# Thesis Research (I)

**Zhenrui Zheng**

Chinese University of Hong Kong, Shenzhen
`225040512@link.cuhk.edu.cn`

# Contents

# Research Topic

## 1.1 Background

The advent of Large Language Models (LLMs) has fundamentally shifted the paradigm of artificial intelligence from supervised pattern recognition to In-Context Learning (ICL) and sequential decision-making. Recent advancements, such as the Searchformer" architecture, suggest that Transformers can transcend simple sequence prediction by internalizing the dynamics of search algorithms like A*. However, a critical bottleneck persists: the inability to achieve "length generalization" the capacity to solve problems with longer horizons or higher complexity than those encountered during training. While empirical evidence indicates that reasoning performance is highly sensitive to data distributions and prompting strategies, the mechanistic drivers behind these failures remain under-explored. Specifically, the influence of structural inductive biases (such as positional encodings) and the qualitative nature of reasoning traces (Chain-of-Thought) on the model's internal "world model" is not fully understood. This research utilizes pathfinding as a canonical proxy to investigate how Transformers represent spatial logic and, crucially, where their reasoning capabilities collapse when pushed beyond training boundaries.

## 1.2 Research Questions

This study seeks to address four core questions regarding the algorithmic capabilities of Transformers:

- **RQ1 (Data Distribution):** How does the distribution of sequence lengths in the training data causally constrain the model's ability to generalize to longer horizons during inference?
- **RQ2 (Structural Bias):** To what extent do different positional encoding schemes influence the simulation of spatial algorithms, and can a novel relative encoding mechanism improve structural reasoning in graph-based tasks?
- **RQ3 (Failure Mechanisms):** What are the primary failure modes (e.g., looping, hallucination, goal drift) when a model attempts to generalize, and what do these patterns reveal about the fragility of its internal state representation?
- **RQ4 (Data Augmentation):** How do data augmentation strategies that modify the reasoning path (Chain-of-Thought) compare to surface-level augmentations in enhancing the robustness of algorithmic generalization?

# Literature Review

## 2.1 Model In-Context Learning

In-context learning (ICL) represents a fundamental shift in how models utilize pre-training data. Dong et al. provide a comprehensive survey of this paradigm, defining ICL as the ability of LLMs to make predictions based on contexts augmented with a few examples, effectively using the context window as a temporary learning buffer [1]. While ICL is empirically powerful, its underlying mechanisms have been subject to intense scrutiny. Min et al. challenge the conventional intuition that models "learn" the mapping from demonstrations in the strict sense. Their work, *Rethinking the Role of Demonstrations*, reveals that ground-truth labels in demonstrations are often unnecessary; instead, the model primarily leverages the label space, input distribution, and formatting of the demonstrations to steer generation [2]. Complementing this empirical analysis, Chan et al. investigate the data-centric origins of this capability. In *Data Distributional Properties Drive Emergent In-Context Learning*, they demonstrate that ICL is not merely an architectural byproduct but an emergent behavior driven by specific properties of the training data, such as "burstiness" and skewed Zipfian distributions, which force the model to infer local context to minimize prediction error [3]. To understand the computational limits of these architectures, Weiss et al. propose a theoretical framework in *Thinking Like Transformers*. They introduce the Restricted Access Sequence Processing (RASP) language, mapping the Transformer's attention and feed-forward mechanisms to programming primitives. This abstraction allows for a precise discussion of what algorithms a Transformer can express, providing a theoretical grounding for the empirical successes of ICL [4].

## 2.2 In-Context Reinforcement Learning (ICRL)

The principles of ICL have been successfully translated to Reinforcement Learning (RL), where the "context" becomes a history of state-action-reward sequences. Laskin et al. introduce *Algorithm Distillation* (AD), proposing that RL algorithms themselves can be distilled into a neural network. By modeling training histories with a causal sequence model, AD allows transformers to improve policies entirely in-context, effectively learning a reinforcement learning algorithm that is more data-efficient than the source algorithm used to generate the data [5]. Similarly, Lee et al. show in *Supervised Pretraining Can Learn In-Context Reinforcement Learning* that a Decision-Pretrained Transformer (DPT) can exhibit complex RL behaviors—such as online exploration and offline conservatism—simply via supervised pretraining on diverse interaction datasets. They theoretically connect this to Bayesian posterior sampling, suggesting that transformers implement efficient posterior inference [6]. Generalization remains a core challenge in this domain. Raparthy et al. explore *Generalization to New Sequential Decision Making Tasks*, highlighting that naive application of transformers often fails in stochastic environments. They echo the findings of Chan et al., showing that training on trajectories with high burstiness and diversity is crucial for enabling models to generalize to unseen tasks like MiniHack and Procgen without weight updates [7]. Bridging planning and learning, Lehnert et al. propose *Searchformer* in *Beyond A**. By training transformers to predict the search dynamics (e.g., the open and closed lists) of the A* algorithm rather than just the optimal path, they demonstrate that models can outperform traditional symbolic planners in complex tasks like Sokoban, effectively “bootstrapping” better planning capabilities via sequence modeling [8].

## 2.3 Model Reasoning Capabilities

The reasoning capacity of Transformers is often scrutinized through the lenses of planning, arithmetic, and length generalization. The distinction between memorization and true generalization is critical. Chu et al. provide a comparative study in *SFT Memorizes, RL Generalizes*, arguing that while Supervised Fine-Tuning (SFT) tends to memorize training distributions, Reinforcement Learning (RL)—particularly with outcome-based rewards—drives true generalization in rule-based and visual reasoning tasks. They suggest SFT is best used to stabilize output formats, while RL is necessary for acquiring robust reasoning logic [9]. In the domain of mathematics, Lee et al. investigate *Teaching Arithmetic to Small Transformers*. They find that standard training data is suboptimal for arithmetic; however, simple formatting changes and Chain-of-Thought (CoT) data can induce sharp phase transitions in learning, enabling even small models to master arithmetic operations from scratch. This work highlights the importance of instructive data compatible with the next-token prediction objective [10]. Finally, a major bottleneck for reasoning is length generalization—the ability to solve harder (longer) problems than those seen during training. Anil et al. explore this in *Exploring Length Generalization in Large Language Models*. They establish that standard fine-tuning fails to generalize to longer sequences. However, they show that combining ICL with “scratchpad” prompting (forcing the model to output intermediate steps) yields dramatic improvements, suggesting that the reasoning failure is often due to a lack of intermediate computation rather than inherent architectural limitations [11].

# Research Overview

This study investigates the mechanistic constraints of Transformer-based models in performing sequential reasoning and planning tasks. Inspired by the work of Lehnert and Tiomkin (2024) on “Searchformer,” which demonstrated that Transformers can learn to predict search algorithm dynamics, this research utilizes grid and non-grid pathfinding problems as a proxy for evaluating general reasoning capabilities. The primary objective is to deconstruct the factors enabling or inhibiting *length generalization*—the ability of a model to solve problems more complex than those seen during training. Through a series of controlled experiments, this work explores four critical dimensions: (1) the causal relationship between training data sequence length distributions and model inference (generalization) performance; (2) the role of positional encoding schemes in representing spatial and graph structures, proposing and evaluating a novel relative positional encoding mechanism; (3) a quantitative analysis of failure modes to map the boundaries of the model’s internal “world model”; and (4) the efficacy of data augmentation strategies, specifically distinguishing between surface-level augmentations and those that modify the Chain-of-Thought (CoT) reasoning path. The findings suggest that while Transformers are capable of simulating search algorithms, their robustness is heavily dependent on specific structural biases introduced by positional encodings and the diversity of the reasoning trajectories (CoT) rather than mere data volume.

## 3.1 Introduction

### Background and Motivation

- The shift from pattern matching to algorithmic reasoning in LLMs.
- Pathfinding as a canonical task for evaluating planning and sequential logic.
- The challenge of length generalization in current Transformer architectures.

### Research Questions

- **RQ1:** How does the distribution of sequence lengths in the training set constrain or enable the model's ability to generalize to longer horizons?
- **RQ2:** To what extent do different positional encoding schemes affect the model's ability to simulate spatial algorithms, and does a novel relative encoding scheme offer improvements?
- **RQ3:** What are the characteristic patterns of failure when the model attempts to generalize, and what do they reveal about its internal state representation?
- **RQ4:** How do different data augmentation strategies—specifically those altering the reasoning trace (CoT)—impact generalization performance?

## 3.2 Methodology

### Task Formulation

We formulate the pathfinding problem as a sequence generation task, where the model must output the reasoning steps and the final solution path given a textual description of the environment. We investigate two distinct topological settings:

**Grid Environment.** The environment is defined as an $N \times N$ grid (e.g., $5 \times 5$, $8 \times 8$), where each node is uniquely identified by its Cartesian coordinates $(x, y)$. The grid contains a set of obstacles (walls) that block traversal. The objective is to find the shortest path from a start node $s$ to a target node $t$ avoiding obstacles.

**Non-Grid Graph Environment.** To evaluate generalization beyond Euclidean structures, we utilize a graph environment consisting of discrete nodes (typically $|V| = 8$). In the baseline setting, nodes are labeled with sequential integers $0 \sim 7$. To test the model's robustness to token representation, subsequent experiments introduce variations such as shuffled identifiers or randomly assigned large-integer IDs, thereby decoupling the node identity from its ordinal position.

### Sequence Representation

Following the "Searchformer" paradigm, we serialize the problem instance and the algorithm's execution trace into a unified token sequence. The sequence consists of two distinct segments: the *Prompt* and the *Event Trace*.

**Prompt (Problem Definition).** The prompt encodes the static environment and the query. It is structured as a flattened list of attributes:

$$\mathcal{P} = [\texttt{start}, x_s, y_s, \texttt{goal}, x_t, y_t, \texttt{wall}, x_{w1}, y_{w1}, \dots]$$

This segment provides the full observability required for the planning task.

**Event Trace (Reasoning and Solution).** The event trace represents the dynamic execution of the pathfinding algorithm (specifically Dijkstra's algorithm). It serves as the Chain-of-Thought (CoT) for the model. The sequence begins with a special start-of-sequence token (`bos`) appended to the prompt. The trace includes:

- **Algorithm Dynamics:** Tokens representing the internal state changes of the search algorithm. We use `create` to denote adding a node to the priority queue (Open Set) and `close` to denote visiting a node (Closed Set), accompanied by their coordinates and current distance values.
- **Solution Path:** Once the target is reached, the model generates the optimal path sequence, denoted by `path` tokens followed by the coordinate sequence, terminating with an end-of-sequence (`eos`) token.

Formally, the model processes the concatenated sequence $S = [\mathcal{P}; \texttt{bos}; \mathcal{E}]$, where $\mathcal{E}$ is the event trace.

### Training Objective

We employ a decoder-only Transformer architecture trained via standard teacher-forcing. The model optimizes the autoregressive log-likelihood of the event trace given the prompt:

$$\mathcal{L}(\theta) = -\sum_{t=1}^{T} \log P_\theta(x_t \mid x_{<t}, \mathcal{P})$$

where $x_t$ represents the tokens in the event trace. During inference, the model is provided with the Prompt and the `bos` token, and it must generate the subsequent reasoning steps and the final path.

### Model Architecture

Unlike the encoder-decoder architecture with learned absolute positional embeddings used in prior work (e.g., BART in Lehnert & Tiomkin, 2024), we adopt the **Llama** architecture, a decoder-only Transformer. The choice of architecture is critical for length generalization. Absolute positional encodings (as used in T5 or standard BERT/GPT) bind the model's learned representations to specific integer indices seen during training. This creates a "horizon barrier," preventing the model from operating on sequences longer than the maximum training length. To mitigate this, our baseline model utilizes **Rotary Positional Embeddings (RoPE)**. RoPE encodes position information by rotating the query and key vectors in the attention mechanism. Theoretically, this allows for better extrapolation, as the attention score depends on the relative distance between tokens rather than their absolute positions. Unless otherwise stated, all subsequent experiments utilize the Llama architecture equipped with RoPE.

## 3.3 Theoretical Foundations: Computation and Induction (RQ0)

Before analyzing the empirical failures of Transformers in pathfinding, it is crucial to frame the problem through the lens of computational complexity and algorithmic information theory. This theoretical groundwork provides the necessary context for interpreting why models struggle with length generalization and why data augmentation strategies behave as they do.

## The Computational Budget of Transformers

A fundamental limitation of the standard Transformer architecture is its fixed computational depth. For a model with $L$ layers, any single forward pass represents a constant-time $O(1)$ computation (or more precisely, a circuit of fixed depth).

- **The Mismatch:** Many algorithmic tasks, including Breadth-First Search (BFS), require computation time that scales linearly or super-linearly with the input size (e.g., graph diameter).
- **The Role of Chain-of-Thought (CoT):** Attempting to output the final answer (e.g., the shortest path distance) directly from the prompt forces the model to compress a linear-time algorithm into a constant-depth circuit. This is theoretically impossible for arbitrary inputs, leading to the "Computational Gap" described in recent literature. The autoregressive generation of a CoT (the *Event* trace in our dataset) is not just an interpretability tool; it is a computational necessity. It unfolds the $O(1)$ circuit into a sequential $O(T)$ process, granting the model the linear time budget required to execute the algorithm.

Therefore, our experiments focus strictly on the model's ability to generate the step-by-step trace, as this is the only regime where algorithmic generalization is theoretically plausible.

## Solomonoff Induction and the "Window Effect"

Even when provided with a CoT, why do models often learn position-dependent heuristics (the "Window Effect") instead of the true graph traversal algorithm? This can be viewed through the lens of Solomonoff Induction.

- **The Learning Objective:** Training a Transformer is essentially searching for a program (encoded in weights $\theta$) that minimizes the description length of the training data $D$. According to Occam's Razor, the model prefers the simplest program that fits the data.
- **The Ambiguity of Finite Data:** On a finite dataset of small grids (e.g., $5 \times 5$), the "True Algorithm" (BFS) is not the only solution. There exist many "Approximate Programs"—such as "output `path_found` after $k \approx 10$ steps"—that may fit the training distribution equally well or even more efficiently (lower complexity penalty) than the rigorous BFS logic.
- **Spurious Correlations:** If the training data lacks diversity (e.g., all paths are roughly the same length), the "Approximate Programs" based on positional heuristics becomes a statistically valid shortcut. The model converges to this local minimum because it is a simpler explanation for the observed data than the general algorithm.

## The Role of Stochasticity

This theoretical perspective also elucidates the high variance observed across random seeds (discussed in RQ3). The loss landscape contains multiple basins of attraction: some correspond to the robust "True Algorithm," while others correspond to fragile "Approximate Programs."

- Since both solutions yield low loss on the training set, the optimizer's trajectory is heavily influenced by the initial random weights.

- A "lucky" seed might initialize the weights closer to the algorithmic solution, while an "unlucky" seed might gravitate towards the positional heuristic.

This suggests that generalization in LLMs is not solely a function of architecture or data size, but also a probabilistic outcome of the optimization dynamics, where the model selects one of many valid programs that explain the limited observations.

## 3.4 The Impact of Sequence Length Distribution (RQ1)

In this section, we investigate how the distribution of sequence lengths in the training data constrains the model's reasoning capabilities. It is important to clarify that "length" here refers to the total number of tokens in the sequence (Prompt + Event Trace). While we maintain a fixed grid size (e.g., $8 \times 8$) within each comparative experiment, the variations in obstacle placement (walls) and start/goal positions naturally yield a wide variance in the complexity of the pathfinding problem, and consequently, the length of the reasoning trace. We conducted three controlled experiments to isolate the effects of length distribution.

### Experiment 1: Extrapolation to Longer Horizons

We first examined the standard length generalization setting: training on short sequences and testing on longer ones. The model was trained on a dataset where sequence lengths were strictly bounded by an upper limit $L_{train}$. We then evaluated its performance on test cases requiring significantly longer reasoning traces ($L_{test} > L_{train}$). **Observation:**

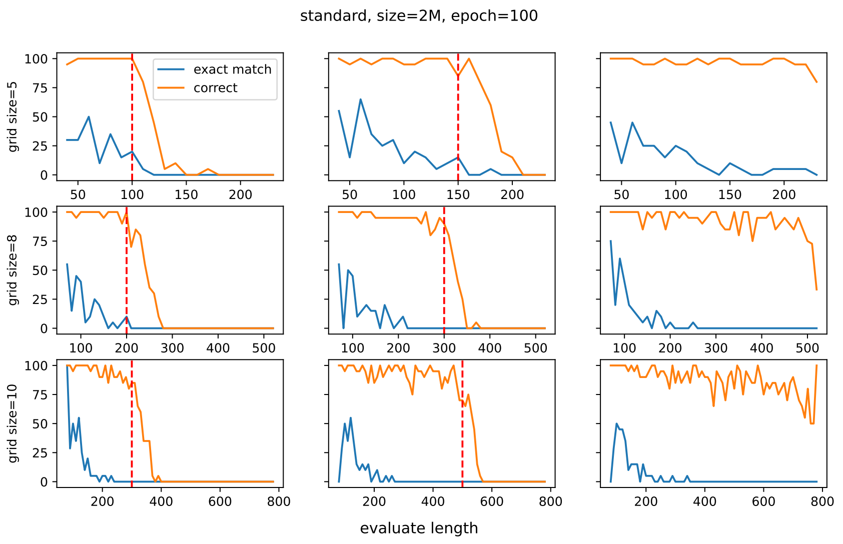


Figure 3.1: Success rate decay on out-of-distribution long sequences. The vertical line indicates the maximum length seen during training.

As shown in Figure 3.1, the model's success rate does not drop to zero immediately upon exceeding the training length. Instead, it exhibits a *gradual decay*. This indicates that the model has learned the local algorithmic rules (e.g., valid node transitions) but struggles to maintain global coherence as the sequence extends into unfamiliar positional territory.

### Experiment 2: The "Priming" Effect

To understand if the failure in long sequences is due to a lack of capacity or a lack of positional calibration, we introduced a "Priming" experiment.

- **Method:** We first trained the model on the short dataset (as in Exp 1). Then, we fine-tuned it on a very small set of long sequences.

- **Control:** A model trained *only* on the small set of long sequences from scratch.

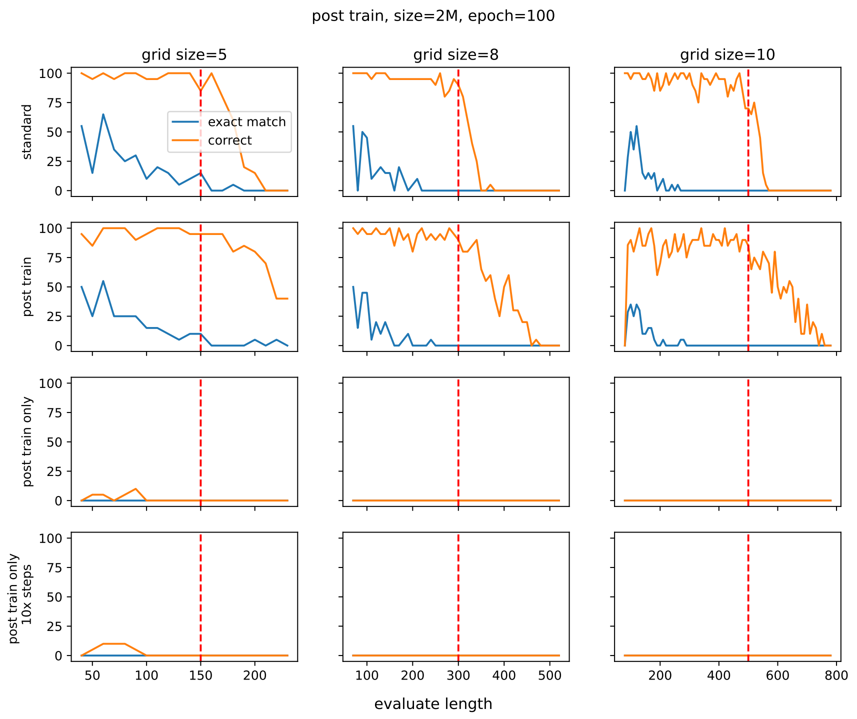


Figure 3.2: Comparison of success rates between the Primed model and the Control model.

**Observation:** Figure 3.2 demonstrates that the "Primed" model rapidly generalizes to the long sequences, achieving high accuracy. In contrast, the Control model fails to converge. This suggests that the "algorithm" (the logic of Dijkstra's search) is learned efficiently from the abundant short data, while the long data merely serves to calibrate the model's positional biases to a larger context window.

## Experiment 3: The "Window" Effect

Finally, we investigated a "Middle-Out" training scenario, a setting rarely explored in current LLM literature which typically focuses solely on context extension. We trained the model on a dataset of "medium" length sequences (e.g., $L \in [L_{min}, L_{max}]$) and tested it on both longer ($L > L_{max}$) and shorter ($L < L_{min}$) sequences. **Observation:** As

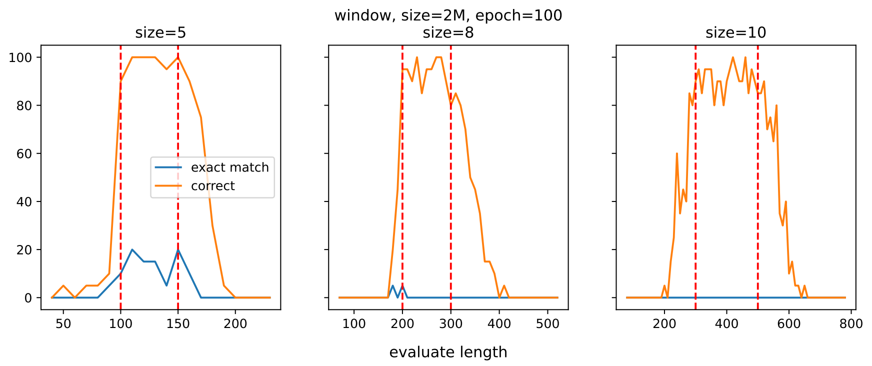


Figure 3.3: The "Window Effect": Performance degradation on both shorter and longer sequences compared to the training distribution.

illustrated in Figure 3.3, we observed a bell-shaped performance curve. The model's performance degrades not only when extrapolating to longer sequences but also when operating on sequences *shorter* than those seen during training.

### Mechanistic Interpretation

The results from these three experiments suggest that the model's reasoning is heavily entangled with its positional encoding.

1. **Positional Overfitting:** The "Window Effect" implies that the model does not just learn what to do next, but what to do next *at this specific position.* If the training data suggests that the solution path typically emerges around token index 500, the model suppresses the generation of solution tokens at index 100, leading to failure on short sequences.
2. **Algorithmic vs. Positional Learning:** The success of the Priming experiment indicates a decoupling of skills. The *algorithmic logic* (local transition rules) is position-agnostic and learned from the bulk of the data. The *global planning capability*, however, is constrained by the positional embeddings (RoPE). The model requires exposure to the specific range of position indices to "unlock" the ability to apply its algorithmic logic in those regions.
3. **Bias Accumulation:** The gradual decay in Exp 1 suggests that errors are cumulative. As the sequence grows beyond the training distribution, the attention mechanism's precision likely drifts, causing the model to lose track of the "Open Set" state, eventually leading to a breakdown in the search process.

## 3.5 Positional Encodings and Structural Reasoning (RQ2)

### The Positional Bias Hypothesis

A fundamental requirement for robust algorithmic execution is *translation invariance.* The next step in a pathfinding algorithm should depend solely on the current state of the search, not on the absolute index of the current token. However, our experiments reveal a persistent **Positional Bias** in standard Transformer architectures. Even with RoPE, models tend to overfit to specific absolute positions, leading to performance degradation when inference sequences fall outside the length range seen during training. This suggests the model learns heuristics tied to specific indices (e.g., token 50 usually implies a "close" operation) rather than the underlying causal dynamics.

### Proposed Method: Pair-Wise Rotary Embedding (PWRE)

To address this, we introduce **Pair-Wise Rotary Embedding (PWRE)**. The core motivation is to bound the positional input space. In standard schemes, a longer sequence introduces larger position indices (e.g., $pos = 2048$) that the model may never have encountered during training. PWRE eliminates this by calculating the rotation for every pair of tokens dynamically and normalizing these relative distances to the closed interval $[0, 1]$.

- **Mechanism:** For any two tokens at indices $i$ and $j$, the rotation is derived from a normalized distance $\delta'_{ij} = \frac{|i-j|}{L_{max}}$, where the result is strictly bounded between 0 and 1.
- **Theoretical Advantage:** By compressing all positional information into the $[0, 1]$ range, the model theoretically never encounters an "out-of-distribution" position value during inference, regardless of the actual sequence length. This ensures that the attention mechanism operates purely on relative proximity within a fixed numerical domain.

### Experimental Outcome and Limitations

Despite the promising theoretical properties of PWRE, practical implementation presented significant hurdles:

- **Computational Complexity:** PWRE requires computing a unique rotation matrix for every pair of tokens, scaling quadratically $O(N^2)$ or linearly $O(N)$ per step during generation. This is computationally expensive compared to the $O(1)$ lookup of standard RoPE.
- **Optimization Difficulties:** The dynamic, pair-wise nature of the computation is incompatible with standard FlashAttention kernels, leading to prohibitive training times. Furthermore, the model struggled to converge, likely due to the difficulty of learning stable representations when relative distances are continuously rescaled as the sequence grows.

Given these challenges, we paused the architectural modifications to focus on data-centric approaches for improving generalization.

## 3.6 Failure Mode Analysis (RQ3)

To understand the mechanistic causes of the "Window Effect" observed in RQ1, we conducted a granular analysis of the model's token-level predictions. Specifically, we investigated whether the model's failures were uniformly distributed across all token types or concentrated on specific algorithmic transitions.

### Perplexity Matrix Analysis

We employed a Perplexity Matrix visualization to audit the model's probability distribution.

- **Method:** We took the model trained on medium-length sequences (the "Window" model) and evaluated it on three test sets: *Shorter-than-Training*, *In-Distribution*, and *Longer-than-Training*.
- **Metric:** For each position in the test sequences, we fed the ground truth prefix to the model and recorded the output probability distribution. We then aggregated these distributions to form a confusion matrix where the y-axis represents the Ground Truth Token class, and the x-axis represents the Model's Predicted Probability distribution (averaged across samples). Darker colors on the diagonal indicate confident, correct predictions.

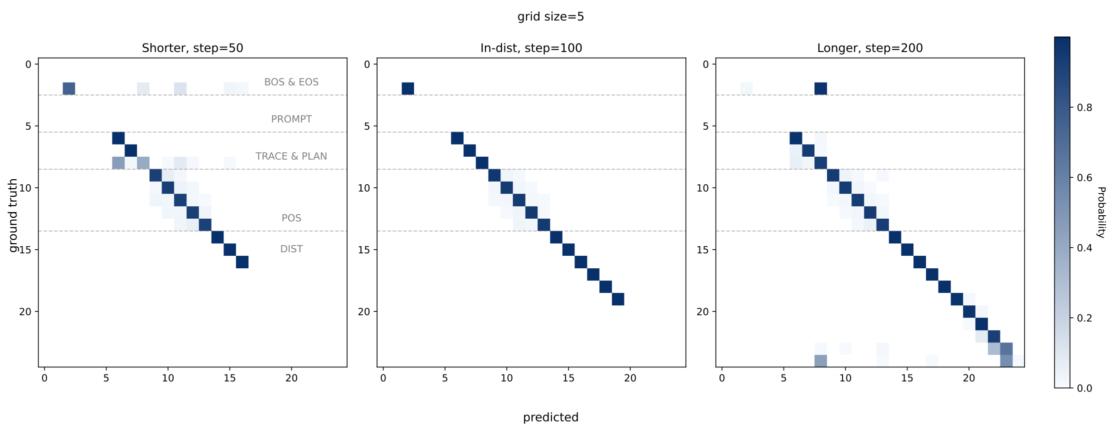
### Key Observations

As illustrated in Figure 3.4, the behavior varies drastically across the three regimes:

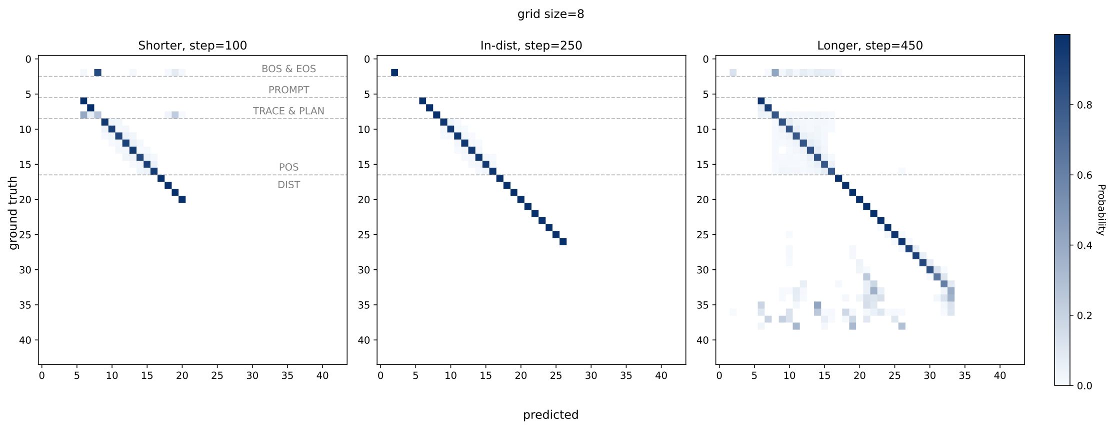
**In-Distribution (Center).** The matrix shows a sharp diagonal, indicating that the model has perfectly mastered the algorithmic logic within the training length range.

**Out-of-Distribution (Left & Right).** Significant off-diagonal noise emerges, but the errors are not random.

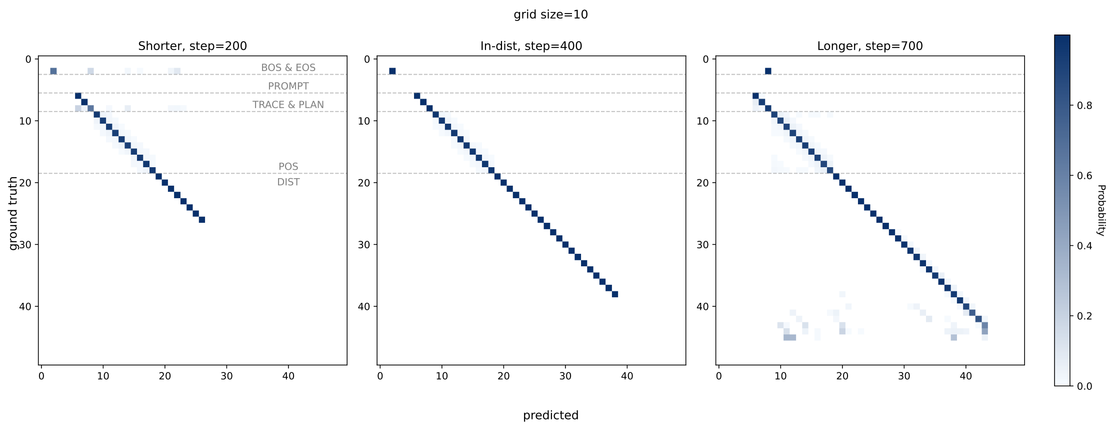
1. **The "Termination" Bias:** The most prominent failure mode occurs when the ground truth requires the model to transition to the solution phase (e.g., outputting `path` or `eos` tokens).



(a) Grid Size Small (5 × 5)



(b) Grid Size Medium (8 × 8)



(c) Grid Size Large (10 × 10)

Figure 3.4: Perplexity Matrices across three different grid sizes. Each subfigure displays the confusion matrices for Short (Left), In-Distribution (Center), and Long (Right) test sequences. The y-axis represents the ground truth token, and the x-axis shows the predicted probability distribution.

- In *Shorter* sequences, when the algorithm finishes early, the model fails to predict the termination tokens, instead assigning high probability to continuation tokens (like `create` or `close`).
- This strongly supports our hypothesis: the model has learned a spurious correlation between the *absolute position* and the *termination of the algorithm*. It expects the path to be found only after a certain number of steps (consistent with the training mean), ignoring the actual state of the search graph.

2. **Distance Token Collapse (Longer Sequences):** In the *Longer* regime, we observe a collapse in predicting distance tokens. This is an expected artifact: longer paths generate distance values (integers) that were never present in the training vocabulary. While this is a vocabulary limitation rather than a reasoning failure, it highlights the necessity of number-handling mechanisms (e.g., digit-level tokenization) for infinite generalization, which we address in subsequent experiments.

### The Role of Stochasticity

It is crucial to note that the severity of these failure modes is not deterministic. During our analysis, we discovered that the model's ability to generalize—or its tendency to overfit to position—is highly sensitive to the random seed used for initialization and data shuffling.

- Some seeds produced models that were surprisingly robust to length variations, while others (with identical hyperparameters) exhibited severe "Window Effects."
- This suggests that the "algorithm" learned by the Transformer is not unique; the optimization landscape contains multiple local minima, some of which rely on positional heuristics while others approximate the true causal algorithm. To ensure reliability, all results reported in this paper are averaged across multiple random seeds to filter out these initialization artifacts.

## 3.7 Data Augmentation and Chain-of-Thought (RQ4)

While the previous sections focused on grid-based maps to study length generalization, real-world reasoning often involves arbitrary graph topologies. In this section, we shift our focus to **Non-Grid Graphs** (random graphs with fixed node counts but varying connectivity). This setting allows for faster iteration and enables us to isolate the factors that contribute to **structural generalization**—the ability to solve pathfinding problems on graph structures unseen during training.

### Experimental Setup: Data Augmentation Strategies

In computer vision, data augmentation (e.g., rotation, flipping) is a standard technique to inject inductive biases (invariance) into the model. However, for algorithmic reasoning in LLMs, the optimal augmentation strategy is less understood. We designed a controlled experiment using a base dataset of random graphs (8 nodes). We compared nine different augmentation strategies to determine which factors facilitate better generalization. The strategies are categorized as follows:

- **Baseline:**
  - **1. None:** The standard training set without any modification.
- **Symbolic Remapping (Label-N):** These strategies remap the node indices (originally $0 \sim 7$) to a larger integer range $[0, N]$. This preserves the graph topology and the validity of the path but changes the surface-level tokens.
  - **2. Label-10:** Nodes remapped to $0 \sim 10$.
  - **3. Label-100:** Nodes remapped to $0 \sim 100$.
  - **4. Label-1000:** Nodes remapped to $0 \sim 1000$.
- **Prompt Permutation:**
  - **5. Edge-Order:** Randomly shuffles the order of edges presented in the prompt description. This does not affect the graph structure or the solution path.

- **Problem Resampling (CoT Diversity):** These strategies involve altering the problem definition, which necessitates generating a *new* Event Trace (Chain of Thought).
  - **6. Edge-Weight:** Re-assigns random weights to existing edges; requires re-calculating the shortest path.
  - **7. Edge-All:** Completely re-samples the edges (existence and weights) while keeping the start/goal nodes fixed.
  - **8. Start-Goal (Sink-Source):** Keeps the graph structure fixed but randomly selects new start and goal nodes.
  - **9. All:** Completely re-generates the graph, weights, start, and goal for every sample (simulating an infinite dataset).

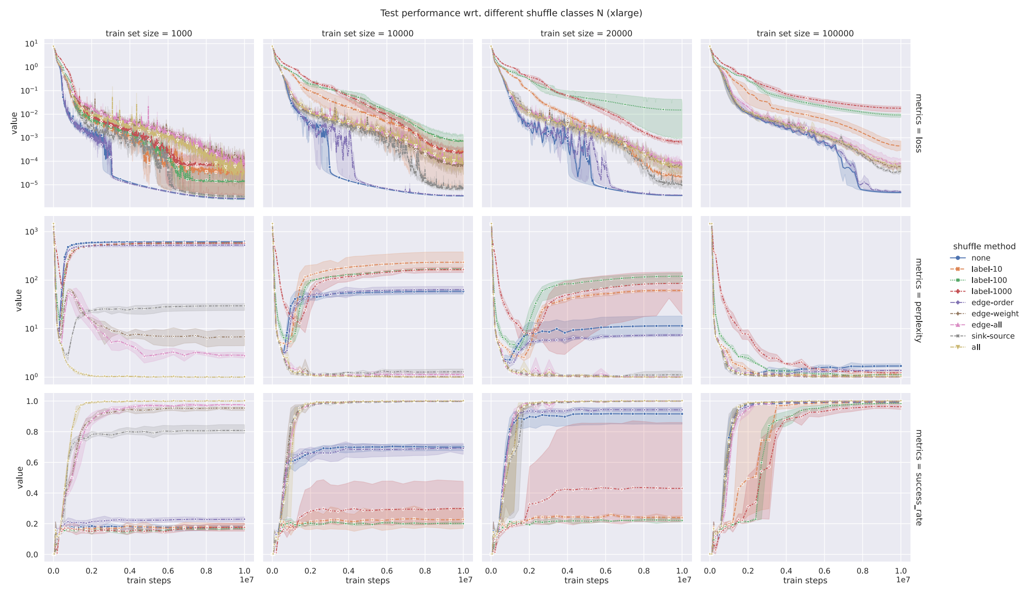


Figure 3.5: Comprehensive performance analysis of nine data augmentation strategies. The figure is organized in a matrix form, where columns correspond to different training dataset sizes, and rows represent three evaluation metrics: **training loss**, **test perplexity**, and **test success rate**. Each subplot contains 9 curves, representing different augmentation strategies. The results are reported as the mean and 95% confidence interval (shaded) over 4 random seeds. Please note the clear distinction between strategies that modify CoT (events) and those that only modify the Prompt. The large image is provided in Appendix .1.

## Results and Analysis

The results, summarized in Figure 3.5, reveal three critical insights regarding how Transformers learn algorithms:

**1. Prompt Permutation is Ineffective.** The performance of *Edge-Order* is statistically indistinguishable from the *None* baseline across all metrics and dataset sizes. This suggests that the model is naturally robust to the permutation of input facts (edges) and that simply shuffling the input context does not provide a meaningful training signal for structural generalization.

**2. Symbolic Diversity Can Hinder Learning.** Counter-intuitively, the *Label-N* strategies (2-4) often underperform the baseline, particularly when the dataset size is small.

- **Observation:** Increasing the label space (e.g., to 1000) degrades training efficiency. The model struggles to converge compared to the fixed 0-7 label set.
- **Hypothesis:** This highlights the cost of **Symbol Binding**. When labels are fixed to a small set ($0 \sim 7$), the model can overfit to these specific token embeddings. When labels are expanded to 1000, the model is forced to learn a label-agnostic algorithm. While this is theoretically desirable for infinite generalization, it significantly increases the difficulty of the learning task. The model must decouple the "algorithm" from the symbols, and without sufficient data volume, this added complexity outweighs the benefits of diversity.

**3. CoT Diversity is Key.** All strategies that involve re-generating the reasoning path (Strategies 6-9) yield significant performance gains.

- *Edge-Weight*, *Edge-All*, and *Start-Goal* all outperform the baseline and symbolic augmentations.
- This implies that the most effective way to improve algorithmic reasoning is not to vary the *input format* (Prompt), but to vary the *reasoning trajectory* (Event). The model benefits most from seeing the *same* algorithm applied to *different* states and transitions, rather than the same state described differently.

## Probing the Solution Landscape: Is Generalization Nearby?

The results above suggest that models trained on static, limited datasets ("None" strategy) fail to generalize to unseen structures. A fundamental question arises: **Is this failure due to a lack of information, or a failure of optimization?** If the "overfitted" solution found on the limited dataset is topologically distinct from the generalizable solution (i.e., they reside in distant basins of attraction in the parameter space), then the limited data is fundamentally insufficient. However, if a generalizable solution exists in the close neighborhood of the overfitted solution, it implies that the model *could* have generalized, but the optimizer settled on a fragile heuristic.

**Experimental Design: Regularized Fine-Tuning.** To test this, we devised a "tethered" training experiment.

1. **Anchor Models ($\theta_{anchor}$):** We first obtain a model pre-trained on the limited dataset (from the "None" experiment) which achieves low training loss but poor generalization.
2. **Target Task:** We then continue training this model on the *infinite* dataset (Strategy 9, All).
3. **Constraint:** Crucially, we add an L2 regularization term penalizing the deviation from the initial anchor weights. The loss function becomes:

$$\mathcal{L}_{total} = \mathcal{L}_{infinite}(\theta) + \lambda ||\theta - \theta_{anchor}||_2^2 \tag{3.1}$$

   where $\lambda$ controls how strictly the model is forced to stay near the limited-data solution.

We compare this against two controls: (1) Tethering to a random initialization (to test if the limited-data initialization is better than random), and (2) Tethering to a model already trained on infinite data (as an upper bound).

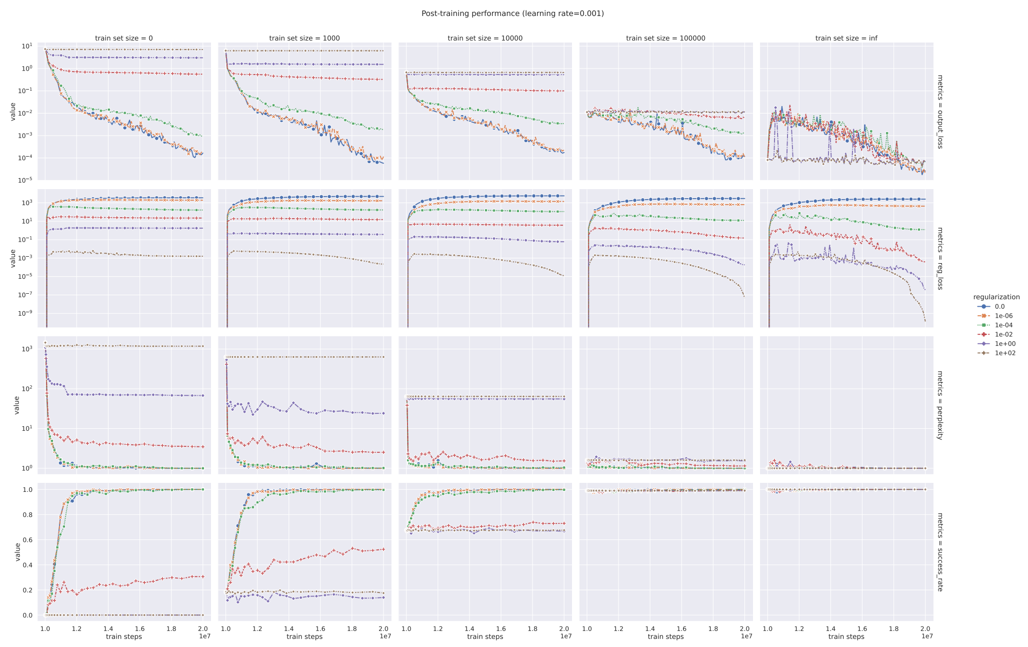


Figure 3.6: Preliminary results of the Regularized Fine-Tuning experiment. The plots show the trade-off between adhering to the anchor parameters (Regularization Loss) and solving the general task (Test Success Rate) under varying $\lambda$. A full breakdown of metrics including Training Loss and Test Perplexity is provided in Appendix .2.

**Preliminary Interpretation.** As shown in Figure 3.6, we observe the model's behavior as we relax the constraint (decreasing $\lambda$).

- If high generalization accuracy can be achieved while maintaining a low distance from $\theta_{anchor}$ (low regularization loss), it suggests that the generalizable algorithm is accessible from the limited-data solution.
- Conversely, if the model must move significantly far from $\theta_{anchor}$ to solve the task, it implies that the limited-data model learned a fundamentally different, non-generalizable mechanism.

While this experiment is ongoing, early results suggest a complex relationship between dataset size and the distance to generalization, hinting that for very small datasets, the learned features may be orthogonal to the true algorithm. We leave a comprehensive analysis of these landscape dynamics for future work.

# Conclusion and Future Work

## 4.1 Conclusion

This study has systematically investigated the mechanistic constraints of Transformer-based models in learning algorithmic reasoning, using pathfinding as a canonical proxy. By deconstructing the interplay between sequence length distributions, positional encodings, and data augmentation strategies, we have arrived at several critical insights regarding the nature of "In-Context Algorithm Learning."

First, we identified the "Window Effect" (RQ1), demonstrating that models do not merely learn algorithms but learn *position-dependent* heuristics. The bell-shaped performance curve reveals that generalization fails not only on longer sequences (extrapolation) but also on shorter ones (interpolation), suggesting that the model's internal "world model" is rigidly calibrated to the specific positional indices seen during training.

Second, our investigation into Positional Encodings (RQ2) and Failure Modes (RQ3) confirmed that this rigidity is exacerbated by absolute positional biases. The Perplexity Matrix analysis exposed a clear "termination bias," where the model hallucinates algorithmic steps simply because the current token index does not match the expected solution length. This highlights a fundamental conflict between the translation-invariant nature of algorithms and the position-bound nature of standard Transformer architectures.

Third, through extensive Data Augmentation experiments (RQ4), we established that diversity in the *Chain-of-Thought* (Event Trace) is far more critical than diversity in the *Prompt*. Strategies that forced the model to generate novel reasoning trajectories (e.g., re-weighting edges) significantly outperformed surface-level permutations. This validates the theoretical view (RQ0) that the CoT serves as a necessary computational budget, and that the model learns best when forced to decouple the algorithmic logic from static symbolic representations.

Finally, our preliminary exploration of the Loss Landscape suggests that generalizable solutions may exist in the close neighborhood of overfitted ones, but standard optimization trajectories on limited data fail to locate them. This points to a "optimization gap" as much as a "data gap."

## 4.2 Future Work

While this work sheds light on the failure modes of algorithmic learning, several avenues remain open for deeper investigation.

### Decoupling Length Generalization Factors

Our findings confirm that performance degrades on both sides of the training distribution (shorter and longer), but it remains unclear if these failures stem from the same mechanistic root. Current literature largely relies on Perplexity (PPL) as a proxy for failure in Language Models, which is often too coarse to capture algorithmic logic errors. Future work should aim to establish a rigorous, quantitative link between specific Positional Embedding (PE) schemes and *algorithmic* generalization error. We aim to develop fine-grained metrics that can distinguish between "syntax errors" (e.g., invalid token formats) and “logic errors” (e.g., valid moves that violate the shortest-path constraint) across different length regimes. This would allow us to isolate whether the “Window Effect” is purely a failure of attention calibration or a deeper failure of state tracking.

### Revisiting Relative Positional Encodings

In this study, we proposed a Pair-Wise Rotary Embedding (PWRE) scheme to bound the input space, but practical implementation was hindered by computational overhead and optimization instability. However, the theoretical argument for a fully relative, bounded encoding remains strong. Future research should revisit this direction, not merely to engineer a faster implementation, but to understand *why* such relative schemes struggle to converge. Is the difficulty inherent to the loss landscape when relative distances are dynamically rescaled? Investigating this could yield a “middle ground” architecture that retains the efficiency of RoPE while achieving the translation invariance of relative attention, potentially solving the horizon barrier for algorithmic tasks.

### Testing the Solomonoff Hypothesis: From Complex to Simple

Our theoretical framework (RQ0) posits that Transformers act as Solomonoff inductors, preferring the simplest program that fits the data. We observed that on simple data, they learn simple (but wrong) positional heuristics. To rigorously test this hypothesis, we propose an inverse experiment: training on *complex* scenarios (e.g., large, dense graphs with high path variability) and testing on *simple* ones (e.g., small, linear paths). If the Solomonoff hypothesis holds, a model forced to learn the complex, general algorithm to satisfy the difficult training set should trivially solve the simple test set without overfitting to position. Conversely, if it fails, it would suggest that the model is not learning a unified algorithm at all, but rather a patchwork of memorized sub-routines. This would fundamentally challenge our understanding of how Transformers internalize algorithmic logic.

# Bibliography

[1] Dong, Q., Li, L., Dai, D., Zheng, C., Wu, Z., Chang, B., Sun, X., Xu, J., & Sui, Z. (2022). A Survey on In-context Learning. *arXiv preprint arXiv:2301.00234.*

[2] Min, S., Lyu, X., Holtzman, A., Artetxe, M., Lewis, M., Hajishirzi, H., & Zettlemoyer, L. (2022). Rethinking the Role of Demonstrations: What Makes In-Context Learning Work? *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing (EMNLP).*

[3] Chan, S. C., Santoro, A., Lampinen, A. K., Wang, J. X., Singh, A., Richemond, P. H., ... & Hill, F. (2022). Data Distributional Properties Drive Emergent In-Context Learning in Transformers. *Advances in Neural Information Processing Systems (NeurIPS).*

[4] Weiss, G., Goldberg, Y., & Yahav, E. (2021). Thinking Like Transformers. *Proceedings of the 38th International Conference on Machine Learning (ICML).*

[5] Laskin, M., Wang, L., Oh, J., Parisotto, E., Mnih, V., Geffner, T., & Salakhutdinov, R. (2022). In-context Reinforcement Learning with Algorithm Distillation. *International Conference on Learning Representations (ICLR).*

[6] Lee, J. N., Xie, A., Pacchiano, A., Chandak, Y., Finn, C., Brunskill, E., & Mazumdar, A. (2023). Supervised Pretraining Can Learn In-Context Reinforcement Learning. *Advances in Neural Information Processing Systems (NeurIPS).*

[7] Raparthy, S. C., Hambro, E., Risi, S., & Raileanu, R. (2023). Generalization to New Sequential Decision Making Tasks with In-Context Learning. *arXiv preprint arXiv:2312.03801.*

[8] Lehnert, L., & Tiomkin, S. (2024). Beyond A*: Better Planning with Transformers via Search Dynamics Bootstrapping. *International Conference on Machine Learning (ICML).*

[9] Chu, Z., Liu, J., Yuan, Z., Wang, X., Yang, Q., Wen, S., ... & Qiao, Y. (2024). SFT Memorizes, RL Generalizes: A Comparative Study of Foundation Model Post-training. *arXiv preprint arXiv:2401.12954.*

[10] Lee, N., Sreenivasan, K., Penentler, J., & Papailiopoulos, D. (2023). Teaching Arithmetic to Small Transformers. *International Conference on Learning Representations (ICLR).*

[11] Anil, C., Wu, Y., Andreassen, A., Lewkowycz, A., Misra, V., Ramasesh, V., ... & Dyer, E. (2022). Exploring Length Generalization in Large Language Models. *Advances in Neural Information Processing Systems (NeurIPS).*

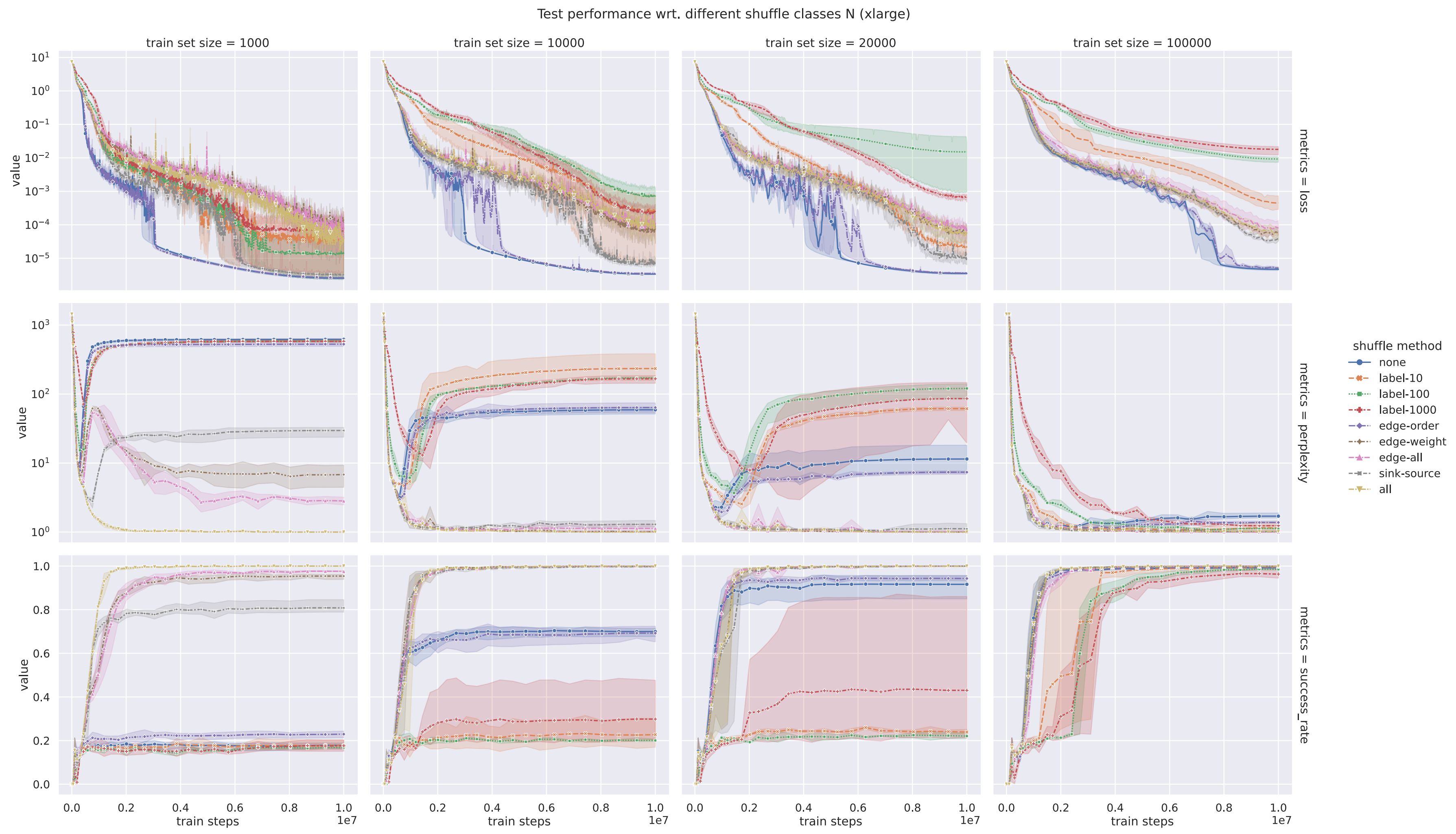


Figure .1: comprehensive performance analysis of nine data augmentation strategies. The figure is organized in a matrix form, where columns correspond to different training dataset sizes, and rows represent three evaluation metrics: **training loss**, **test perplexity**, and **test success rate**. Each subplot contains 9 curves, representing different augmentation strategies. The results are reported as the mean and 95% confidence interval (shaded) over 4 random seeds. Please note the clear distinction between strategies that modify CoT (events) and those that only modify the Prompt. The large image is provided in .1.

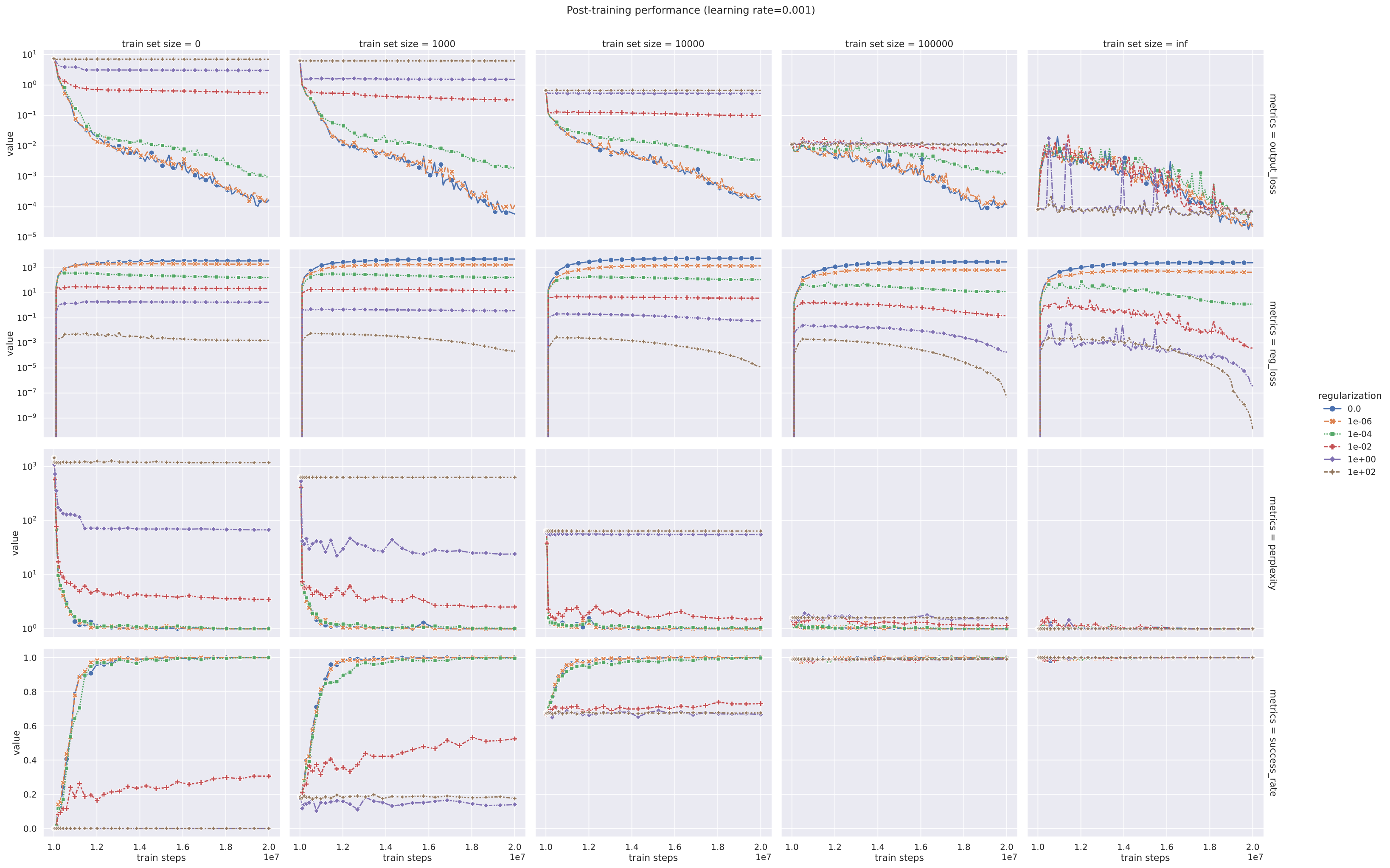


Figure .2: Preliminary results of the Regularized Fine-Tuning experiment. The plots show the trade-off between adhering to the anchor parameters (Regularization Loss) and solving the general task (Test Success Rate) under varying $\lambda$. A full breakdown of metrics including Training Loss and Test Perplexity is provided in Appendix .2.