From Pattern Recognition to Reasoning:
A Survey for Transformer Generalization and Recall in Algorithmic Tasks

Dumitru Verşebeniuc
Student at Maastricht University
d.versebeniuc@student.maastrichtuniversity.nl

Aki Härmä
Assistant Professor at Maastricht University
aki.harma@maastrichtuniversity.nl

Abstract

This paper surveys recent advances in enhancing the generalization capabilities of Transformer models in algorithmic tasks, marking a shift from pattern recognition to algorithmic reasoning. We introduce a comprehensive taxonomy that differentiates between length generalization -—- the ability to extend fixed computational procedures to longer inputs —— and compositional generalization, where models dynamically assemble learned subroutines to solve more complex problems. Our review covers key techniques that improve Transformer performance on arithmetic tasks, such as integer addition, by leveraging innovative data formatting strategies (including reversed formats, index hints, random space augmentation, and zero-padding) alongside advanced positional encoding methods (e.g., absolute, additive relative, position coupling, and randomized encodings). At the end, this paper outlines the present challenges and highlights possible future directions for research and development.

1 Introduction

Transformer models have revolutionized various fields, including natural language processing (Liu et al., 2023), computer vision (Dosovitskiy et al., 2020), and even protein folding (Jumper et al., 2021). Their impressive ability to learn complex patterns has driven progress in areas such as machine translation, text summarization, and question-answering. Furthermore, Transformers are proficient in complex tasks like mathematical reasoning, code synthesis, and theorem proving (Guan et al., 2025). However, despite these successes, a crucial limitation persists: generalization, which is the ability of a model to extrapolate to input exceeding those encountered during training. Generalization poses a significant challenge for Transformers, raising questions about whether they truly understand the underlying algorithms or simply rely on shallow shortcuts that break down when faced with longer or more complex inputs (Anil et al., 2022; Veličković et al., 2022; Zhang et al., 2023; Dziri et al., 2023; Bayat et al., 2024). This inability to generalize to unseen data hinders the development of robust and reliable AI systems that handle real-world complexities.

Generalization in Transformers can be viewed as a form of generalized reasoning about the underlying task or understanding of inherent patterns. It indicates whether a Transformer model has learned to apply a systematic, algorithmic approach to a task, rather than merely memorizing specific input-output examples. Analyzing generalization through the lens of reasoning offers a valuable perspective on Transformer capabilities (Elhage et al., 2021; Olsson et al., 2022). Several studies support this perspective:

- Zhou et al., 2024 demonstrated that Transformers can achieve length generalization, but this ability is not always robust. The study highlights that the success of length generalization is heavily influenced by factors like data format, positional encodings, random weight initialization, and training data order, leading to significant variance in performance. This suggests that while Transformers are capable of learning generalizable solutions, their performance is fragile and easily disrupted by changes in the training environment.

- Zhou et al., 2023 proposed the RASP-Generalization Conjecture, which suggests that Transformers demonstrate length generalization when the underlying algorithm can be easily represented in the RASP programming language. RASP (Recursive Attentive State Programming Language) is specifically designed to analyze algorithms that can be represented by Transformers. This conjecture implies that the length generalization indicates a deeper understanding of the task's structure

rather than just surface-level pattern matching. The study suggests that if an algorithm can be written as a short program in RASP-L, then the Transformer is likely to generalize well to longer sequences.

- Induction Heads, see (Elhage et al., 2021; Olsson et al., 2022), have been proposed as a mechanism for generalization that facilitate in-context learning and generalization in Transformers. Induction heads are specialized attention mechanisms that learn to copy and complete patterns. This suggests a link between pattern recognition, induction, and the ability to generalize to new situations or lengths.

In this paper, we aim to provide a framework for understanding when Transformer models generalize or fail to generalize in algorithmic scenarios. Our principal goal is to define a taxonomy that captures different levels of generalization and memorization, shedding light on when models truly learn underlying algorithms versus when they merely memorize patterns. In doing so, we seek to clarify the limits of Transformers' reasoning capabilities and offer insights into how these capabilities can be strengthened.

To achieve this, we build upon a wide range of existing research, incorporating theoretical perspectives and empirical findings to establish a view of transformer generalization. We examine the impact of critical design considerations, such as positional encoding strategies, data formats, training parameter settings, and architectural nuances. Together, these elements interact in complex ways to determine whether a Transformer develops robust algorithmic reasoning or tends to memorization. In addition, we explore foundational algorithmic tasks as a simple yet effective way to study generalization. By combining rapid model scaling with well-structured training protocols in these simpler tasks, we can better identify what leads to success or failure in Transformer-based models. Foundational algorithmic tasks not only help uncover the underlying mechanisms, but also provide a broader understanding of generalization in real-world scenarios.

The rest of this paper is organized as follows. In the **Taxonomy Section**, we describe our proposed taxonomy, detailing the dimensions along which generalization and memorization can be classified in algorithmic tasks. In the **Algorithmic Tasks Section**, we introduce possible simple algorithmic tasks that help illustrate the specific types of generalization that we aim to achieve. The **Preliminaries Section** discusses critical factors such as positional encoding, data formats, training setups, and architectural considerations. The **Results Section** presents experimental results, examining the impact of these factors on one of the length generalization problems. Finally, the **Discussion and Conclusion Sections** summarize our key findings and outline open questions for future work, underlining the need for further exploration of robust, reasoning-oriented Transformer models that can reliably tackle the challenges of length and task extrapolation.

2 Taxonomy

In this section, we introduce a taxonomy that categorizes the key dimensions affecting Transformer generalization in algorithmic tasks, including aspects such as data distribution, memorization, and compositional reasoning.

2.1 Length Generalization

Length Generalization refers to a model's ability to **internalize the underlying algorithmic pattern** of a task, enabling it to extrapolate from shorter training sequences to longer test sequences. This tests whether the model has learned the *core procedure* of the task (e.g., digit-wise addition with carry operations) rather than relying on surface-level patterns tied to specific input lengths. Success here implies the model executes the algorithm "mechanically," without needing to reason about the task structure actively.

Example 1 (Length Generalization)

Training Scenario: The model is trained on short integer addition problems like:

$$12 + 23 = 35$$

Testing Scenario: It must solve longer, unseen sequences, e.g.:

$$123456 + 123456 = 246912$$

This tests whether the model has internalized the *algorithm* (e.g., iterating digit-by-digit, handling carries). If successful, it demonstrates procedural mastery of addition, not just memorization.

2.2 Compositional Generalization

Compositional Generalization measures a model's capacity to **actively reason** by combining learned

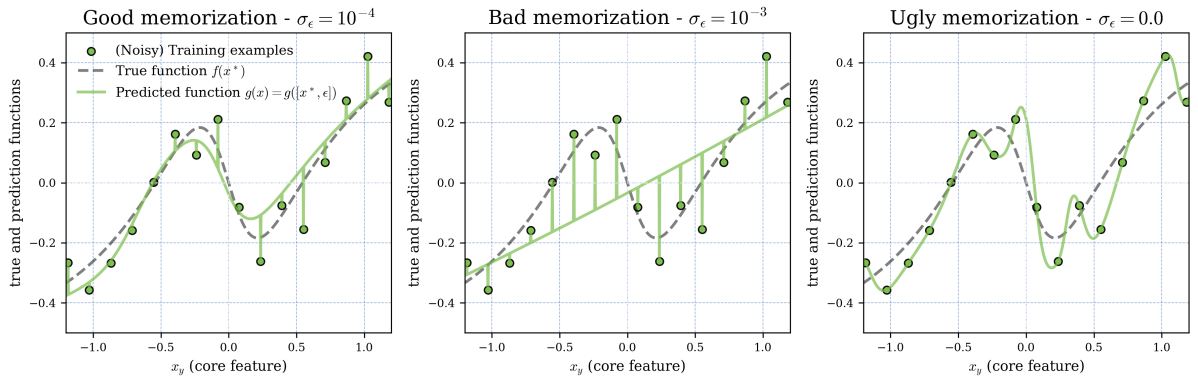

Figure 1: Three types of memorization in regression models trained with different levels of example-specific features (σ_ϵ). The plots show the Empirical Risk Minimization (ERM)-trained model $g(x) = g(x_y, \epsilon)$ (solid green line) versus the true underlying function $f(x_y)$ (dashed gray line) and the noisy training examples. In all three cases, the models are trained until the training loss goes below 10^{-6} (Bayat et al., 2024).

subroutines in novel ways to solve tasks more complex than those seen during training. Unlike Length Generalization, it requires the model to *structure its approach* dynamically, drawing on abstract representations of operations rather than executing a fixed procedure.

Example 2 (Compositional Generalization)

Training Scenario: The model learns expressions with isolated operations, e.g.:

$$2 \times 2 + 2 = 6$$

Testing Scenario: It must now reason about *how to combine operations* in new hierarchies, e.g.:

$$2 \times (2 + 2 \times 2) = 12$$

Here, the model must infer operator precedence and nested structure—tasks requiring planning, not just executing a known algorithm. Success implies it understands operations as modular tools for reasoning.

2.3 Key Distinction between Length and Compositional Generalization

- **Length Generalization** reflects *algorithmic internalization*: the model applies a fixed procedure to arbitrary input lengths.
- **Compositional Generalization** reflects *algorithmic reasoning*: the model dynamically assembles subroutines into novel workflows.

While overlapping in tasks like addition (where length and composition interact), they test distinct capabilities: one tests *mastery of a procedure*, and the other tests *flexibility in problem-solving*.

2.4 Memorization vs. Generalization

Formalizing the interplay between memorization and generalization is crucial, since spurious correlations can lead to poor generalization when combined with memorization. Memorization can reduce training loss to zero, leaving no incentive to learn robust, generalizable patterns. In Figure 1, adopted from Bayat et al., 2024, we look into a simple regression task to understand different types of memorization and their effects on generalization.

- **Good Memorization (Left, $\sigma_\epsilon = 10^{-4}$):** Model learns the true function $f(x_y)$ well but slightly memorizes residual noise in the training data using the input example-specific features ϵ. This type of memorization is benign, as it does not compromise generalization.
- **Bad Memorization (Middle, $\sigma_\epsilon = 10^{-3}$):** The model relies more on example-specific features than learning the true function $f(x_y)$, leading to partial learning of $f(x_y)$ and fitting of noise-dominated input features. This type of memorization impedes the learning of generalizable patterns and is considered malign.
- **Ugly Memorization (Right, $\sigma_\epsilon = 0.0$):** Without example-specific features, the model overfits the training data, including label noise, resulting in a highly non-linear and complex model that fails to generalize to new data. This type is referred to as catastrophic overfitting.

3 Length Generalization

In this paper, we focus primarily on Length Generalization, as this area has seen substantial foundational work (Zhou et al., 2023, 2024; Cho et al.,

2024), enabling clearer formalization of its core principles. By contrast, Compositional Generalization, while a prominent topic in modern research, often centers on large-scale reasoning models such as Gemini Thinking, ChatGPT variants (e.g., o1, o3), DeepSeek-R1, and rStar-Math (Guan et al., 2025). These models demonstrate impressive performance in algorithmic reasoning tasks, largely due to training methodologies involving supervised fine-tuning (SFT) and reinforcement learning with one of reward policy: Group Relative Policy Optimization (GRPO, a variant of Proximal Pol icy Optimization (PPO)) (Shao et al., 2024), Process Reward Models (PRMs) (Zhang et al., 2025), or Monte Carlo Tree Search (MCTS) (Guan et al., 2025). However, advancements in compositional generalization tend to emphasize training procedures (e.g., reward shaping or search-based optimization) rather than explaining how models internally generalize patterns. Furthermore, this field evolves rapidly, with shifting priorities toward enhancing "algorithmic thinking" rather than architectural interpretability.

Consequently, length generalization provides a more stable and structured framework for investigating architectural mechanisms in Transformers. By analyzing how models extrapolate to longer sequences independent of training dynamics, we can better isolate and improve their inherent generalization capabilities. Although this approach is usually tested on small transformer models and does not scale well to real-world problems in practice, it can serve as a foundation in the right direction.

Table 1 illustrates the diversity of algorithmic tasks studied in the literature — ranging from mathematical and reasoning tasks such as Addition (Nye et al., 2021), Polynomial Evaluation, Sorting, Summation (Saxton et al., 2019), Parity (Anil et al., 2022), LEGO (Zhang et al., 2023) — this paper prioritizes **addition** as a foundational case study for probing length generalization in Transformers. We justify this choice as follows:

1. **Simplicity and Interpretability**: Addition is a well-defined, deterministic task with minimal combinatorial complexity compared to operations like polynomial evaluation or sorting. Its stepwise nature (e.g., digit-wise processing with carry propagation) allows for granular analysis of how Transformers encode sequential dependencies and positional reasoning.

2. **Controlled Scalability**: The input length, in addition algorithmic tasks, can be systematically extended (e.g., from 5-digit to 10-digit numbers) without changing the underlying algorithm. This facilitates precise evaluation of generalization beyond training lengths.

3. **Prior Work**: Addition has served as a canonical task in length generalization studies (Zhou et al., 2023; Cho et al., 2024; Zhou et al., 2024), enabling direct comparisons with existing architectural modifications (e.g., positional encoding schemes, attention biases) and training paradigms.

4 Preliminaries

Several recent architectural improvements, especially in position encoding (Kazemnejad et al., 2023; Ruoss et al., 2023; Wang et al., 2024) and attention mechanisms (Duan et al., 2024; Dubois et al., 2020), have been proposed to tackle the length generalization challenge in arithmetic tasks using Transformers. However, these modifications are often limited by their ad-hoc nature or poor performance on longer sequences. Although scaling model and dataset sizes are known to improve performance, it might not be sufficient for generalizing to test sequences longer than those seen during training (Anil et al., 2022). Therefore, in addition to architectural improvements, data-centric AI has driven research (Kumar et al., 2024; Motamedi et al., 2021) to refine data formats to improve the learning quality of Transformers. This section will review common data formats (Section 4.1) and positional embeddings/encoding methods (Section 4.2) relevant to length generalization with a focus on decoder-only Transformers that solve the tasks using next-token prediction (See Appendix A for the brief background of decoder-only Transformers)

4.1 Data Formats

The structuring of data plays a role in improving the length generalization capabilities of Transformer models by reformatting data into a representation that facilitates more effective learning. Below, we provide an overview of the existing methodologies in this domain.

4.1.1 Reversed Format

Recent studies have demonstrated that reversing the response in arithmetic problems can substantially improve both performance and sample efficiency

Table 1: Examples of the input and output of the algorithmic tasks.

Task Type	Question	Answer
Addition	Compute: $53726 + 19177$	72903
Polynomial Eval.	Evaluate $x = 3$ in $3x^0 + 1x^1 + 1x^2 \mod 10$	5
Sorting	Sort the numbers: 3,1,4,1,5	1,1,3,4,5
Summation	Compute $(1 + 2 + 3 + 4 + 7) \mod 10$	7
Parity	Are the number of 1's even in [1,0,0,1,1]?	No
LEGO	$a = -1$; $b = -a$; $c = +b$; $d = +c$. Find c?	$+1$

in neural models. For instance, (Lee et al., 2023) show that transforming an expression such as

`653 + 49 = 702`

into its reversed format,

`653 + 49 = 207`,

allows a decoder-only Transformer to generate the answer starting from the least significant digit (LSD) and proceeding towards the most significant digit (MSD). This reversal mirrors the traditional algorithm taught in elementary school, where addition is performed digit-by-digit from the LSD to the MSD.

Standard arithmetic expressions are typically written as

$$A_3A_2A_1 + B_3B_2B_1 = C_3C_2C_1,$$

where A_1 and B_1 denote the LSDs. This ordering poses a challenge for autoregressive models because they generate outputs sequentially beginning with the MSD, thus misaligning with the natural computational process. In contrast, the reversed format

$$A_1A_2A_3 + B_1B_2B_3 = C_1C_2C_3$$

aligns the generation order with the algorithmic steps of addition. The learning task is thereby simplified to computing a function that depends only on the two corresponding operand digits and the carry from the previous addition step (Lee et al., 2023; Shen et al., 2023; Zhou et al., 2023).

4.1.2 Index Hints

Index hinting is an input augmentation technique introduced by (Zhou et al., 2023) to explicitly encode positional structure into arithmetic tasks. In this method, index hints are inserted into both the query and the response. For example, the arithmetic expression (`42 + 39 = 81`) is represented during training and inference as (`a4b2 + a3b9 = a8b1`), thereby enabling transformers to execute indexing via induction heads (Olsson et al., 2022).

4.1.3 Random Space Augmentation

(Shen et al., 2023) investigated how inserting random spaces between digits in addition tasks could disrupt the model's dependency on fixed positional cues. Their findings indicate that while the model successfully generalized from 10-digit to 11-digit additions, its performance declined when handling even longer sequences.

4.1.4 Zero-padding

Zero-padding ensures that both operands in a query have equal lengths and that the response maintains a fixed length corresponding to the operand length. In practice, padding an M-digit plus an N-digit addition with zeros reformulates the problem so that both operands have $\max\{M, N\}$ digits and the response has $\max\{M, N\} + 1$ digits. For example, the expression (`653 + 49 = 702`) is transformed into (`653 + 049 = 0702`) (Cho et al., 2024).

4.2 Positional Embeddings/Encodings (PE)

Transformers' difficulty in extrapolating to longer sequences is largely attributed to their positional encoding mechanisms (Shaw et al., 2018). In the following section, we examine various positional encoding strategies, with a particular focus on their capacity for length generalization.

4.2.1 Absolute Positional Encoding (APE)

APE incorporates positional information into Transformer models by assigning each position i a unique vector $\mathbf{p}_i$, which is combined with the token embedding (typically by addition) before entering the model. There are two main approaches to generating these position vectors. One approach uses a predefined sinusoidal function to produce periodic embeddings that can naturally extrapolate to unseen positions (Vaswani et al., 2023). The alternative is to learn the position embeddings jointly with the model parameters, as seen in works such as (Devlin et al., 2019; Brown et al., 2020; Zhang

et al., 2022).

Although APE offers a simple and effective mechanism for encoding position, both variants have limitations in generalizing to longer sequences. The learned version, in particular, is restricted to a fixed context window, which can hinder performance on inputs longer than those seen during training (Press et al., 2022; Kazemnejad et al., 2023).

4.2.2 Additive Relative Positional Encoding (RPE)

RPE enhances the self-attention mechanism by incorporating a position-dependent bias into the pre-softmax attention logits. Originally introduced by (Shaw et al., 2018), this approach modifies the keys (and optionally the values) in each attention layer. T5 further advanced the concept by mapping the relative distance between tokens to a scalar bias using a lookup table; this bias is then added to the dot product of queries and keys (Raffel et al., 2023).

More recent methods build on this idea by proposing different functions for the scalar bias $b(i, j)$, which depends on the distance between positions i and j. For example, Alibi (Press et al., 2022) subtracts a bias that grows linearly with the token distance to induce a recency bias, KerpleLog (Chi et al., 2022) uses a logarithmic function, and FIRE (Li et al., 2024) employs a learnable MLP-based function to compute $b(i, j)$. In general, the modified attention logits can be expressed as:

$$A_{\text{RPE}}(X) = XW_Q(XW_K)^\top + B,$$

where X, W_Q, and W_K denote the input and weight matrices for queries and keys, and the bias matrix $B \in \mathbb{R}^{n\times n}$ is determined by the function $b(i, j)$.

4.2.3 Position Coupling

Position coupling assigns position IDs that encode the structure of a task by leveraging the inherent grouping of tokens. The method involves two key steps:

1. **Token Partitioning:** The input sequence is divided into groups of consecutive tokens where each token within a group carries a unique semantic meaning. This grouping enables a one-to-one correspondence between tokens across different groups that are relevant to the task.

2. **Position ID Assignment:** For each group, a sequence of consecutive numbers (typically positive integers) is assigned as position IDs, beginning from a random number during training or a fixed number during evaluation. Tokens that represent the same significance across different groups are given the same position ID (i.e., their positions are *coupled*).

For example, in a decimal addition task, the expression `653 + 49 = 702` is transformed (via reversal and zero-padding) into a format like `$653 + 049 = 2070$`. Here, tokens are partitioned into three groups: (1) the first operand along with the `'+'` token, (2) the second operand, and (3) the `'='` token together with the sum. Position IDs are then assigned such that digits with the same significance across the operands and the sum receive the same ID. For instance, if the starting position ID is 6, the operands might be labeled 6, 7, and 8, while the sum's digits are labeled in reversed order (e.g., 5, 6, 7, 8), with non-digit symbols receiving IDs based on their adjacency to numerical tokens (see Figure 2).

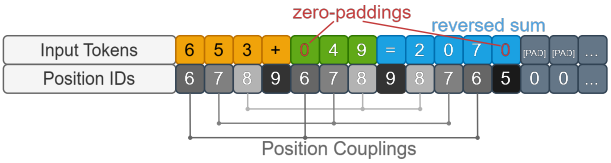

Figure 2: Position coupling for decimal integer addition task, displaying 653 + 49 = 702 with appropriate input formats. The starting position ID '6' is an arbitrarily chosen number (Cho et al., 2024).

4.2.4 Randomized Position Encoding

(Ruoss et al., 2023) introduced Randomized PE, a method that enhances traditional positional encodings by sampling from a range that extends beyond the typical test-time length while maintaining token order. This training strategy enables Transformers to adapt to larger positional encodings, thereby effectively mitigating issues with out-of-distribution position encodings during testing.

4.2.5 No Positional Encoding (NoPE)

Encoder-only Transformers, such as BERT (Devlin et al., 2019), maintain invariance to the order of tokens even without positional encodings. In contrast, decoder-only models using causal attention have been shown by (Haviv et al., 2022) to develop positional awareness on their own without explicit PE. Moreover, (Kazemnejad et al., 2023) have recently demonstrated that, for simple algorithmic tasks, models without any positional encodings can out-

perform those that employ specialized positional encoding techniques.

4.2.6 Rotary Positional Encoding (RoPE)

RoPE, as introduced by (Su et al., 2023), incorporates positional information into the attention logits by applying a rotational transformation to the query and key vectors based on their relative positions. Although this method is both simple and effective, its ability to generalize to longer sequences remains limited (Kazemnejad et al., 2023; Press et al., 2022). Extensions like Position Interpolation (Chen et al., 2023; Peng et al., 2023) can extend the context length of RoPE, but they do not necessarily enhance its generalization performance on algorithmic tasks where understanding the underlying algorithm is critical.

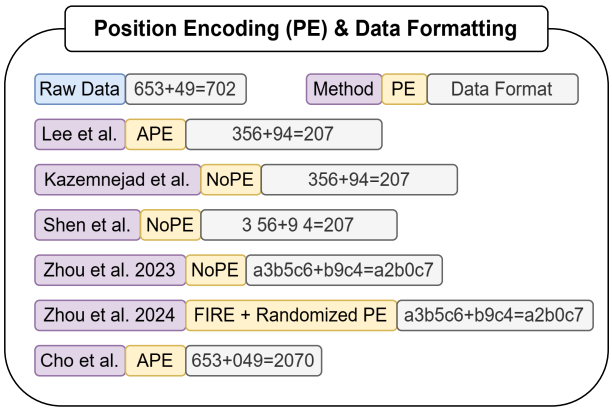

Figure 3: Comparative analysis of PEs and data formats: Unlike most studies that primarily explore APE or NoPE, (Zhou et al., 2024) method combines FIRE (Li et al., 2024) with Randomized PE (Ruoss et al., 2023). All approaches adopt a reversed data format, except for (Cho et al., 2024), which applies the reversed format only to the sum of integers, incorporating zero-padding and position coupling.(Shen et al., 2023) further improve this format by introducing random space augmentation, while both (Zhou et al., 2023) and (Zhou et al., 2024) methods leverage index hints to enhance performance.

5 Results

In our investigation of transformer generalization on the integer addition task, we observe that the ability to extrapolate to sequences longer than those seen during training is highly sensitive to both the choice of positional encoding and the adopted data formatting strategy. Standard Transformers — with conventional absolute positional encodings (APE) — often fail to generalize beyond the training range, typically achieving minima extension (approximately $1\times$) when evaluated on longer-digit addition problems (Kazemnejad et al., 2023). In contrast, several modifications have been proposed to overcome this limitation.

5.1 Enhancing Generalization via Positional Encoding and Data Formatting

Recent work by (Zhou et al., 2024) demonstrates that by carefully tailoring the data format (using a reversed digit order and explicit index hints (Zhou et al., 2023)) and incorporating an expressive positional encoding — namely FIRE (a learned additive bias function) (Li et al., 2024) — a standard transformer trained on addition tasks up to 40 digits can successfully generalize to 100-digit additions. This yields an extension ratio of approximately $2.5\times$. The study shows that the combination of reversed formatting (which aligns the computation order with the natural progression of carry propagation (Zhou et al., 2023)) and index hints (which help the model pinpoint the relevant digit positions (Olsson et al., 2022)) plays a crucial role in this improvement.

Similarly, (Jelassi et al., 2023) explore the use of relative positional encodings. Their findings indicate that for simple tasks such as addition, training on 5-digit numbers can lead to correct 15-digit computations — implying a roughly $3\times$ extension. The relative positional framework appears to mitigate the model's dependency on absolute token positions by focusing on the invariant relations between digits.

5.2 Leveraging Task Structure: Position Coupling and Structural Symmetry

More aggressive approaches directly embed the inherent structure of the arithmetic task into the model's design. (Cho et al., 2024) introduce the concept of *position coupling*, where digits of the same significance (for example, all least significant digits across operands) are assigned the same position identifier. This modification allows a 1-layer Transformer, trained on addition problems with operands ranging from 1 to 30 digits, to generalize to problems with up to 200 digits—corresponding to an extension factor of approximately $6.67\times$. The theoretical analysis further shows that such coupled positional representations are necessary for solving the addition task over exponentially many digits.

In another line of work, (Sabbaghi et al., 2024) propose explicitly encoding the inherent structural symmetry of arithmetic problems. By modifying

Table 2: A compact comparison of approaches for improving length generalization in arithmetic Transformers. The table lists various methods along with their training ranges (in digits), the maximum sequence length to which they generalize, and the corresponding extension factors, indicating the multiplicative improvement over the training length.

Approach	Training Range	Generalizes to	Extension Factor
Standard APE (Baseline)	Up to 40	$\sim$40	1–1.125$\times$
FIRE + Reversed + Index Hints	Up to 40	$\sim$100	2.5$\times$
Relative Positional Encoding	Up to 5	$\sim$15	3$\times$
Position Coupling	1–30	$\sim$200	6.67$\times$
Explicit Structural Symmetry	Up to 5	$\sim$50	10$\times$

the number formatting and designing custom positional encodings that capture the right-to-left symmetry (i.e., aligning digits by their significance), their method enables a Transformer trained on numbers with at most 5 digits to successfully perform 50-digit addition. This approach achieves an impressive 10$\times$ extension and emphasizes that when task-specific structure is explicitly incorporated, the model can overcome the limitations of conventional encoding schemes.

Table 2 summarizes the key approaches discussed above along with their main mechanisms, the training range, and the resulting length extension factor.

6 Discussion

Collectively, these studies indicate that while a standard Transformer architecture is prone to overfitting the training length distribution, targeted modifications to positional encoding and data representation can dramatically boost length generalization. For example, incorporating task-specific modifications —— such as reversed token orders, explicit index hints, and position coupling —— has enabled models to generalize to lengths 2.5–6.67$\times$ beyond their training range. At the same time, our observations of the grokking phenomenon (i.e., a sudden phase change from memorization to generalization; see (Nanda et al., 2023)) reveal that such improvements come with trade-offs. Many enhanced methods remain sensitive to initialization and training order, and the robustness of generalization varies substantially across different random seeds (Zhou et al., 2024). Together, these findings underscore the need for further work on improving stability and understanding the precise mechanisms by which structural and representational modifications affect both memorization and generalization.

7 Conclusion

The summary of this paper, as illustrated throughout our analysis, underscores the evolving role of Transformer models in advancing beyond pattern recognition to algorithmic reasoning. By exploring the generalization capabilities of Transformers, this study distinguishes between length generalization and compositional generalization.

The survey highlights key methodologies that have been proposed to enhance Transformer length generalization in the integer addition task, including advanced data formatting strategies and specialized positional encoding techniques. These approaches, such as position coupling and structural symmetry encoding, represent incremental advancements that progressively improve model performance in out-of-distribution settings.

Despite these advancements, significant research opportunities remain in improving the robustness and adaptability of Transformers, particularly in scaling their generalization to broader, more complex tasks. Transformer-based reasoning is also extending into new domains, including large-scale symbolic computation and real-world problem-solving, requiring novel adaptations to handle dynamic and structured data effectively. This expansion highlights the growing practical implications of Transformer generalization, drawing increasing attention from both academic and industry sectors.

As research in Transformer-based generalization continues to evolve, there is a pressing need to refine evaluation methodologies that can accurately measure reasoning-oriented extrapolation. Establishing rigorous and representative benchmarks will be essential in fully capturing the contributions of these models to AI research and their potential for real-world deployment.

8 Limitations

Despite the comprehensive analysis presented in this paper, several limitations must be acknowledged:

- **Task Scope:** Our survey primarily focuses on controlled algorithmic tasks —– such as integer addition —– to illustrate the challenges of Transformer generalization. While these tasks provide clear insights into the mechanisms of length and compositional generalization, they may not fully capture the complexities encountered in real-world applications.

- **Emphasis on Length Generalization:** Although our taxonomy differentiates between length and compositional generalization, the discussion and empirical focus have largely centered on length generalization. The dynamics of compositional generalization, particularly in the context of large-scale reasoning models, require further in-depth exploration.

- **Sensitivity to Experimental Settings:** Many of the techniques reviewed, including specific data formatting strategies and positional encoding modifications, are sensitive to factors such as random initialization, training order, and hyperparameter choices. This sensitivity may limit the reproducibility and robustness of the reported improvements across diverse settings.

- **Evolving Landscape:** The field of Transformer research is rapidly evolving. New architectural innovations and training methodologies continue to emerge, which may not be fully captured in our current analysis. Future work will need to continuously update the survey framework to integrate these advancements.

References

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. *Preprint*, arXiv:2207.04901.

Reza Bayat, Mohammad Pezeshki, Elvis Dohmatob, David Lopez-Paz, and Pascal Vincent. 2024. The pitfalls of memorization: When memorization hurts generalization.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Shouyuan Chen, Sherman Wong, Liangjian Chen, and Yuandong Tian. 2023. Extending context window of large language models via positional interpolation. *Preprint*, arXiv:2306.15595.

Ta-Chung Chi, Ting-Han Fan, Peter J. Ramadge, and Alexander I. Rudnicky. 2022. Kerple: Kernelized relative positional embedding for length extrapolation. *Preprint*, arXiv:2205.09921.

Hanseul Cho, Jaeyoung Cha, Pranjal Awasthi, Srinadh Bhojanapalli, Anupam Gupta, and Chulhee Yun. 2024. Position coupling: Improving length generalization of arithmetic transformers using task structure. *Preprint*, arXiv:2405.20671.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. *Preprint*, arXiv:1810.04805.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.

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

Yann Dubois, Gautier Dagan, Dieuwke Hupkes, and Elia Bruni. 2020. Location Attention for Extrapolation to Longer Sequences. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 403–413, Online. Association for Computational Linguistics.

Nouha Dziri, Ximing Lu, Melanie Sclar, Xiang Lorraine Li, Liwei Jiang, Bill Yuchen Lin, Peter West, Chandra Bhagavatula, Ronan Le Bras, Jena D. Hwang, Soumya Sanyal, Sean Welleck, Xiang Ren, Allyson Ettinger, Zaid Harchaoui, and Yejin Choi. 2023. Faith and fate: Limits of transformers on compositionality. *Preprint*, arXiv:2305.18654.

Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda

Askell, Yuntao Bai, Anna Chen, Tom Conerly, Nova DasSarma, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. 2021. A mathematical framework for transformer circuits. *Transformer Circuits Thread*. Https://transformer-circuits.pub/2021/framework/index.html.

Xinyu Guan, Li Lyna Zhang, Yifei Liu, Ning Shang, Youran Sun, Yi Zhu, Fan Yang, and Mao Yang. 2025. rstar-math: Small llms can master math reasoning with self-evolved deep thinking. *Preprint*, arXiv:2501.04519.

Adi Haviv, Ori Ram, Ofir Press, Peter Izsak, and Omer Levy. 2022. Transformer language models without positional encodings still learn positional information. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 1382–1390, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

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

John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. 2021. Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589.

Amirhossein Kazemnejad, Inkit Padhi, Karthikeyan Natesan Ramamurthy, Payel Das, and Siva Reddy. 2023. The impact of positional encoding on length generalization in transformers. *Preprint*, arXiv:2305.19466.

Tanishq Kumar, Zachary Ankner, Benjamin F. Spector, Blake Bordelon, Niklas Muennighoff, Mansheej Paul, Cengiz Pehlevan, Christopher Ré, and Aditi Raghunathan. 2024. Scaling laws for precision. *Preprint*, arXiv:2411.04330.

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

Shanda Li, Chong You, Guru Guruganesh, Joshua Ainslie, Santiago Ontanon, Manzil Zaheer, Sumit Sanghai, Yiming Yang, Sanjiv Kumar, and Srinadh Bhojanapalli. 2024. Functional interpolation for relative positions improves long context transformers. *Preprint*, arXiv:2310.04418.

Yiheng Liu, Tianle Han, Siyuan Ma, Jiayue Zhang, Yuanyuan Yang, Jiaming Tian, Hao He, Antong Li, Mengshen He, Zhengliang Liu, et al. 2023. Summary of chatgpt-related research and perspective towards the future of large language models. *arXiv preprint arXiv:2304.01852*.

Mohammad Motamedi, Nikolay Sakharnykh, and Tim Kaldewey. 2021. A data-centric approach for training deep neural networks with less data. *Preprint*, arXiv:2110.03613.

Neel Nanda, Lawrence Chan, Tom Lieberum, Jess Smith, and Jacob Steinhardt. 2023. Progress measures for grokking via mechanistic interpretability. *Preprint*, arXiv:2301.05217.

Maxwell Nye, Anders Johan Andreassen, Guy Gur-Ari, Henryk Michalewski, Jacob Austin, David Bieber, David Dohan, Aitor Lewkowycz, Maarten Bosma, David Luan, Charles Sutton, and Augustus Odena. 2021. Show your work: Scratchpads for intermediate computation with language models. *Preprint*, arXiv:2112.00114.

Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, Nova DasSarma, Tom Henighan, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Scott Johnston, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. 2022. In-context learning and induction heads. *Transformer Circuits Thread*. Https://transformer-circuits.pub/2022/in-context-learning-and-induction-heads/index.html.

Bowen Peng, Jeffrey Quesnelle, Honglu Fan, and Enrico Shippole. 2023. Yarn: Efficient context window extension of large language models. *Preprint*, arXiv:2309.00071.

Ofir Press, Noah A. Smith, and Mike Lewis. 2022. Train short, test long: Attention with linear biases enables input length extrapolation. *Preprint*, arXiv:2108.12409.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2023. Exploring the limits of transfer learning with a unified text-to-text transformer. *Preprint*, arXiv:1910.10683.

Anian Ruoss, Grégoire Delétang, Tim Genewein, Jordi Grau-Moya, Róbert Csordás, Mehdi Bennani, Shane Legg, and Joel Veness. 2023. Randomized positional encodings boost length generalization of transformers. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 1889–1903, Toronto, Canada. Association for Computational Linguistics.

Mahdi Sabbaghi, George Pappas, Hamed Hassani, and Surbhi Goel. 2024. Explicitly encoding structural symmetry is key to length generalization in arithmetic tasks. *Preprint*, arXiv:2406.01895.

David Saxton, Edward Grefenstette, Felix Hill, and Pushmeet Kohli. 2019. Analysing mathematical reasoning abilities of neural models. *Preprint*, arXiv:1904.01557.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *Preprint*, arXiv:2402.03300.

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. 2018. Self-attention with relative position representations. *Preprint*, arXiv:1803.02155.

Ruoqi Shen, Sébastien Bubeck, Ronen Eldan, Yin Tat Lee, Yuanzhi Li, and Yi Zhang. 2023. Positional description matters for transformers arithmetic. *Preprint*, arXiv:2311.14737.

Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. 2023. Roformer: Enhanced transformer with rotary position embedding. *Preprint*, arXiv:2104.09864.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2023. Attention is all you need. *Preprint*, arXiv:1706.03762.

Petar Veličković, Adrià Puigdomènech Badia, David Budden, Razvan Pascanu, Andrea Banino, Misha Dashevskiy, Raia Hadsell, and Charles Blundell. 2022. The CLRS algorithmic reasoning benchmark. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 22084–22102. PMLR.

Jie Wang, Tao Ji, Yuanbin Wu, Hang Yan, Tao Gui, Qi Zhang, Xuanjing Huang, and Xiaoling Wang. 2024. Length generalization of causal transformers without position encoding. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 14024–14040, Bangkok, Thailand. Association for Computational Linguistics.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. 2022. Opt: Open pre-trained transformer language models. *Preprint*, arXiv:2205.01068.

Yi Zhang, Arturs Backurs, Sébastien Bubeck, Ronen Eldan, Suriya Gunasekar, and Tal Wagner. 2023. Unveiling transformers with lego: a synthetic reasoning task. *Preprint*, arXiv:2206.04301.

Zhenru Zhang, Chujie Zheng, Yangzhen Wu, Beichen Zhang, Runji Lin, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. 2025. The lessons of developing process reward models in mathematical reasoning. *Preprint*, arXiv:2501.07301.

Hattie Zhou, Arwen Bradley, Etai Littwin, Noam Razin, Omid Saremi, Josh Susskind, Samy Bengio, and Preetum Nakkiran. 2023. What algorithms can transformers learn? a study in length generalization. *Preprint*, arXiv:2310.16028.

Yongchao Zhou, Uri Alon, Xinyun Chen, Xuezhi Wang, Rishabh Agarwal, and Denny Zhou. 2024. Transformers can achieve length generalization but not robustly. *Preprint*, arXiv:2402.09371.

A Decoder-Only Transformer Architecture

The decoder-only Transformer is an autoregressive model designed to predict each new token based on the tokens observed so far. Its key feature is the use of *causal masked self-attention*, which prevents the model from accessing future tokens in the sequence. This ensures that predictions are based solely on current and past information, making the model suitable for next-token prediction tasks such as language modeling.

Each decoder layer in the architecture is usually composed of two main components:

1. **Masked Self-Attention:** Enables the model to attend only to preceding tokens and itself, ensuring the causal property required for autoregressive modeling.

2. **Feed-Forward Neural Network (FFN):** Applies a position-wise nonlinear transformation to enhance the expressiveness of token representations.

During training, the model uses *teacher forcing*, where ground-truth tokens are provided as inputs at each step. The training objective is to minimize the cross-entropy loss between the predicted token probabilities and the actual tokens in the sequence.

The addition example in Figure 3 illustrates how this architecture can be applied to sequence prediction tasks, highlighting the step-by-step generation of the output tokens.

B Taxonomy

B.1 Data Distribution

Data Distribution classifies the degree to which a model's learned representations and decision boundaries remain valid under conditions that deviate from the training distribution. In algorithmic tasks, this dimension is particularly critical when inputs vary in length, structure, or domain, as shifts in these aspects often reveal the difference between memorization and generalizable knowledge.

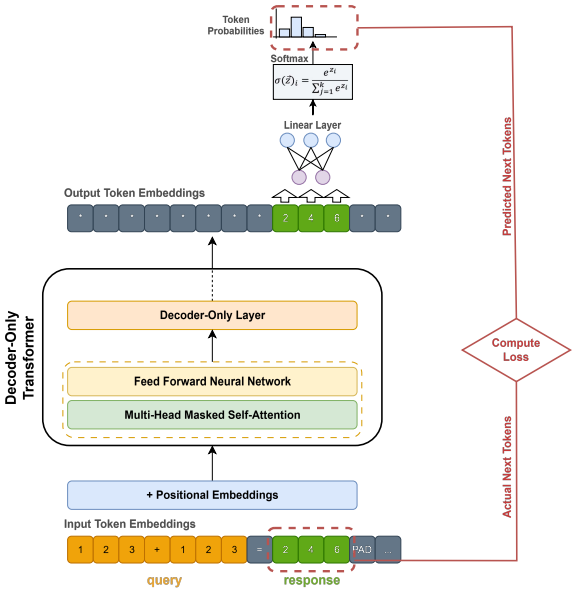

Figure 4: Schematic representation of a decoder-only Transformer. Each input token embedding is processed through a stack of decoder-only layers, consisting of masked self-attention and feed-forward neural network (FFN) sub-layers. The architecture generates output representations for each token, with a linear layer followed by a softmax function computing the probabilities of the next token in the sequence.

- *In-Distribution (ID) Performance:* Assesses how well a model performs when the evaluation data closely matches the training set in terms of length, format, or domain. High ID accuracy can be achieved through memorized patterns, so it does not by itself guarantee robust extrapolation.
- *Out-of-Distribution (OOD) Performance:* Evaluates the model's ability to handle input distributions that differ significantly from training such as much longer sequences, unfamiliar data formats, or novel domains. Strong OOD performance is a key indicator of true algorithmic understanding, instead of depending on superficial patterns or memorized examples.

B.2 Scratchpads / Chain-of-Thought (CoT)

Scratchpads and Chain-of-Thought (CoT) prompting enable models to express their reasoning through intermediate computational steps. While beneficial for both generalization types, they primarily enhance **compositional generalization** by providing explicit ground for novel combinations of learned operations. For length generalization, scratchpads help manage positional dependencies (e.g., tracking carries in long addition via induction mechanism (Zhou et al., 2023)), but extend an already internalized algorithm. In contrast, compositional tasks leverage CoT to *dynamically structure* solutions — for instance, breaking nested arithmetic expressions into executable sub-steps. This explicit step-by-step reasoning mimics human problem-solving strategies, transforming abstract capability into verifiable algorithmic reasoning.