It Ain't That Bad: Understanding the Mysterious Performance Drop in OOD Generalization for Generative Transformer Models

Xingcheng Xu[1], **Zihao Pan**[2], **Haipeng Zhang**[2*] and **Yanqing Yang**[1,3*]
[1]Shanghai Artificial Intelligence Laboratory
[2]ShanghaiTech University
[3]Fudan University
xingcheng.xu18@gmail.com, {panzh,zhanghp}@shanghaitech.edu.cn, yanqingyang@fudan.edu.cn

Abstract

Large language models (LLMs) have achieved remarkable proficiency on solving diverse problems. However, their generalization ability is not always satisfying and the generalization problem is common for generative transformer models in general. Researchers take basic mathematical tasks like n-digit addition or multiplication as important perspectives for investigating their generalization behaviors. It is observed that when training models on n-digit operations (e.g., additions) in which both input operands are n-digit in length, models generalize successfully on unseen n-digit inputs (in-distribution (ID) generalization), but fail miserably on longer, unseen cases (out-of-distribution (OOD) generalization). We bring this unexplained performance drop into attention and ask whether there is systematic OOD generalization. Towards understanding LLMs, we train various smaller language models which may share the same underlying mechanism. We discover that the strong ID generalization stems from structured representations, while behind the unsatisfying OOD performance, the models still exhibit clear learned algebraic structures. Specifically, these models map unseen OOD inputs to outputs with learned equivalence relations in the ID domain, which we call the *equivalence generalization*. These findings deepen our knowledge regarding the generalizability of generative models including LLMs, and provide insights into potential avenues for improvement.

1 Introduction

Large language models (LLMs) such as ChatGPT [Ouyang *et al.*, 2022], GPT-4 [OpenAI, 2023], Claude [Anthropic, 2023], PaLM [Chowdhery *et al.*, 2023], Llama [Touvron *et al.*, 2023a,b] have exhibited remarkable advancements across diverse domains, prominently in natural language processing (NLP). The LLMs have demonstrated exceptional versatility, showcasing profound efficacy in tackling a myriad of tasks, ranging from natural language challenges to code translation, mathematical reasoning, and more [Bubeck *et al.*, 2023; Trummer, 2022; Zong and Krishnamachari, 2023]. Although these accomplishments are undoubtedly impressive, the generalization ability of LLMs and generative transformer models in general is not fully understood and not always satisfactory in issues such as natural language understanding [Bender *et al.*, 2021], and mathematical reasoning [Anil *et al.*, 2022].

Given the complexity of natural language tasks and the black-box nature of these models, researchers view basic mathematical tasks such as n-digit addition or multiplication as valuable avenues for gaining insights into their generalization behaviors [Lee *et al.*, 2023; Anil *et al.*, 2022]. Among them, many have observed an interesting yet mysterious phenomenon when training on n-digit operations [Brown *et al.*, 2020; Anil *et al.*, 2022; Jelassi *et al.*, 2023]. In cases where both input operands are n-digit long, the models demonstrate excellent generalization on unseen n-digit inputs. However, they unexpectedly and miserably struggle when faced with longer, unseen cases (inputs with more than n digits). For instance, when trained with operations like $349 + 705 = 1054$, the model would perform well on unseen input $350 + 705$. But when the inputs are $1349 + 2705$ which are longer in digits, the model gives a wrong answer. This creates a clear distinction between the former, known as *in-distribution (ID) generalization*, and the latter, termed *out-of-distribution (OOD) generalization*.

Seeking to bridge this generalization gap, scholars have undertaken various efforts to enhance OOD generalization. The techniques employed in this pursuit encompass a diverse spectrum, including modifying position embeddings [Jelassi *et al.*, 2023] and attention mechanisms [Dubois *et al.*, 2019], fine-tuning using extended data samples, prompting and Scratchpad [Anil *et al.*, 2022], priming through selective longer-length data [Jelassi *et al.*, 2023], and even utilizing chain-of-thought (CoT) style data [Lee *et al.*, 2023].

In spite of these different techniques, there is still a lack of understanding regarding the underlying mechanism. The proposed solutions may therefore have questionable robustness and become vulnerable to circumstance changes [Jelassi *et al.*, 2023]. Considering the evident and notably poor OOD performance, it is natural to ask whether it stems solely from random errors or if there is anything informative learned by these models.

*Corresponding authors.

In this paper, we bring the mystery into attention and seek from the mechanistic perspective [Nanda and Lieberum, 2022; Zhong *et al.*, 2023] in model interpretability. This avenue of study offers a macroscopic understanding of how neural networks work and has helped identify and interpret significant phenomena such as "grokking", also known as delayed generalization where models exhibit improved generalization long after over-fitting their training set [Liu *et al.*, 2022].

When conducting experiments, it is intuitive to test models with ID samples as well as OOD ones to make comparisons. However, it is not feasible if we use well-known LLMs such as GPT-4 or Llama, since we do not know the exact data that they are trained on and therefore cannot distinguish between ID and OOD samples. On the other hand, training LLMs is inevitably very expensive [Brown *et al.*, 2020]. Nonetheless, the study by Anil *et al.* [2022] shows that when the model scale increases, the model ability to generalize across different task lengths does not improve. This suggests that the underlying mechanism may be irrelevant to model scale and all generative models may share the same mechanism. Inspired by this and just like many other studies [Lee *et al.*, 2023; Jelassi *et al.*, 2023], we dig smaller models for insights that could apply to LLMs. Besides, we further increase the model scales to examine the consistency in the range of scales that we can reach.

Tasks such as n-digit (modular) addition and multiplication are tools for investigating issues including length generalization [Anil *et al.*, 2022] and "grokking" [Liu *et al.*, 2022]. Albeit simple, they offer clearer, more controlled conditions, which can lead to more reliable observations and interpretations. In this paper, through training a set of small generative language models, including NanoGPT and MinGPT [Karpathy, 2022], on n-digit addition and multiplication tasks, we have made an intriguing discovery. We find that the strong ID generalization stems from structured representations, while the models have learned a clear algebraic structure behind the unsatisfying OOD performance. Specifically, these models map unseen OOD inputs to outputs with equivalence relations in the ID domain, which we call the phenomenon as *equivalence generalization*. The representation learning process plays a crucial role in facilitating both ID and OOD generalization observed in these models. Initially, the representations are random. But as training progresses, the structure of the learned representations becomes increasingly refined,equivalence generalization eventually allowing the models to accurately encode every input in the ID domain. Concurrently, these structured representations are continuously extended to map the unseen OOD domain. However, this extension does not occur as ideally anticipated, resulting in the poor OOD performance. Thus, the representation learning enables powerful ID generalization, but the continuous extrapolation of these representations to OOD inputs gives rise to systematic, rather than random, errors. The mechanistic insights from the discovered patterns also highlight the potential of these models to make use of the information for better generalization.

As a note, we perform several robustness studies in this work, such as changing the encoding method and varying the training data scheme. We find that the equivalence generalization phenomenon is robust. In addition, we conduct a detailed examination of the results across different model scales. Notably, our results remain consistent as the model scales increase, which strengthens our confidence that these results might be extended to LLMs.

Our main contributions are as follows:

- **Showcasing the power of mechanistic empirical evaluation for LLM generalization**: We train small generative language models (e.g., NanoGPT, MinGPT) on arithmetic tasks to directly investigate ID vs. OOD generalization, rather than resorting to workarounds. As a result, our approach provides macroscopic insights. To facilitate relevant research, we opensource our code[1].
- **Discovering learned structure for OOD generalization**: The discernible algebraic structure and the equivalence generalization would hopefully guide robust essential solutions for strong OOD generalization.
- **Understanding the role of representations in generalization**: We show that representation learning enables strong ID performance, while unanticipated extension of representations to OOD inputs leads to systematic errors.

2 Related Work

2.1 Generalization of Language Models in Arithmetic

Various studies have examined the performance of Transformer-based language models in tasks involving arithmetic operations. Brown *et al.* [2020] investigated the ability of GPT-3 to perform basic arithmetic operations without task-specific training. Nogueira *et al.* [2021] explored the limitations of transformers in handling simple arithmetic operations. Subsequent studies have further explored the generalization capabilities of language models in arithmetic tasks. Qian *et al.* [2022] discovered that language models exhibit poor OOD generalization, and traditional methods such as explicit positional markers and fine-grained computation steps do not effectively address this issue. To enhance the generalization ability of the model, certain studies have approached the issue starting from a microscopic perspective. For instance, Jelassi *et al.* [2023] replaced absolute position embeddings with relative position embeddings. Additionally, Dubois *et al.* [2019] suggested that utilizing a location-based attention mechanism proves effective in the Lookup Table task. Other research has focused on the intermediate learning process of the model. Anil *et al.* [2022] observed that requesting the model to generate intermediate arithmetic steps before providing the final output can improve generalization. Jelassi *et al.* [2023] arrived at similar conclusions by decomposing the arithmetic pipeline and improving generalization in five-digit addition tasks. In contrast, Lee *et al.* [2023] presented a different perspective, emphasizing the importance of high-quality, instructive data that can quickly elicit arithmetic capabilities.

[1]The code is available at https://github.com/xingchengxu/ExploreGPT

While previous studies have primarily focused on evaluating or improving the generalization capabilities of language models, our work has a different objective, we aim to uncover the underlying mechanisms that govern generalization. This explanatory goal, which seeks to understand the foundations of generalization, has not been explicitly addressed in prior research.

2.2 Mechanistic Interpretability

Neural network interpretation has seen numerous studies focusing on various types of models, including deep neural networks (DNNs) [Nam *et al.*, 2020; Barbiero *et al.*, 2022], convolutional neural networks (CNNs) [Yuan *et al.*, 2019; Akhtar and Ragavendran, 2020], and graph neural networks (GNNs) [Yuan *et al.*, 2020; Xuanyuan *et al.*, 2023]. These works demonstrate diverse microscopic interpretation techniques tailored to different architectures. From a macroscopic perspective, Liu *et al.* [2022] tackle delayed generalization or "grokking" using addition and modular addition tasks. They provide intuitive explanations using effective theories and phase diagrams. Similarly, Zhong *et al.* [2023] use modular addition to mechanistically explain algorithm discovery in neural networks. Our work contributes to this growing field of mechanistic interpretability by offering a macroscopic explanation specifically for generative Transformer models.

3 Preliminary and Experimental Setup

3.1 Model Details

We employ the model framework of GPT, a Transformer with a decoder-only architecture comprising multiple layers and multi-head attentions. We train several small-scale models, namely NanoGPT and MinGPT Karpathy [2022], from random initialization using character-level tokenization and the conventional next-token prediction objective. The training is conducted on basic mathematical operations, specifically addition and multiplication of integers. Detailed hyperparameters are shown in Table 1.

Hyperparameter	Addition	Multiplication
num layer	3	6
num head	3	6
dim embd	48	192
vocab size	10	10
context window	15	19
dropout prob	0.1	0.1
optimizer	AdamW	AdamW
learning rate	0.0005	0.0005
betas	(0.9, 0.95)	(0.9, 0.95)
weight decay	0.1	0.1
grad norm clip	1.0	1.0

Table 1: Hyperparameter Information

3.2 Dataset

The dataset is structured as a concatenation of operand pairs in a natural order, with the reversed order of the operation results. This format, demonstrated to be more conducive for learning in next-token prediction models Lee *et al.* [2023], offers a more approachable learning process. For instance, consider the 3-digit addition $a+b=c$, represented as "$a_2a_1a_0+b_2b_1b_0=c_3c_2c_1c_0$" in the standard format. By reversing the output order of "c", we obtain the reversed data format "$a_2a_1a_0+b_2b_1b_0=c_0c_1c_2c_3$". As we train addition and multiplication models as distinct entities, we omit both the operation symbols, i.e., $+$ and $\times$, and the equal sign, i.e., $=$, from the dataset. Subsequently, the data undergoes character-level tokenization, resulting in a vocabulary size of 10, corresponding to digits from 0 to 9. When the context window surpasses the requisite size for a 3-digit addition, we pad zeros before numbers "a", "b", and "c". For instance, in the case of 3-digit addition with a context window of 15, the addition expression "$349+705=1054$" will be encoded as "0034900705450100".

The dataset is partitioned into three distinct subsets: the training set $\mathscr{D}_1$, randomly sampled from n-digit operations; the test set $\mathscr{D}_2$, also drawn from n-digit operations but intentionally devoid of any overlap with the training set (termed as the ID test set); and an additional test set $\mathscr{D}_3$, sampled from m-digit operations with $m > n$, where the value at positions greater than n is non-zero (referred to as the OOD test set).

In the experiments, we set $n=3$ and $m=5$ for both addition and multiplication operations. Subsequently, from each of the datasets $\mathscr{D}_1$, $\mathscr{D}_2$, and $\mathscr{D}_3$, we select 10,000 data points as the training set for addition and 50,000 for multiplication. We sample 10,000 for the ID test set and OOD test set, respectively for both operations.

3.3 ID and OOD Domains

The data space is compartmentalized into three non-overlapping regions: $\mathscr{D}_1, \mathscr{D}_2$, and $\mathscr{D}_3$. The union of $\mathscr{D}_1$ and $\mathscr{D}_2$ constitutes an ID domain, whereas $\mathscr{D}_3$ represents an OOD domain. The models learn a function

$$f : \mathscr{D}_1 \cup \mathscr{D}_2 \cup \mathscr{D}_3 \to \mathscr{S},$$

where $\mathscr{S}$ could be the output operation result space $\mathbb{N}$, the output probability space, or the learned representation space.

Since the model is exclusively trained on the $\mathscr{D}_1$ space, the acquired knowledge concerning $\mathscr{D}_2$ and $\mathscr{D}_3$ is an extension of $\mathscr{D}_1$, albeit with an unclear underlying structure. This constitutes the core aspect we seek to understand.

3.4 Equivalence Classes

When training for addition and multiplication on n-digit operations, we have identified a discernible algebraic structure. This is encapsulated in the definition of the equivalence class $[(a,b)]_p$ for modular p, which is elucidated as follows:

$$[(a,b)]_p := \{(x,y) \in \mathbb{N}^2 |\ x \equiv a \,(\mathrm{mod}\, p),\ y \equiv b (\mathrm{mod}\, p)\}.$$

The ensemble of these equivalence classes is denoted as

$$\mathbf{Z}_p^2 = \mathbf{Z}_p \times \mathbf{Z}_p = \{[(a,b)]_p |\ (a,b) \in \mathbb{N}^2\},$$

where $\mathbf{Z}_p = \mathbb{Z}/p\mathbb{Z}$ on non-negative integers.

To illustrate, when training a model on 3-digit addition, we observe that the learned operation function $f_{op} : \mathbb{N}^2 \to \mathbb{N}$ effectively translates to $f_{op}(a,b) = f_{op}([(a,b)]_{10^3})$, which will be stated in the result section.

In alternative training data scenarios, the definition of equivalence classes necessitates adaptation to accommodate specific contexts.

4 Results

In this section, we present the key results and findings from our experiments. These include observations on the phenomenon of generalization exhibited by the models, the learned algebraic structure, as well as the probability and representation structures in the model's learning process.

4.1 Generalization in OOD Domain

Figure 1 depicts the training, ID test, and OOD test accuracy for addition and multiplication operations in domains $\mathscr{D}_1, \mathscr{D}_2$, and $\mathscr{D}_3$ across different iterations. Panel (a) displays the training curve for addition learned by NanoGPT, while Panel (b) showcases the curve for multiplication learned by MinGPT. The hyperparameters employed by NanoGPT and MinGPT can be found in Table 1.

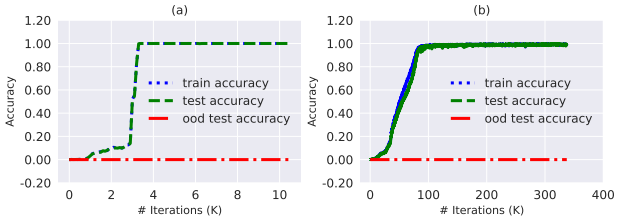

Figure 1: Training curves in addition and multiplication operations.

By examining the figure, it becomes evident that both addition and multiplication quickly converge to a stable state, achieving (almost) 100% accuracy in training and ID testing in $\mathscr{D}_1$ and $\mathscr{D}_2$. However, throughout the entire training process, the OOD test accuracy remains zero for both 3-digit addition and multiplication in $\mathscr{D}_3$. These results align with the discoveries made by Jelassi *et al.* [2023] and Lee *et al.* [2023]. When training on n-digit operations with n-digit operands, the models demonstrate excellent generalization on unseen n-digit inputs. Yet, they perform abysmally and mysteriously on longer, unseen cases, establishing a contrast between ID generalization and OOD generalization. Given the strikingly poor OOD performance, it is natural to question whether it solely stems from random errors or if there is any meaningful knowledge learned. The solution to the problem will be presented in the subsequent subsection.

4.2 Algebraic Structure

The mysterious absence of generalizability in the OOD domain prompts us to delve deeper into the results. We begin by examining some samples from domains $\mathscr{D}_2$ and $\mathscr{D}_3$. These examples are illustrated in Table 2. When observing the 3-digit addition and multiplication cases, we notice that the trained models produce incorrect results for the 4-digit instances. Strikingly, these erroneous outputs mirror the results obtained from the 3-digit cases. It appears that the model's outputs peculiarly "disregard" the thousands digit of the input numbers, irrespective of whether it is an addition or multiplication operation.

To systematically analyze the behavior in the OOD domain $\mathscr{D}_3$, we explore the entire two-dimensional lattice of 4-digit integers, namely $\mathbb{N}^2 \cap [0, 10^4)^2$. Figure 2 presents the contour

Operands	Output Result	Correct Result
349 + 705	1,054	1,054
1,349 + 2,705	1,054	4,054
128 × 256	32,768	32,768
3,128 × 4,256	32,768	13,312,768

Table 2: Examples on models' outputs for addition and multiplication.

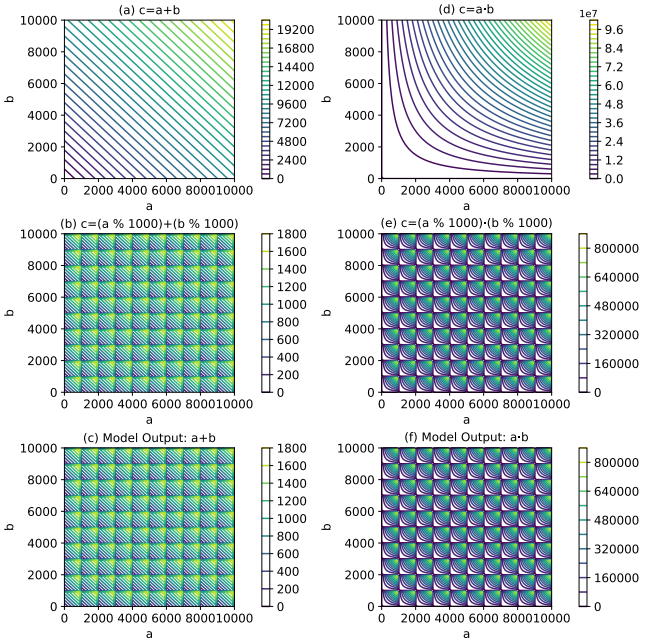

Figure 2: Contour plots for addition and multiplication operations.

plots for the ground truth results of addition operation $c = a + b$ (Panel (a)) and multiplication operation $c = a \cdot b$ (Panel (d)), with the number a on the horizontal axis and the number b on the vertical axis. These landscapes represent the expected learning and generalization capabilities of the models on this lattice space.

However, when we utilize our trained models to generate results based on 3-digit operations, an unmistakably distinct pattern emerges, as depicted in Panel (c) for addition and Panel (f) for multiplication. This prompts us to investigate what structure the models have learned. We discover that there is a modular relationship between the operands a and b. The learned structure can be represented as $c = (a \bmod 10^3) \circ (b \bmod 10^3)$, where $\circ$ represents either addition $+$ or multiplication $\times$. The ground truth landscapes of these functions on the 4-digit integer lattice are exhibited in Panel (b) for addition and Panel (e) for multiplication. Visually, these two panels are identical to Panel (c) and Panel (f), respectively. We compare the results of the operation $(a \bmod 10^3) \circ (b \bmod 10^3)$ with the outputs produced by the model. Surprisingly, they are identical across the entire space $\mathbb{N}^2 \cap [0, 10^4)^2$.

To formalize the results, we recall the definition of equiva-

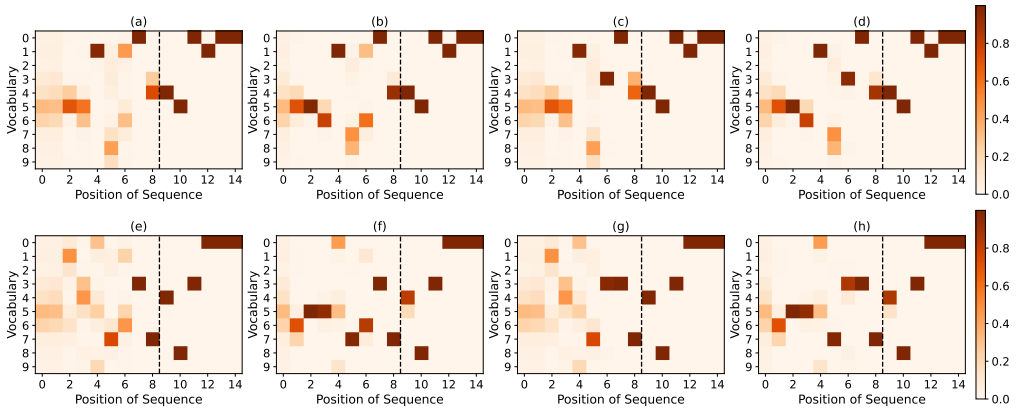

Figure 3: The probability distribution of each digit of the sequence in an addition operation $c = a + b$. The left side of the black dashed line represents the input $a + b$, while the right side is the result c. Figure 3(a) and Figure 3(e) represent the $349 + 705$ and $128 + 256$, and the outputs are $1,054$ and 384 (450100 and 483000 in actual sequence output), respectively. In the second column, we perturb the thousands digit of a: Figure 3(b) represents $1,349 + 705$, and Figure 3(f) represents $3,128 + 256$. In the third column, we perturb the thousands digit of b: Figure 3(c) represents $349 + 2,705$, and Figure 3(g) represents $128 + 4,256$. In the fourth column, we simultaneously perturb the thousands digit of a and b: Figure 3(d) represents $1,349 + 2,705$, and Figure 3(h) represents $3,128 + 4,256$.

lence classes $[(a,b)]_p$ for modular $p = 10^3$:

$$[(a,b)]_p := \{(x,y) \in \mathbb{N}^2 | \; x \equiv a \,(\mathrm{mod}\, p),\; y \equiv b (\mathrm{mod}\, p)\}.$$

As $[(a,b)]_p$ is an equivalence class, we use the element in $\mathbb{N}^2 \cap [0,10^3)^2 = \mathscr{D}_1 \cup \mathscr{D}_2$ to serve as the representative of the class. The ensemble of these equivalence classes then forms the space

$$\mathbf{Z}_p^2 = \mathbf{Z}_p \times \mathbf{Z}_p = \{[(a,b)]_p | \; (a,b) \in \mathbb{N}^2\},$$

where $\mathbf{Z}_p = \mathbb{Z}/p\mathbb{Z} = \{[0]_p, [1]_p, \cdots, [p-1]_p\}$.

The models trained on 3-digit addition and multiplication actually learned the operation functions $f_{op} : \mathbf{Z}_p \times \mathbf{Z}_p \to \mathbb{N}$ for all integer paris on $\mathbb{N} \times \mathbb{N}$ such that $f_{op}(a,b) = f_{op}([(a,b)]_p)$ with $p = 10^3$. As an example, $f_{op}(1349, 2705) = f_{op}([(349,705)]_{10^3})$. For addition, the learned operation is $f_+(1349,2705) = f_+([(349,705)]_{10^3}) = 1054$, while the learned multiplication is $f_\times(1349,2705) = f_\times([(349,705)]_{10^3}) = 246045$.

As a summary of the results, the models learn to generalize the input in the OOD domain $\mathscr{D}_3$ by assimilating equivalence classes in the ID domain $\mathscr{D}_1 \cup \mathscr{D}_2$. This result shows the limitations of the models. However, this capability allows the models to extend their learned knowledge beyond the ID domain $\mathscr{D}_1 \cup \mathscr{D}_2$ shaped by the specific training data $\mathscr{D}_1$. Even though the output is wrong, it is not so bad. They have still managed to acquire useful information and demonstrate some level of learning.

In order to gain a deeper understanding of the training process for Transformer models, we examine their token-level mapping using addition as an example. Consider two (n+1)-digit numbers, where $a = a_n \times 10^n + \cdots a_1 \times 10 + a_0$ and $b = b_n \times 10^n + \cdots b_1 \times 10 + b_0$. When training a Transformer model using randomly sampled (n+1)-digit numbers, the model learns an approximate mapping from the token-level input to the true function $c = a + b = c_{n+1} \times 10^{n+1} + \cdots c_1 \times 10 + c_0$. The learned approximation allows the model to perform classification for each digit of the resulting sum c, as follows:

$$f_{Trans}(a_n, \cdots, a_0, b_n, \cdots, b_0) \approx (c_0, c_1, \cdots, c_{n+1}).$$

However, if the highest digit is completely absent from the training data and is instead padded with zeros, the training only guarantees learning of low n-digit addition. In other words:

$$f_{Trans}(0, a_{n-1}, \cdots, a_0, 0, b_{n-1}, \cdots, b_0) \approx (c_0, c_1, \cdots, c_n, 0).$$

This limitation may explain why it is challenging to generalize to higher digits when the model is trained solely on lower digits. The absence of examples with higher digits restricts the model's ability to accurately predict and generalize beyond the low-digit addition it has been trained on.

Building upon the observed algebraic structure discussed in the previous context of this subsection, we also know that when testing the Transformer models on higher digits that are non-zero, they do not significantly impact the classification of each digit of c.

As a remark, it is important to note that when dealing with alternative training data scenarios, the definition of equivalence classes may need to be adjusted accordingly. For example, if the training data consists exclusively of 1 and 3-digit operations, while OOD testing involves 2 and 4-digit numbers, or if the training data includes 3-digit numbers for operand a and 4-digit numbers for operand b, the equivalence classes would require redefining to account for these specific contexts.

4.3 Probability Structure

In the preceding subsection, we examined the structured algebraic patterns present in the output results. Considering that generative Transformer models generate outputs based on probability distributions, our model employs a greedy approach to select the output sequence with the maximum probability. We now shift our focus from algebraic structures to

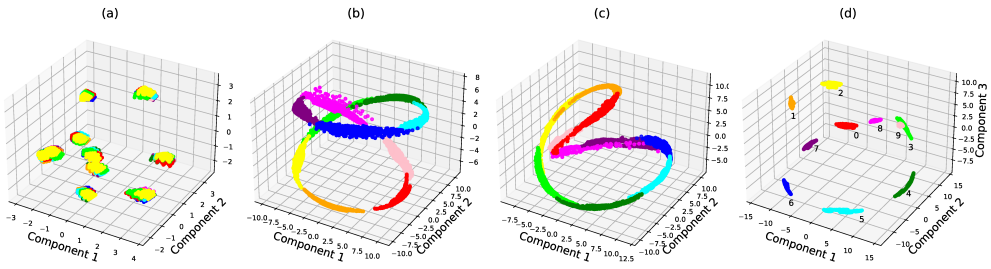

Figure 4: 3D representation structure of the first three principal components in the addition operation. Figure 4(a) to Figure 4(d) represent the initial model, model with 14%, 51%, and 100% test accuracy, respectively.

the probability distribution of the output sequences. Our objective is to investigate the underlying factors that contribute to the emergence of these structured algebraic patterns.

Specifically, we take two examples of 3-digit addition, namely $a+b$. We introduce perturbations to the thousands digit of both a and b, enabling us to compare the variations in probability distributions before and after the perturbations occur. This comparative study will shed light on the mechanisms underlying the observed structured algebraic patterns.

Figure 3 displays the probability distributions of the next tokens in the vocabulary at each position within the sequence for two examples. The plot showcases the probabilities before and after perturbations for each token. Remarkably, we observe that regardless of whether we perturb a and b separately or simultaneously, the probability distribution in the model's output sequence remains largely unchanged.

Furthermore, we note that the digits with the highest probability in the output sequence remain consistent. This result implies that the algebraic structure of the model expands from $\mathbf{Z}_p \times \mathbf{Z}_p$ to $\mathbb{N} \times \mathbb{N}$. This expansion elucidates the structured patterns depicted in Figure 2. Additionally, we conducted a systematic examination of the entire integer lattice within $\mathbb{N}^2 \cap [0, 10^4)^2$. Notably, the results obtained from this comprehensive analysis exhibit robustness, further supporting our findings.

4.4 Representation Structure

Within the probability structure, we made a significant observation that the model's output remains insensitive to perturbations in the thousands digit. This probability structure is rooted in the representation of the input sequence, which can be expressed as follows: $\mathbf{P} = \text{Softmax}(\mathbf{WX})$, where $\mathbf{P} \in [0,1]^{V \times L_{input}}$ represents the probability matrix for the next tokens at each position, $\mathbf{W} \in \mathbb{R}^{V \times d_{model}}$ signifies the learned weight matrix, and $\mathbf{X} \in \mathbb{R}^{d_{model} \times L_{input}}$ denotes the learned representation matrix of the input. The variables V, L_{input}, and d_{model} correspond to the vocabulary size, input length, and model embedding dimension, respectively.

In this subsection, we delve deeper into the influence of these representations on the probability structure, thereby shedding light on their role in shaping the observed algebraic properties.

In order to explore the representations of $a+b$ in a systematic manner, we conducted a thorough analysis on the two-dimensional integer lattice of 4-digit numbers. Specifically, for each input sequence $a+b$, we obtained a high-dimensional embedding by considering the last column of the learned representation matrix $\mathbf{X}$. Subsequently, we applied principle component analysis (PCA) to project these embeddings into three dimensions.

Figure 4 showcases the four different phases of representation observed during the learning process of the model. The visualizations in the figure depict the representations using the first three principle components. More specifically, Figure 4(a) to 4(d) correspond to the random initial model, the model with approximately 14%, 51%, and 100% ID test accuracy, respectively. The colors in each figure correspond to the true units digit of the resulting $a+b$.

The observations made from Figure 4 demonstrate that the representations gradually transition from disorderly to structured throughout the learning process. Initially, the representations appear random with colors mixed together (Figure 4(a)). However, as the training progresses, the structure of the learned representations becomes increasingly refined (Figure 4(b) and (c)), ultimately leading to the development of a well-learned representation (Figure 4(d)) where each color is separated according to its true label.

4.5 From Representation to Algebraic Structure

The findings discussed above regarding algebraic structures, probability distributions, and representation structures also hold true for multiplication operations.

The systematic analysis approach outlined earlier provides a comprehensive understanding of the model's generalization capabilities through the assimilation of equivalence classes present within the ID domain. The representation structures successfully incorporate the assimilation of these equivalence classes, thereby extending the ID structure to OOD scenarios via the probability distribution of sequences. Consequently, this assimilation becomes evident within the algebraic structures as well.

5 Robustness Studies

In this section, we conduct thorough empirical analyses using various model sizes (GPT-Nano, GPT-Micro, GPT-Mini) and training data volumes, also exploring different datasets and encoding methods, to validate the robustness of our findings.

(1) **Encoding method:** For the main experiments, we chose the reversed encoding method for n-digit addition and multiplication, due to its faster convergence speed. Here we

test the alternative non-reversed encoding method and obtain consistent results (see V_3 in Table 3). The convergence time, consequently, is approximately 7.44 times longer than that of the reversed encoding method.

(2) **Scope of the dataset and training scheme:** Additional experiments with variations in the training set include setting the rightmost digit to 0 (see V_1 in Table 3), setting the tens digit to 0 (V_2), and extending the OOD test to 10^6 and 10^7 (V_4). All the variations achieve 100% accuracy for ID domain and 0% for OOD domain. The results from V_3 and V_4 in OOD completely correspond with those of the equivalence class $[(a,b)]_{1000}$. Similarly, V_1 and V_2's OOD results are totally consistent with these from the equivalence class $[(a,b)]_{10}$, as defined in the following equations (1) and (2), respectively:

$$[(a,b)]_p := \{(x,y) \in \mathbb{N}^2 \mid x \equiv \lfloor \tfrac{a}{p} \rfloor \cdot p,\ y \equiv \lfloor \tfrac{b}{p} \rfloor \cdot p\}. \quad (1)$$

$$[(a,b)]_p := \{(x,y) \in \mathbb{N}^2 \mid x \equiv \lfloor \tfrac{a}{10p} \rfloor \cdot 10p + a \bmod p,\ y \equiv \lfloor \tfrac{b}{10p} \rfloor \cdot 10p + b \bmod p\}. \quad (2)$$

Versions	ID	OOD
V_1: rightmost digit be 0	100%	0
V_2: tens digit be 0	100%	0
V_3: non-reverse encoding	100%	0
V_4: extended OOD	100%	0

Table 3: The accuracy of ID test and OOD test in different addition variations.

(3) **Model and data scales:** To explore the potential applicability of our findings to large models, we conducted a detailed examination of outcomes across different model and data scales. Our analysis included three distinct model scales with increasing size: GPT-Nano, GPT-Micro, and GPT-Mini, as defined in the code. In addition to model size, we also evaluated the influence of varying training data sizes, specifically 20k and 50k, on the task of 3-digit addition. We focused on the accuracy of OOD test samples by comparing the model outputs with the results on $(a\%1000)+(b\%1000)$. As depicted in Figure 5, there's a noticeable trend where, with progressing training, the above accuracy in all experiments approaches 100%, and the algebraic structure of equivalence classes becomes more evident in OOD tests across different model and data scales. Notably, even as the model scales increase, our findings remain consistent. This consistency reinforces our confidence that these results might extend to larger language models (LLMs).

6 Discussion

In this section, we discuss some aspects of our investigation. Our work corroborates the findings of Anil *et al.* [2022] that increasing the size of a model does not increase its ability to generalize across tasks of different lengths. Through careful robustness studies across a range of model sizes, we make similar observations with respect to equivalence generalization. This suggests that the ability of length generalization

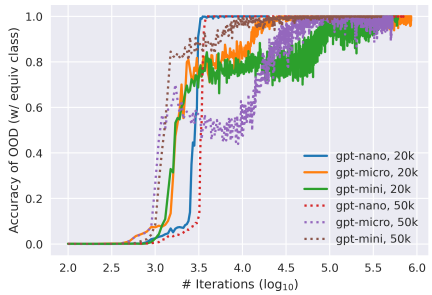

Figure 5: The accuracy of OOD test on equivalence for different model and data scales

may be independent of model size, and that such findings may be applicable to large language models.

Another point to consider is that while we discovered equivalence classes defined by modular $(a\%p, b\%p)$, the output itself $(a\%p)+(b\%p)$ is not modular arithmetic $(a+b)\%p$. This is different from the direct study of modular arithmetic such as conducted in Jelassi *et al.* [2023]. Moreover, Jelassi *et al.* [2023] merely raises the question of why modulo 100 works effectively while modulo 101 fails, without exploring beyond the observation. In contrast, our study highlights the consistency between the definition of equivalence classes and modular arithmetic, enabling us to explain observed differences and offer insights into the behavior of the models.

7 Conclusion

We investigate the length generalization problem in arithmetic tasks for generative language models. We perform mechanistic analysis on smaller models and reveal that these models have strong generalization within the trained distribution. However, our investigation also uncovers an underlying algebraic structure that contributes to the models' unsatisfactory performance on OOD inputs. The models attempt to map OOD inputs using equivalence relations within the ID domain (we call *"equivalence generalization"*), leading to errors and a lack of robustness in OOD scenarios. The representation plays a crucial role in enabling both ID and OOD generalization. The observation that length generalization ability does not vary with model scale, helps us extend our conclusion to LLMs.

Despite challenges in OOD generalization, our findings suggest that these models hold valuable information for improved generalization. However, due to the inherent subjectivity of natural language, much more efforts are needed to establish equivalence in NLP tasks for LLMs. In addition, the finding of equivalence generalization may serve as helpful prior knowledge, guiding the training process of LLMs regarding generalizability. For example, we may stop training once these equivalence classes are formed, reducing the extensive data needed for generalizability. Besides, in domain adaptation, people often finetune existing models, to adapt to OOD data and similarity metrics of equivalence classes may facilitate this process.

Acknowledgments

This work is supported by Shanghai Artificial Intelligence Laboratory.

Contribution Statement

Xingcheng Xu and Zihao Pan contributed equally in this work.

References

Nadeem Akhtar and U Ragavendran. Interpretation of intelligence in cnn-pooling processes: a methodological survey. *Neural computing and applications*, 32(3):879–898, 2020.

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

Anthropic. Model card and evaluations for claude models. 2023.

Pietro Barbiero, Gabriele Ciravegna, Francesco Giannini, Pietro Lió, Marco Gori, and Stefano Melacci. Entropy-based logic explanations of neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 6046–6054, 2022.

Emily M Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. 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, 2021.

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

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

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113, 2023.

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

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

Andrej Karpathy. A minimal pytorch re-implementation of the openai gpt (generative pretrained transformer) training. *GitHub https://github.com/karpathy/minGPT*, 2022.

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

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

Woo-Jeoung Nam, Shir Gur, Jaesik Choi, Lior Wolf, and Seong-Whan Lee. Relative attributing propagation: Interpreting the comparative contributions of individual units in deep neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 2501–2508, 2020.

Neel Nanda and Tom Lieberum. A mechanistic interpretability analysis of grokking. In *Alignment Forum*, 2022.

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

OpenAI. Gpt-4 technical report. *ArXiv*, abs/2303.08774, 2023.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744, 2022.

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

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

Immanuel Trummer. Codexdb: Synthesizing code for query processing from natural language instructions using gpt-3 codex. *Proceedings of the VLDB Endowment*, 15(11):2921–2928, 2022.

Han Xuanyuan, Pietro Barbiero, Dobrik Georgiev, Lucie Charlotte Magister, and Pietro Lió. Global concept-based interpretability for graph neural networks via neuron analysis. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 10675–10683, 2023.

Hao Yuan, Yongjun Chen, Xia Hu, and Shuiwang Ji. Interpreting deep models for text analysis via optimization and regularization methods. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 5717–5724, 2019.

Hao Yuan, Jiliang Tang, Xia Hu, and Shuiwang Ji. Xgnn: Towards model-level explanations of graph neural networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 430–438, 2020.

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

Mingyu Zong and Bhaskar Krishnamachari. Solving math word problems concerning systems of equations with gpt-3. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 15972–15979, 2023.