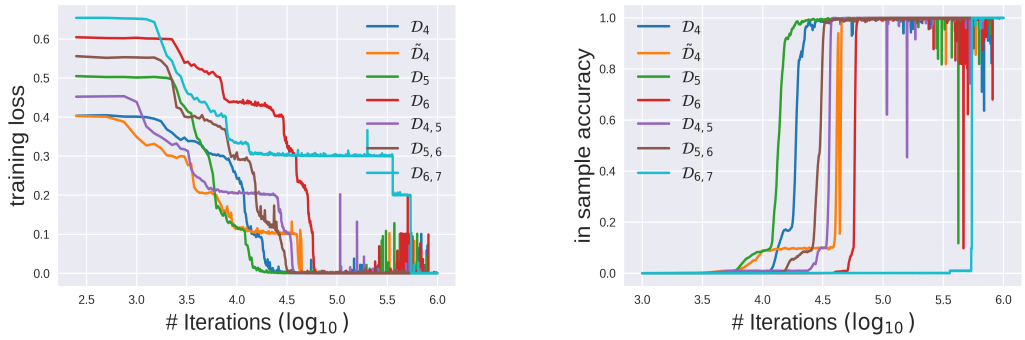
Detailed hyperparameters of the models and training are provided in Table 3.

Hyperparameter	NanoGPT	MicroGPT	MiniGPT
num layer	3	4	6
num head	3	4	6
dim embd	48	128	384
vocab size	16	16	16
context window	256	256	256
dropout prob	0.2	0.2	0.2
optimizer	AdamW	AdamW	AdamW
learning rate	0.001	0.001	0.001
betas	(0.9, 0.99)	(0.9, 0.99)	(0.9, 0.99)
weight decay	True	True	True
grad norm clip	1.0	1.0	1.0

Table 3: Hyperparameter for Arithmetic Operations Training

G Further Results

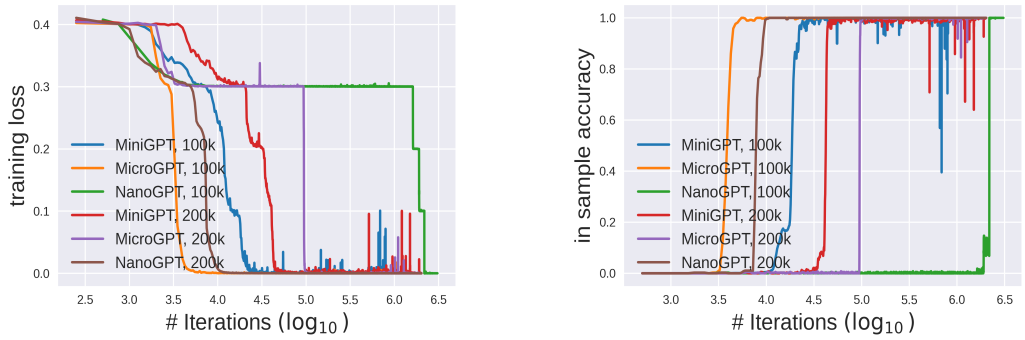
G.1 Further Results on Addition



Training Loss In Sample Accuracy

Figure 2: Training Loss & Out of Sample In-Distribution Test Accuracy on Addition

Note: $\mathcal{D}_i$ is trained on two number addition task with at least one number to be a i-digit number, $\mathcal{D}_{i,j}$ is trained on the combined training dataset of $\mathcal{D}_i$ and $\mathcal{D}_j$.



Training Loss In Sample Accuracy

Figure 3: Training Loss & Out of Sample In-Distribution Test Accuracy on Addition

Note: Robustness study on model and data scales. All models are trained on $\mathcal{D}_4$ where a and b are at least one to be a 4-digit number. NanoGPT represents the smallest model, with MicroGPT being of medium size and MiniGPT the largest. The designations "100k" and "200k" indicate that the training sets are 90% the size of 100,000 or 200,000, respectively.

G.1.1 How Digits are Learned During Training?

The experiment results depicted in Figure 5 illustrate the learning dynamics of each function c_i during the training of Transformer models, using DecisionTreeRegressor to approximate these functions. The R^2 values, which measure how well the model's predictions fit the actual data, indicate that the models effectively learn lower-order digits with high accuracy, achieving R^2 values close to 1. However, higher-order digits present more challenges, resulting in lower and less stable R^2 values. Furthermore, at the early stages of training, the models first learn the higher-order digits (with higher R^2 values) and then proceed to learn the lower-order digits.

From Figure 5, it is evident that the Transformer model trained on $\mathcal{D}_4$ initially focuses on learning the digits at positions 4 and 5 before addressing positions lower than 4. Here, position 6 is trivial since it always equals zero. The Transformer model trained on $\mathcal{D}_5$ first attempts to learn the digits at positions 5 and 6, then proceeds to positions lower than 5. The Transformer model trained on $\mathcal{D}_{4,5}$ starts by learning the digits at positions 4, 5, and 6, and then moves to positions lower than 4. In our theoretical analysis, the

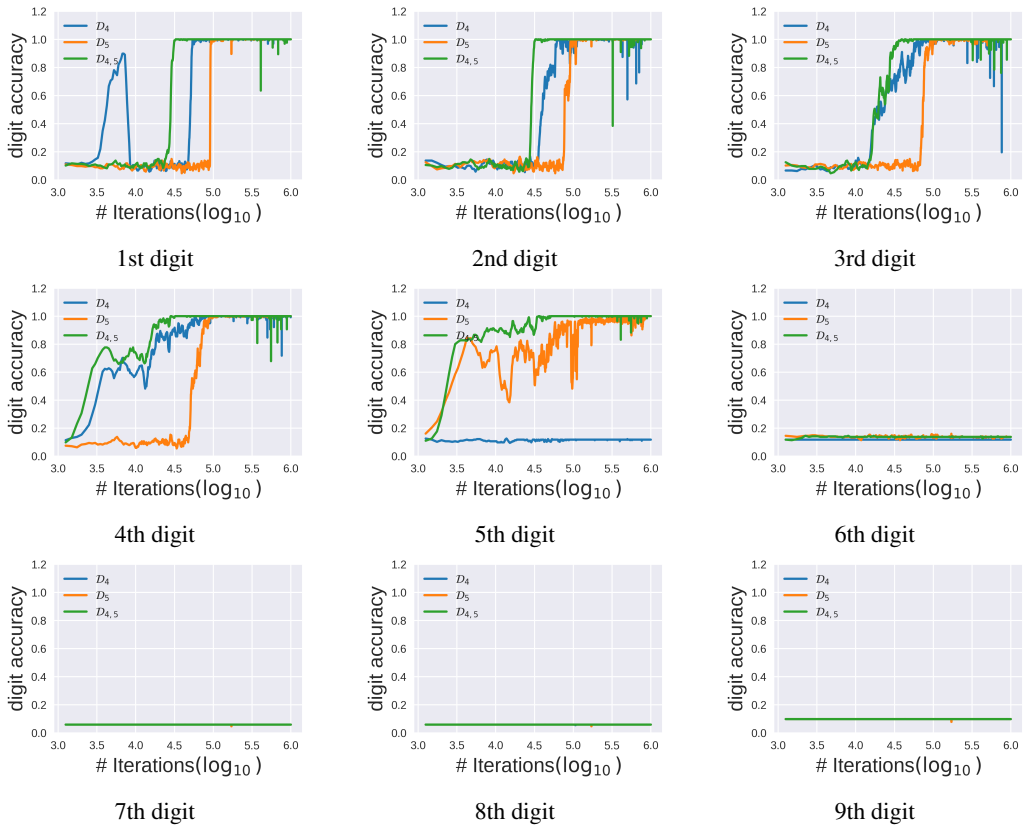


Figure 4: Digit-Wise Test Accuracy of Transformer Models with APE for Addition Tasks

Note: In this figure, we present the results of three different experiments using distinct training datasets. For all experiments, we employ the MiniGPT model equipped with a learned APE. In the legend, the label $\mathcal{D}_4$ indicates that the MiniGPT model is trained on a random sample from dataset $\mathcal{D}_4$. The label $\mathcal{D}_5$ denotes training on a random sample from dataset $\mathcal{D}_5$, while $\mathcal{D}_{4,5}$ signifies training on a combined subset from $\mathcal{D}_4$ and $\mathcal{D}_5$. Each subfigure illustrates the digit-wise test accuracy on a combined random sample sets $\mathcal{D}_{\leq 9}$ for these models throughout the training process.

Training Data	Test Accuracy (%) w.r.t. the Ground Truth on the Domain $\mathcal{D}_i$ 1	2	3	4	5	6	7	8	9
$\mathcal{D}_4$	100	100	100	100	0	0	0	0	0
$\widetilde{\mathcal{D}}_4$	100	100	72.6	100	0	0	0	0	0
$\mathcal{D}_5$	100	100	100	100	100	0	0	0	0
$\mathcal{D}_6$	100	100	100	100	100	100	0	0	0
$\mathcal{D}_{4,5}$	100	100	100	100	100	0	0	0	0
$\mathcal{D}_{5,6}$	100	100	100	100	100	100	0	0	0
$\mathcal{D}_{6,7}$	100	100	100	100	100	100	100	0	0

Table 4: Standard Addition: Test Accuracy w.r.t. the Ground Truth $f(a,b) = a+b$ on the Domain $\mathcal{D}_i$ for $i = 1, 2 \cdots, 9$. All models are instances of MiniGPT. The accuracy is tested on 10,000 random test samples (when $n > 2$), otherwise on the entire dataset. The outputs of models are generated using maximum probability sampling.

Training Data	Test Accuracy (%) w.r.t. the Modular Truth on the Domain $\mathcal{D}_i$ 1	2	3	4	5	6	7	8	9
$\mathcal{D}_4$	100	100	100	100	100	100	100	100	100
$\widetilde{\mathcal{D}}_4$	100	99.9	72.3	100	99.7	99.7	99.6	99.7	99.5
$\mathcal{D}_5$	100	100	100	100	100	100	100	100	100
$\mathcal{D}_6$	100	100	100	100	100	100	100	100	100
$\mathcal{D}_{4,5}$	100	100	100	100	100	100	100	100	100
$\mathcal{D}_{5,6}$	100	100	100	100	100	100	100	100	100
$\mathcal{D}_{6,7}$	100	100	100	100	100	100	100	100	100

Table 5: Standard Addition: Test Accuracy w.r.t. the Modular Truth $\hat{f}(a,b) = \overline{a}^{10^n} + \overline{b}^{10^n}$ on the Domain $\mathcal{D}_i$ for $i = 1, 2 \cdots, 9$. All models are instances of MiniGPT, and test methods are indicated as above.

most challenging parts are c_n and c_{n+1} when training the model with data in $\mathcal{D}_n$, since these positions never encounter $a_n = b_n = 0$ and require OOD generalization. The models prioritize learning the hardest positions first, followed by the easier positions in these experiments.

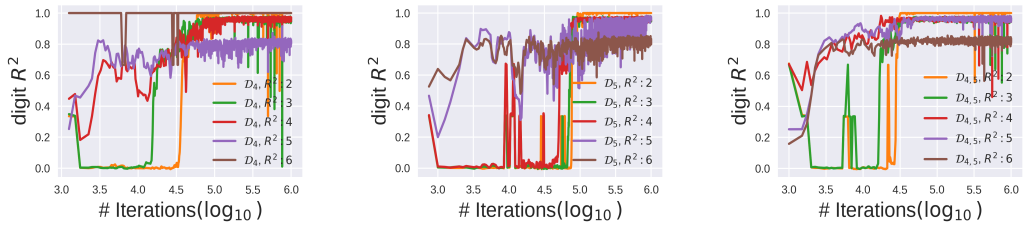
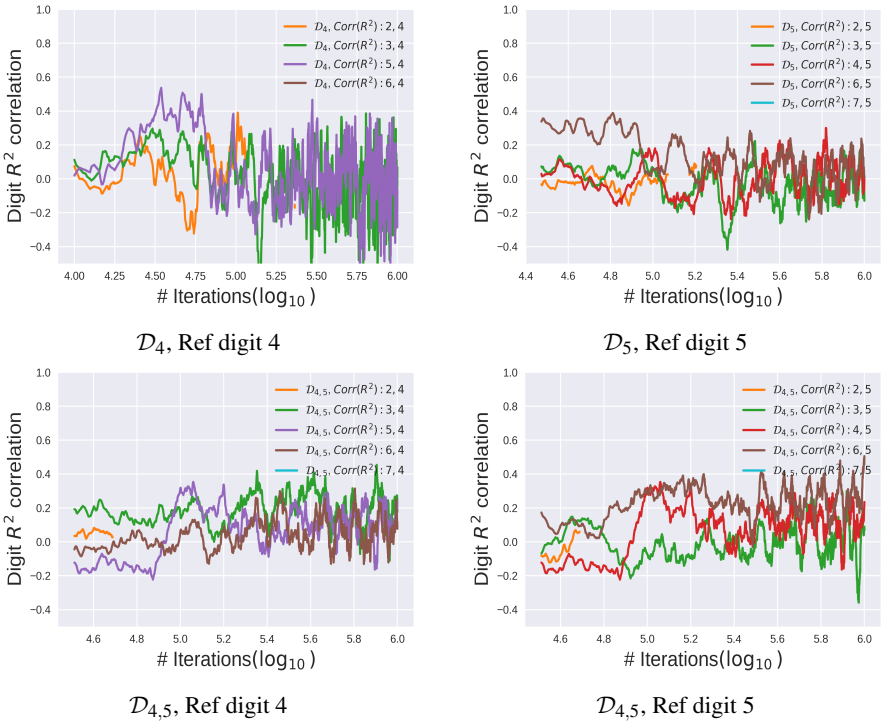


Figure 5: Learning Dynamics of Each Function $c_i = \zeta(a_i + b_i + c_{i-1}^{\chi})$ for Addition

Another notable result from the experiments is that the correlation of R^2 values between different digit pairs is around zero (see Figure 6 in this Appendix). This indicates that changes in the approximation for one position have little impact on other positions. This finding suggests that the Transformer model is flexible enough to handle different tokens independently, even though they share parameters.

G.1.2 Learning Addition Under Relative Positional Embedding



$\mathcal{D}_4$, Ref digit 4 $\mathcal{D}_5$, Ref digit 5

$\mathcal{D}_{4,5}$, Ref digit 4 $\mathcal{D}_{4,5}$, Ref digit 5

Figure 6: Correlation Between Digit Pairs of Learning c_i and c_j for Addition

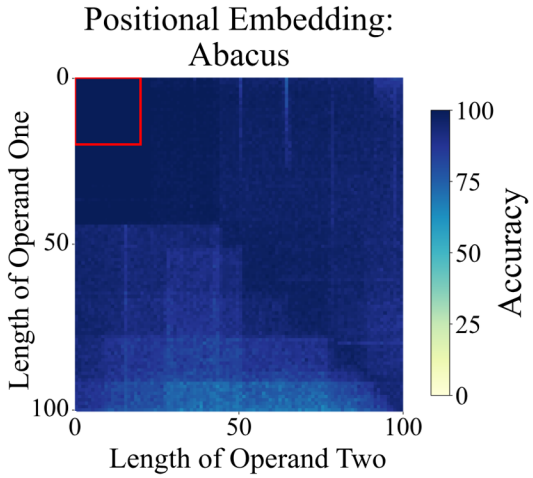
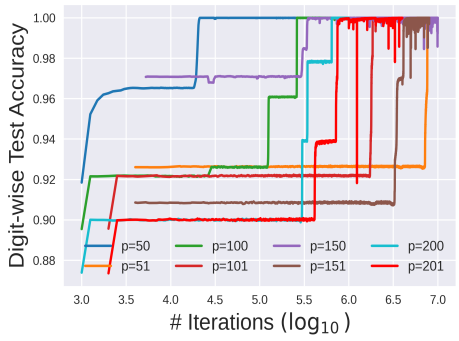


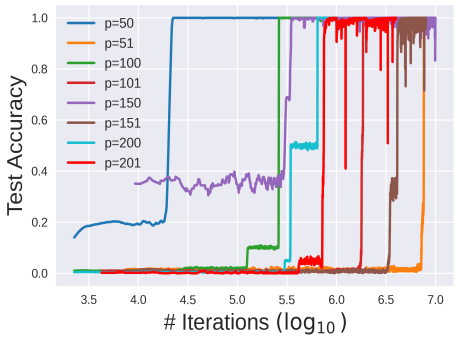
Figure 7: Test Accuracy on Addition When Training Short and Testing Long using a 16-Layer Transformer (Decoder only) Model with Abacus Positional Embedding.

Note: The image is extracted from the work McLeish et al. (2024) and is a screenshot of their Figure 1. The interior of the red box represents the training data domain $\mathcal{D}_{\leq 20}$. Code to reproduce the result can be found on the GitHub: `https://github.com/mcleish7/arithmetic`. The obtained result constitutes empirical evidence that validates our Theorem 3. The result is very clear. We will not repeat the same procedures. Use this as a reference in the present context.

G.2 Further Results on Modular Addition



Digit-wise Accuracy



In Sample Accuracy

Figure 8: Digit-wise In-Distribution Test Accuracy & Total Accuracy for Modular addition

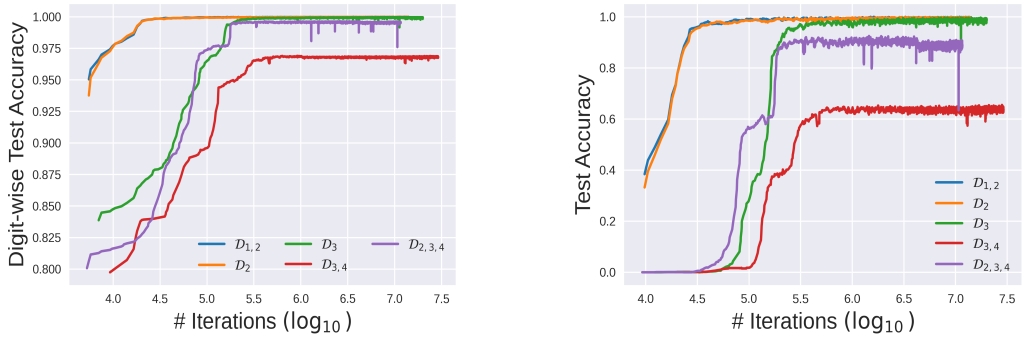
Note: These results correspond to modular addition tasks with the modulus p taking values in the set $\{50, 51, 100, 101, 150, 151, 200, 201\}$. Each model is trained using the MiniGPT model with a sample drawn from the domain $\mathcal{D}_4$ (except $p = 150$, which is on $\mathcal{D}_{\leq 4}$).

	Test Accuracy (%) w.r.t. the Modular Truth on the Domain $\widetilde{\mathcal{D}}_i$								
Modulus	1	2	3	4	5	6	7	8	9
$p = 50$	100	100	100	100	99.3	92.0	93.1	95.2	91.4
$p = 51$	100	98.5	99.9	99.3	95.1	94.4	92.6	91.3	92.4
$p = 100$	100	100	100	100	100	100	100	100	100
$p = 101$	100	100	100	100	100	100	100	100	100
$p = 150$	100	100	100	100	100	100	100	99.8	99.7
$p = 151$	100	99.9	99.9	100	99.9	99.7	99.6	99.1	99.2
$p = 200$	100	100	100	100	99.8	98.9	93.7	94.1	93.5
$p = 201$	100	100	99.9	99.9	96.4	96.6	95.7	90.4	91.2

Table 6: Modular Addition: Test Accuracy w.r.t. the Modular Truth $\hat{f}^p(a,b) = \overline{\overline{a}^{10^n} + \overline{b}^{10^n}}^p$ on the Domain $\widetilde{\mathcal{D}}_i$ for $i = 1, 2 \cdots, 9$.

Note: All the Transformer models in above experiments are instances of MiniGPT, which have been trained on a random sample drawn from $\mathcal{D}_4$ (except $p = 150$). The accuracy is tested on 10,000 random test samples (when $n > 2$), otherwise on the entire dataset. The outputs of models are generated using maximum probability sampling.

G.3 Further Results on Multiplication



Digit-wise Accuracy In Sample Accuracy

Figure 9: Digit-wise In-Distribution Test Accuracy & Total Accuracy for Multiplication

Note: These results correspond to multiplication tasks. The models trained on $\mathcal{D}_{1,2}$ and $\mathcal{D}_2$ are instances of MicroGPT, while others are of MiniGPT.

	Test Accuracy (%) w.r.t. the Ground Truth on $\mathcal{D}_i$								
Training Data	1	2	3	4	5	6	7	8	9
$\mathcal{D}_{1,2}$	100	100	0.1	0	0	0	0	0	0
$\mathcal{D}_2$	80.0	99.4	0.1	0	0	0	0	0	0
$\mathcal{D}_3$	100	96.4	99.0	0	0	0	0	0	0
$\mathcal{D}_{2,3,4}$	100	100	98.9	80.5	0	0	0	0	0

Table 7: Standard Multiplication: Test Accuracy w.r.t. the Ground Truth $f(a,b) = a \cdot b$ on the Domain $\mathcal{D}_i$ for $i = 1, 2 \cdots, 9$. The models trained on $\mathcal{D}_{1,2}$ and $\mathcal{D}_2$ are instances of MicroGPT, while others are of MiniGPT. The accuracy is tested on 10,000 random test samples (when $n > 2$), otherwise on the entire dataset. The outputs of models are generated using maximum probability sampling.

	Test Accuracy (%) w.r.t. the Modular Truth on $\mathcal{D}_i$								
Training Data	1	2	3	4	5	6	7	8	9
$\mathcal{D}_{1,2}$	100	99.9	93.0	90.1	86.0	82.6	80.6	78.2	77.7
$\mathcal{D}_2$	85.0	99.4	98.1	96.7	89.0	88.9	88.4	89.8	88.7
$\mathcal{D}_3$	100	96.2	98.8	98.9	99.0	97.9	97.9	97.2	97.1
$\mathcal{D}_{2,3,4}$	100	100	98.9	81.0	75.6	76.2	73.8	67.5	66.9

Table 8: Standard Multiplication: Test Accuracy w.r.t. the Modular Truth $\hat{f}(a,b) = \overline{a}^{10^n} \cdot \overline{b}^{10^n}$ on the Domain $\mathcal{D}_i$ for $i = 1, 2 \cdots, 9$. The models and test methods are indicated as above.

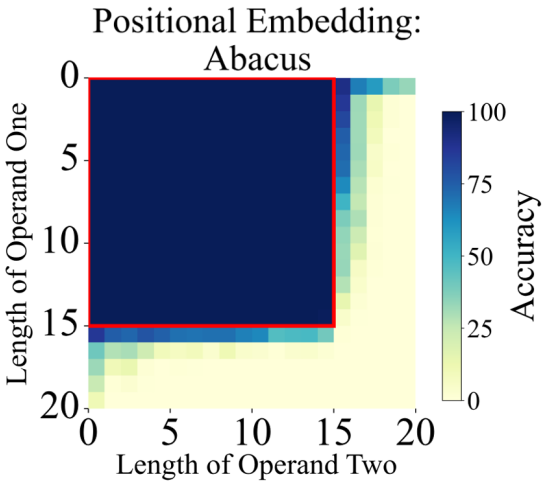
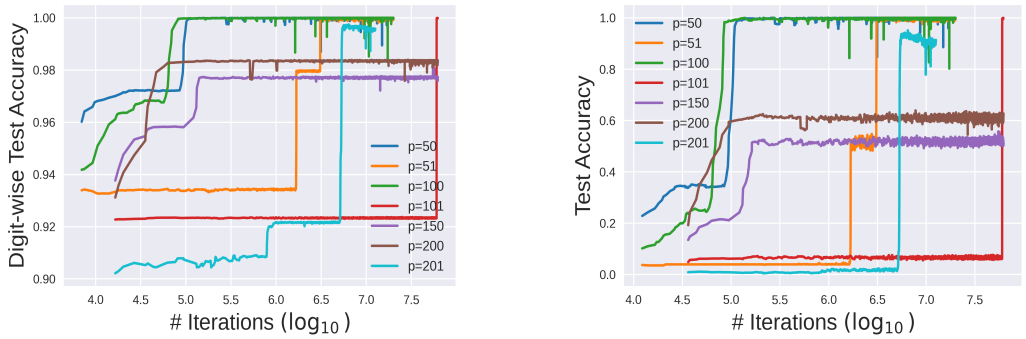


Figure 10: Test Accuracy on Multiplication When Training Short and Testing Long using a Looped Transformer Models with Abacus Positional Embedding.

Note: The image is extracted from the work McLeish et al. (2024) and is a screenshot of their Figure 5. The interior of the red box represents the training data domain $\mathcal{D}_{\leq 15}$.

G.4 Further Results on Modular Multiplication



Digit-wise Accuracy

In Sample Accuracy

Figure 11: Digit-wise In-Distribution Test Accuracy & Total Accuracy for Modular Multiplication

Note: These results correspond to modular multiplication tasks. The models are instances of MiniGPT and trained on $\mathcal{D}_3$.

Modulus	Test Accuracy (%) w.r.t. the Ground Truth on the Domain $\widetilde{\mathcal{D}}_i$ 1	2	3	4	5	6	7	8	9	Theor. Acc.
$p = 50$	100	100	100	100	100	100	100	100	100	100
$p = 51$	100	100	99.7	2.6	2.5	2.8	2.4	2.5	3.2	2.4
$p = 100$	100	100	100	100	100	100	100	100	100	100
$p = 101$	100	100	100	1.1	1.0	1.2	0.9	1.1	1.0	1.0
$p = 150$	30.0	56.4	55.5	46.9	46.5	46.3	47.4	46.9	47.0	40.8
$p = 200$	100	63.3	61.8	62.1	62.6	62.9	62.4	61.7	62.6	100
$p = 201$	80.0	78.3	92.2	0.7	0.6	0.5	0.6	0.6	0.6	0.6

Table 9: Modular Multiplication: Test Accuracy w.r.t. the Ground Truth $f^p(a,b) = \overline{a \cdot b}^p$ on $\widetilde{\mathcal{D}}_i$

Note: All the Transformer models in above experiments are instances of MiniGPT, which have been trained on a random sample drawn from $\mathcal{D}_3$. The accuracy is tested on 10,000 random test samples (when $i > 2$), otherwise on the entire dataset. The outputs of models are generated using maximum probability sampling. When $p = 150$ and $p = 200$, there is a significant difference between the experimental accuracy and the theoretical accuracy, which is *due to the fact that these two models have not yet achieved sufficient training on the training set, or in other words, they are under-trained*. This can be observed from the test accuracy in columns 1, 2, and 3 of the table above.

Training Data	Test Accuracy (%) w.r.t. the Modular Truth on $\widetilde{\mathcal{D}}_i$ 1	2	3	4	5	6	7	8	9
$p = 50$	100	100	100	100	100	100	100	100	100
$p = 51$	100	100	99.7	99.8	98.4	84.4	81.9	68.6	57.2
$p = 100$	100	100	100	100	100	100	100	100	100
$p = 101$	100	100	100	86.6	73.6	71.7	68.1	65.7	54.5
$p = 150$	42.0	55.7	56.0	51.0	51.2	50.0	50.0	50.3	50.1
$p = 200$	100	62.6	62.2	62.7	62.3	62.4	62.7	62.3	61.9
$p = 201$	71.0	79.5	92.1	90.9	90.7	90.5	88.7	87.9	85.0

Table 10: Modular Multiplication: Test Accuracy w.r.t. the Modular Truth $\hat{f}^p(a,b) = \overline{\bar{a}^{10^n} \cdot \bar{b}^{10^n}}^p$ on the Domain $\widetilde{\mathcal{D}}_i$ for $i = 1, 2 \cdots, 9$. The models and test methods are as indicated in the above table.