Frontier Language Models Struggle to Copy: Text Can Be Better Viewed in 2D

Haodong Wen[1*] **Yiran Zhang**[1*] **Yingfa Chen**[2*] **Kaifeng Lyu**[1‡]
[1]Institute for Interdisciplinary Information Sciences, Tsinghua University
[2]Department of Computer Science and Technology, Tsinghua University
{whd25,zhangyir22}@mails.tsinghua.edu.cn
chenyingfa1999@gmail.com
klyu@mail.tsinghua.edu.cn

Abstract

While large language models (LLMs) can solve advanced reasoning problems in seconds, we show that even frontier models fail to perform a much simpler operation: exactly copying an input string that lies well within their context windows. We attribute this failure to positional encodings in Transformer architectures, whose inductive bias favors copying through a shortcut based on matching local contexts rather than carefully locating the corresponding input positions. To address this issue, we introduce 2D-RoPE, which organizes text into a 2D grid rather than a 1D sequence and assigns each token a row ID and a column ID. Under this view, copying becomes simply retrieving input tokens from the same column, which makes the task easy to learn. In synthetic copy experiments, shallow Transformers with 2D-RoPE achieve perfect copying at input lengths hundreds of times longer than those seen during training, whereas standard positional encodings fall far behind. We further pretrain 2D-RoPE language models on DCLM at scales up to 1.4B parameters and show that 2D-RoPE substantially improves performance on copy and NIAH tasks. Overall, our results suggest that viewing text in 2D can benefit language modeling, and we hope this encourages future work to further explore the potential of 2D positional encodings.

1. Introduction

While large language models (LLMs) have achieved tremendous success on a wide range of complex reasoning problems, we show that they still struggle with a seemingly simple task: given a string as input, reproduce the exact same string as output, possibly with minor adaptations to match a required output format. We refer to this as the *copy* task and specifically consider the following two variants:

- **Binary Copy.** We give the model a simple string consisting of two types of tokens (such as 0 and 1) and the model is instructed to output the exact same sequence of tokens.
- **Python List Conversion.** We generate a comma-separated list of data points from synthetic physical experiments, feed it to the LLM, and ask the LLM to convert the data into a Python list that can be used to generate plotting code for the data.

Although solving these copy tasks may not require the same level of intelligence as an LLM, it is arguably a fundamental capability that any intelligent agent should possess. For example, an agent may need to copy parameters from a configuration file for function calling, or organize unstructured user input into a specific format for downstream processing. After all, if current LLMs are already intelligent enough to achieve gold-medal performance in Olympiad-level mathematics and competitive programming (OpenAI, 2025; Luong & Lockhart, 2025; Lin & Cheng, 2025), why shouldn't we expect them to perform this very basic copying task perfectly?

[*]Equal contribution
[‡]Corresponding author.

Accepted to FoGen 2026: Foundations of Deep Generative Models: Understanding Memorization, Generalization, and Reasoning, an ICML 2026 workshop (non-archival).

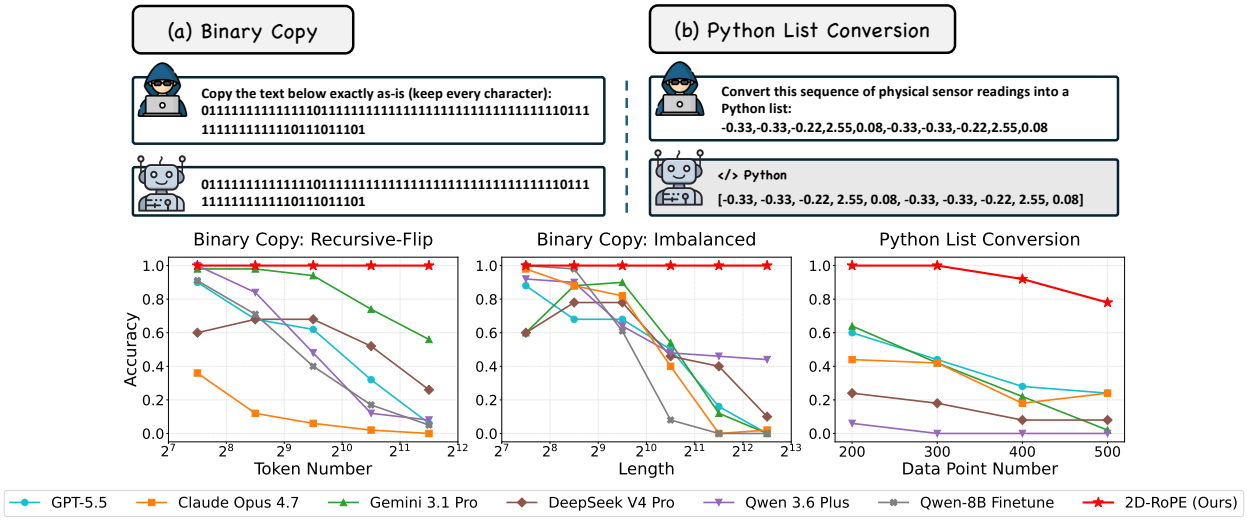


Figure 1. **Top-row:** Examples of Binary Copy and Python list conversion prompts. **Bottom-row:** Frontier language models struggle to copy strings perfectly, while our 2D-RoPE models achieve significantly better performance. See Section 2.1 for details on the benchmark construction.

However, as shown in Figure 1, all the frontier LLMs we evaluate, including GPT-5.5, Gemini 3.1 Pro, and DeepSeek V4 Pro, fail on a substantial fraction of input strings that are well within their context lengths. Their copying accuracy drops further on longer inputs, often falling well below 50%.

Understanding the Failure of Copying. Our experimental design is motivated by the following conjecture: LLMs may not copy a string by directly using the absolute index i to retrieve the i-th character from the input. Instead, copying may be implemented in a way closer to the induction head mechanism (Olsson et al., 2022; Chen et al., 2024b): the model searches for a previous occurrence of a similar local context and then predicts the token that appears right after this occurrence.

Under this view, repeated substrings create ambiguous local matches and can therefore interfere with exact copying. Indeed, in our experiments, the tested strings are not arbitrary but are deliberately chosen to contain repeated substrings. For example, in the Python List Conversion task, part of the data is generated from periodic sine or triangular waves. Our further experiments in Figure 2 show that copying accuracy is negatively correlated with the degree of repetition in the input strings.

In Section 2, we develop expressivity theory and mechanistic interpretability results to support this conjecture, and show that the copying failure is closely tied to the positional encoding used in Transformers. Although RoPE has appealing properties such as shift-invariance, it does not provide a sufficiently strong inductive bias for the model to align the source and target tokens by their relative positions. As a result, the learned attention patterns are not cleanly aligned with the corresponding source positions, but are instead easily attracted by locally similar substrings.

Our Method: 2D Positional Encoding. This analysis points to a natural design goal: a positional encoding with a stronger inductive bias towards copying from the corresponding position in the input. To this end, a key observation is that the copy task has an implicit 2D structure. When copying an input string $x_1, \ldots, x_n$ into an output string $y_1, \ldots, y_n$, if we only view the whole context as a 1D sequence, for example $x_1, \ldots, x_n, <\backslash\mathrm{n}>, y_1, \ldots, y_n$, then producing an output token y_k requires the model to locate the corresponding input token x_k in this 1D sequence through careful arithmetic over positions. To do so, the model may first have to identify the boundary between the input and output, since the relative positions of x_k and y_k depend on the input length n. It is thus no wonder that models may fall back to an easier shortcut such as matching similar substrings. Instead, if we arrange the input and output strings into two rows and use the position k within each string as the column index, then producing y_k reduces to simply retrieving a token from the same column.

Motivated by this key observation, we introduce 2D-RoPE. The construction is inspired by prior uses of 2D positional encodings in vision models that view images as a 2D grid (Jeevan & Sethi, 2022; Heo et al., 2024; Chu et al., 2024). Given any text, 2D-RoPE uses the line break token $<\backslash\mathrm{n}>$ (also called the newline token) as the row separator and views

the text as a 2D grid, where each token is assigned a 2D position ID pair $(\text{row ID}, \text{column ID})$. RoPE attention is then applied to this 2D grid to encode relative positions along both axes. In this view, copying a string can be implemented by a single attention layer that retrieves tokens from the same column, which is arguably a simpler structure for the model to learn.

Empirically, 2D-RoPE leads to strong length generalization on the copy task. Here, length generalization refers to the ability to generalize from short training sequences to longer test sequences. In synthetic copy experiments, we train 2D-RoPE models on the binary copy task and show that one-layer models achieve perfect copying with **up to** $1000\times$ **length generalization**, while 12-layer models maintain the perfect performance up to $100\times$ length generalization. Theoretically, we first prove that a one-layer Transformer equipped with 2D-RoPE can provably express the binary copy problem and achieve length generalization.

Beyond this synthetic setting, many natural documents also contain explicit 2D structures induced by line breaks, such as lists, tables, and code, which today's LLMs do not directly encode in their positional encodings. We therefore pretrain 2D-RoPE models ranging from 0.3B to 1.4B parameters on DCLM (Li et al., 2024). After finetuning, these models exhibit length generalization on copy tasks and significantly improve the long-context retrieval performance on the Needle In A Haystack (NIAH) task (Hsieh et al., 2024) across lengths from 512 to 4K.

Finally, since the 2D structure in natural documents may not always be explicitly marked by line breaks, we introduce Auto-2D-RoPE, which learns a data-dependent transformation to automatically assign 2D coordinates to each token. Our experiments show that Auto-2D-RoPE maintains length generalization on binary copy even without line breaks, whereas the original 2D-RoPE does not.

- We show that frontier LLMs fail on the copy task, and develop expressivity theory and mechanistic interpretability results to explain this failure (Section 2).
- We introduce 2D-RoPE, a positional encoding that views text as 2D rather than 1D. Training 2D-RoPE models specifically on the copy task leads to strong length generalization. We also provide a theoretical guarantee for the length generalization of 2D-RoPE on the copy task. (Section 3).
- Our experiments on LLM pretraining show that 2D-RoPE can exploit the 2D structure of text and significantly improve performance on the copy and NIAH tasks (Section 4).
- To reduce the explicit reliance of 2D-RoPE on line breaks, we introduce Auto-2D-RoPE, which automatically determines the 2D coordinates for each token and length-generalizes on the copy task even without line breaks (Appendix G).

Overall, our results suggest that viewing text in 2D can benefit language modeling, and we hope this encourages future work to further explore the potential of 2D positional encodings.

Comparison with Jelassi et al. (2024). Jelassi et al. (2024) is most closely related to our work, which studies synthetic copy tasks and gives a two-layer Transformer construction based on context matching. Our focus is different in several aspects. First, we show that frontier LLMs fail to copy user-provided strings exactly, even within their context lengths. Second, their construction and analysis focus on the uniform data setting and do not imply perfect copying for arbitrary strings with repeated substrings or provide the length-generalization guarantee considered in our work. Finally, their construction is context-based, whereas our results suggest that repeated substrings make context matching unreliable, and further, we propose 2D-RoPE to solve copy. See Appendix A for additional related works.

2. Language Models Fail to Copy

In this section, we first introduce our copy benchmarks in detail, and then present our analysis to diagnose the failure of LLMs on the copy tasks.

2.1. Our Copy Benchmarks

For a testbed of LLMs' copy capabilities, we construct the following copy tasks. See example prompts in Figure 1 and full prompt templates used in our experiments in Appendix L.

Binary Copy Task. In binary copy, we fix a vocabulary $\mathcal{V}$ of size $|\mathcal{V}| = 2$, which means that the input string s only consists of two types of tokens. Given an input string s, the model is required to output an exact copy of s. For example, the input string can be $s =$ "$\texttt{011010101010100}$", with the vocabulary fixed to $\mathcal{V} = \{\texttt{0}, \texttt{1}\}$. We generate such input

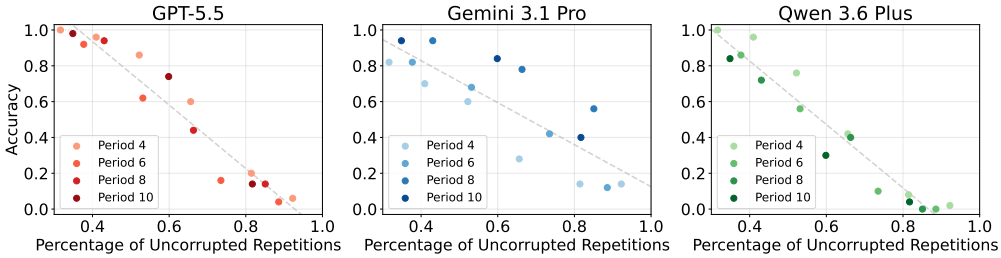


Figure 2. Repeat-structure copy test for frontier LLMs. Accuracy decreases as repetitions are preserved more strongly, suggesting that repetitions make exact copying harder; see Appendix B.2 for details.

strings from three different distributions:

- **Uniform Generation:** Each token is sampled independently and uniformly from $\mathcal{V}$.
- **Imbalanced Generation:** We fix a probability set $p \in \{0.05, 0.15, 0.3, 0.5, 0.7, 0.85, 0.95\}$. For each string, we first sample p uniformly from this set, and then each input token is sampled as the first token in $\mathcal{V}$ with probability p and the second token with probability $1 - p$.
- **Recursive-Flip Generation:** We initialize s with a single token sampled uniformly from $\mathcal{V}$. Then we iteratively update $s \leftarrow s + c + s$ for several rounds, where c is a randomly sampled token from $\mathcal{V}$ in each round. After K rounds, we obtain a string s with length $2^{K+1} - 1$, which contains a complicated recursive structure as well as many random flips. When we need to generate a string with an arbitrary length n, we simply truncate s to its first n tokens.

Python List Conversion Task. We generate a comma-separated list of data points from synthetic physical experiments and ask the LLM to convert the data into a Python list. This mimics the scenario where an agent needs to copy user inputs into a specific format for downstream processing with Python code, such as visualizing the data. Our evaluation includes 9 synthetic experiments in total, and each data-point list is generated randomly from one of the 9 experiments. These experiments can be further categorized into three types: smooth periodic oscillations, piecewise or reset-driven oscillations, and nonlinear or pulse-like signal responses. See Appendix B for details.

2.2. Evaluation of Frontier LLMs

Figure 1 shows that frontier LLMs fail to perform perfect copying on both binary copy and Python list conversion tasks. For binary copy, we test the models on Recursive-Flip and Imbalanced data. For each length interval $[2^k, 2^{k+1} - 1)$ with $k \in \{7, 8, 9, 10, 11\}$, we generate 50 samples with lengths sampled uniformly from that interval, and report the average copying accuracy within each interval. For Recursive-Flip, the vocabulary is either {"`0`", "`1`"} or {"`000`", "`111`"}, depending on whether the model's tokenizer tends to merge consecutive 01 characters into a single token. For Imbalanced data, we fix the same vocabulary for all models: {"` a`", "` b`"}. For the Python List Conversion task, we evaluate four list lengths, $200, 300, 400, 500$, with 50 samples for each length. See details of the evaluation in Appendix B.

There are two notable features in the results. First, as the input length increases, the accuracy of all models decreases. This suggests that the models may only have learned an ad-hoc copy rule that works for some specific input length range, rather than a rule that can be generalized to all lengths. Throughout the paper, we use the term *length generalization* to refer to the ability of a model to perform well on inputs with lengths beyond the training range.

Second, frontier LLMs perform especially poorly on strings with many repeated substrings. To see this, we sample a short binary base string, repeat it multiple times to generate a fixed-length long string, and independently flip each token with probability $1 - p$. As we increase p from 0 to 1, each repeated substring becomes less likely to be corrupted, and thus the input contains more repeated substrings. As shown in Figure 2, the copying accuracy consistently decreases as the number of repeated substrings increases in this way. Details of this experiments can be found in Appendix B.2.

2.3. Why Do LLMs Fail to Copy?

Motivated by the above observations, we now state our conjecture on why frontier LLMs fail to copy. There are two natural algorithms for solving the copy task:

- *Position-based algorithm*: to output the k-th output token y_k, derive some representation of the index k and use it to retrieve the corresponding input token x_k.

- *Context-based algorithm*: first identify the local context around the current output token y_k, such as $y_{k-3}y_{k-2}y_{k-1}$, and then find a similar local context in the input, and then copy the token associated with that matched context.

The position-based algorithm sounds the most natural, but implementing it in a Transformer requires careful arithmetic over positions, since the relative position between x_k and y_k depends on the input length n, which may vary across input strings. Without handling this dependence properly, the model may learn to copy only for some input lengths but not others.

The context-based algorithm is more flexible and can be implemented with a few attention layers, but it is NOT a correct algorithm in general: if there are many repeated substrings in the input, the context-based algorithm may not be able to locate the correct input position.

It is clear from the imperfect copying accuracy that LLMs do not exactly implement either of the two algorithms above, but what do they do instead? We conjecture that LLMs instead implement a mixture of the two algorithms. In other words, LLMs exhibit an inductive bias towards a mixture of position-based and context-based copying algorithms. This conjecture is supported by the two failure modes: copying accuracy decreases as either the input length or the degree of repetition increases.

2.4. Diagnosis via Expressivity Theory and Synthetic Experiments

To further support our conjecture and understand why the inductive bias does not favor the position-based algorithm exactly, we analyze the problem via expressivity theory and synthetic experiments. To help us better understand copying task, we consider the general binary copy task with a fixed template. For a fixed positive integer pair (a, b), we consider such an input $(\mathtt{A}_1, \cdots, \mathtt{A}_a, s, \mathtt{B}_1, \cdots, \mathtt{B}_b)$, where s is a 0/1 string and Transformer is asked to autoregressively output s again. Here $\mathtt{A}_i$ and $\mathtt{A}_j$ are all different tokens other than 0/1.

2.4.1. Evidence from Expressivity Theory

First of all, we ask: can Transformers express the position-based copying algorithm? The answer is mixed. Given the apparent simplicity of the copy task, one might expect a single Transformer layer to be expressive enough to solve it. This expectation is natural: single attention layers are closely related to associative recall (Ramsauer et al., 2021; Bricken & Pehlevan, 2021; Smart et al., 2025), and positional encodings can in principle support retrieval tokens at a fixed relative offset (Vaswani et al., 2017; Weiss et al., 2021). Perhaps surprisingly, this is not the case for binary copy. We show that even for input strings with length at most 5, one-layer Transformers cannot exactly perform copying.

Theorem 2.1. *No one-layer Transformer with shift-invariant positional encoding can perform the binary copy task for all input strings with length $n \leq 5$.*

Note that shift-invariant positional encodings cover a wide range of positional encodings, including RoPE, NoPE, ALiBi, etc. This follows from a simple proof by contradiction. Intuitively, one-layer transformer fails to copy since copy requires retrieval at a length-dependent offset rather than a fixed offset. Thus, the failure of one-layer models reflects a lack of capability for processing length-dependent tasks. The formal statement of this theorem and its proof are in Appendix I.

Beyond one layer, we show that two-layer Transformers with RoPE can express binary copy.

Theorem 2.2. *Let $a, b \geq 2$. There exists a constant embedding dimension $d \geq 1$ such that for any norm constraint $\rho > 10^4$, there exist RoPE frequencies $\boldsymbol{\beta}$ and a parameter vector $\boldsymbol{\theta}$ with $\|\boldsymbol{\theta}\|_2 \leq \rho$ such that the two-layer Transformer with embedding dimension d, RoPE frequencies $\boldsymbol{\beta}$, parameters $\boldsymbol{\theta}$ can perfectly perform copying for all lengths $1 \leq L \leq O(\rho^2/\log\rho)$.*

The key intuition here is that with multiple layers, the model can construct intermediate positional features and use them to align each output position with its corresponding input position. The detailed setup and proof of this theorem is in Appendix J. Notice that in our synthetic experiemnt for binary copy in Section 2.4.2, we set $a = 2, b = 3$, which is within the scope of our theoretical arguments above.

Putting these two theorems together, we can conclude that: (1) the inductive bias induced by standard positional encodings, including RoPE, are not strong enough to support perfect copying for one-layer Transformers; (2) Transformers with multiple layers can express the position-based copying algorithm, suggesting that training, rather than expressivity alone, biases Transformers away from position-based algorithms.

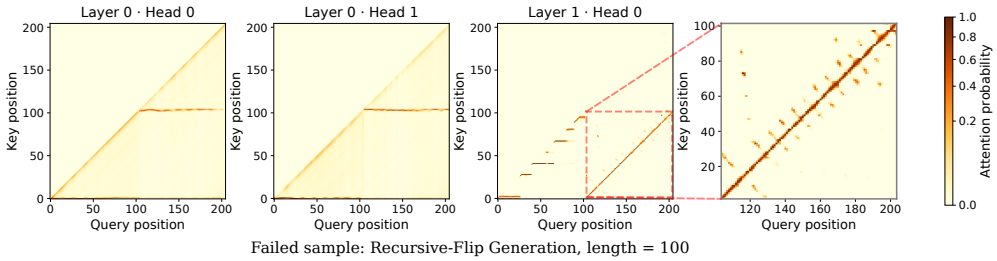


Figure 3. Attention map of a RoPE model on a failed sample with sequence length within training length. In Layer 0, attention heads exhibit both **horizontal patterns** (attending to special tokens before and after the input string) and **diagonal patterns** (semantic/induction-style behavior). In Layer 1, RoPE would attend to wrong tokens near themselves periodically.

2.4.2. Synthetic Experiments

Setup. We further analyze what Transformers learn through synthetic experiments by training on the binary copy task. Here we consider two settings: Transformers with 1 layer and 2 attention heads, and Transformers with 12 layers and 12 attention heads. For each setting, we consider 4 position encodings: RoPE, ALiBi (Press et al., 2021), NoPE, and RNoPE (Yang et al., 2025b). The template of the data are for $a = 2, b = 3$ as defined at the beginning of Section 2.4.

Uniform Data. We first train these models on uniform data with lengths from 1 to 100. While they perform perfectly on uniform data, they perform poorly on other data distributions, such as Imbalanced (see Figure 6). This suggests that the models may have learned a context-based algorithm.

Imbalanced Data. Instead, when we train these models on imbalanced data for sufficient iterations, they can perform almost perfectly on imbalanced data as well as the other data distributions we construct. However, all the position encodings mentioned above fail to generalize to longer inputs. See Appendix C for more details of experimental setups.

Attention Patterns. We further show that these models use a mixture of position-based and context-based algorithms when copying. To see this, we analyze the attention patterns of a trained RoPE model on binary copy under the imbalanced distribution. We consider a small model with 2 heads in the first layer and 1 head in the second layer, trained on sequences with lengths from 1 to 100. Figure 3 shows the attention maps of all layers and heads on representative copy examples.

We first inspect the last layer, since exact copying requires each output position to align with its corresponding input position. However, even for a failed sample with length 100, which is within the training range, the learned attention is not a clean position-based alignment. As shown in the zoomed attention map, the final-layer head attends not only to the correct diagonal positions, but also periodically assigns large attention scores to nearby wrong tokens. This suggests that the final copy decision is affected by ambiguous local patterns.

We then inspect Layer 0 to understand where this imperfect alignment comes from. We observe two types of patterns: **horizontal stripes** on special boundary tokens such as [`<\n>`, `<I:>`, `<O:>`], which behave like locator heads and provide position-related signals; and **diagonal patterns**, which resemble local-context or induction-style behavior (Olsson et al., 2022; Singh et al., 2024; Jelassi et al., 2024). Together, these attention maps suggest that the trained RoPE model does not implement a purely position-based copy rule. Instead, it combines locator-style positional signals with context-based matching, and this mixture can be confused by repeated local structures. We provide further discussion beyond the training length in Appendix D.

3. 2D Positional Information Encoding with 2D-RoPE

Given that current LLMs exhibit inductive bias towards a mixture of position-based and context-based algorithms when copying and can fail, a natural question is whether we can design a positional encoding that explicitly favors the position-based algorithm. In this section, we propose a novel architecture based on 2D-RoPE (Section 3.1). We first show the superior performance of 2D-RoPE in copying (Section 3.3). Then, we present a theoretical analysis of 2D-RoPE (Section 3.4) in expressivity.

3.1. 2D-RoPE Architecture

2D Position ID Generation. The key idea of 2D-RoPE is to assign each token a pair of position IDs instead of a single position ID. We use the line break token `<\n>` as a row separator. Let the 2D position of token i be (r_i, c_i), where r_i is the row index and c_i is the column index within that row. We set the first token position to $(r_1, c_1) = (0, 0)$.

For each subsequent token $i > 1$,

$$(r_i, c_i) = \begin{cases} (r_{i-1},\, c_{i-1} + 1), & \text{if token } i-1 \text{ is not a line break token,} \\ (r_{i-1} + 1,\, 0), & \text{if token } i-1 \text{ is a line break token.} \end{cases}$$

Thus, tokens on the same line share the same row index, and their column index records the offset within the line. In a copy task, this representation makes the source token x_k and the target token y_k share the same column index, so copying can be reduced to retrival from the same column. This assumes that the input and output are separated by a line break. For the case where the separator is not a line break, we propose Auto-2D-RoPE to automatically identify the separator (see Appendix G).

From 1D RoPE to 2D-RoPE. Standard RoPE (Su et al., 2024) applies a positional rotation to the query and key vectors according to the 1D token position. In 2D-RoPE, we apply the same idea separately to the row and column coordinates. Concretely, we split the head dimension into two equal-sized parts: one encodes relative row positions, and the other encodes relative column positions. This gives the model direct access to both row differences and column differences. For copy, the column coordinate is especially useful, because source and target tokens that should be aligned have the same column ID, although their 1D distance depends on the input length.

We now give the formal definition. Let d be the head dimension, divisible by 4. For an input $\boldsymbol{X} \in \mathbb{R}^{T \times d}$, a 2D-RoPE attention layer is defined as

$$\mathsf{Attn}(\boldsymbol{X}) = \mathcal{S}\left(\mathrm{MASK}\left(R(\boldsymbol{X}\boldsymbol{Q})\big(R(\boldsymbol{X}\boldsymbol{K})\big)^\top\right)\right)\boldsymbol{X}\boldsymbol{V},$$

where $\mathcal{S}$ is the row-wise softmax, MASK is the causal attention mask, and $\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}$ are trainable projection matrices.

It remains to define the rotation function R. For any matrix $\boldsymbol{Z} \in \mathbb{R}^{T \times d}$, the i-th row $\boldsymbol{Z}_i$ is mapped to $\boldsymbol{Z}_i R_{r_i, c_i}^\top$, where $\boldsymbol{R}_{r_i, c_i} = \mathrm{diag}\big(\boldsymbol{R}_1(r_i), \ldots, \boldsymbol{R}_{d/4}(r_i), \boldsymbol{R}_1(c_i), \ldots, \boldsymbol{R}_{d/4}(c_i)\big)$. Each block $\boldsymbol{R}_j(t) = \begin{bmatrix} \cos(t\beta_j) & -\sin(t\beta_j) \\ \sin(t\beta_j) & \cos(t\beta_j) \end{bmatrix}$ is the usual RoPE rotation matrix, where β_j is the frequency parameter. In our synthetic experiments in Section 2.4, we use $\beta_j = \theta^{-4j/d}$ where $\theta = 100$. In our LLM experiments in Section 4, we set θ to 1000. For the original RoPE architecture, see Appendix H. We also discuss differences from prior vision 2D-RoPE variants in Appendix E.

3.2. How to Establish 2D Position Encodings in Text

Previous vision models have explored applying RoPE to pixels or image patches using their height and width coordinates, so that attention can encode relative spatial displacement on the image grid (Jeevan & Sethi, 2022; Heo et al., 2024; Chu et al., 2024). In language modeling, the input is typically treated as a 1D token sequence, and there is no fixed 2D grid analogous to image patches. However, many text formats still contain implicit 2D structures. For example, lists use line breaks to separate items, tables use rows and columns to organize aligned entries, and code uses newlines and indentation to express block structure. These line breaks are therefore not merely formatting tokens, but often carry semantic information about how different parts of the text are organized. Motivated by this observation, we use line break tokens to induce a 2D structure in text. Each token receives a row ID, determined by the number of preceding line breaks, and a column ID, determined by its offset within the current line. Thus, the 2D coordinates are not externally given as in images, but are constructed from the textual structure itself.

From Line Break to Adaptive 2D Position. However, the above construction also reveals a limitation of 2D-RoPE. When the input contains few or no line breaks, the row coordinate changes rarely, and 2D-RoPE largely reduces to a 1D positional encoding along the column direction. This motivates a more flexible variant that can infer 2D positional structure directly from the input sequence, rather than relying on a fixed separator token. We introduce Auto-2D-RoPE for this purpose.

At each layer, Auto-2D-RoPE uses the hidden representation of each token to predict two update coefficients through a shared linear projection. These coefficients define a learnable affine update of the token's 2D coordinate, and the final 2D position IDs are obtained by composing these updates along the sequence. In this way, the model can learn when to behave like moving to the next column and when to behave like resetting the column and moving to a new row. The learned coordinates are then used in the same 2D-RoPE rotation as before. Empirical results in Section 2.4.2 and **??** show that Auto-2D-RoPE outperforms standard PEs in binary copy and even remains effective when we replace the line-break token `<\n>` with another separator $*$, where rule-based 2D-RoPE can no longer identify row boundaries and largely loses its length-generalization advantage. See the full architectural details in Appendix G.

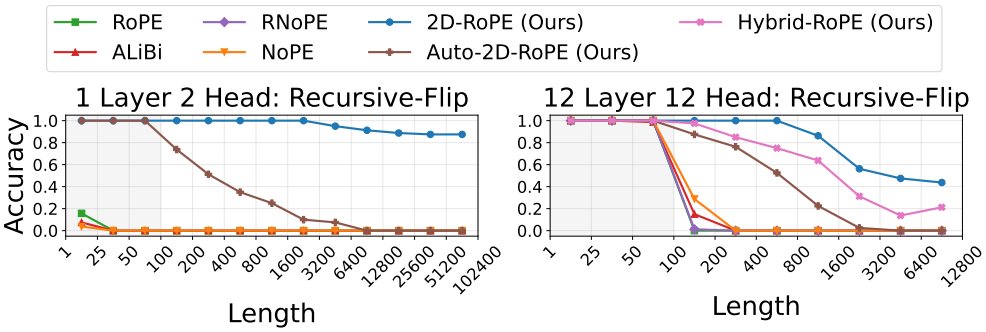


Figure 4. Length generalization on the Recursive-Flip distribution. All models are trained on Imbalanced strings with lengths from 1 to 100, so Recursive-Flip is an out-of-distribution. Full results on all test distributions are shown in Appendix C, Figure 5.

3.3. Empirical Results of 2D-RoPE

Results of Binary Copy Test. In Figure 1, the 2D-RoPE model maintains near-perfect accuracy on Recursive-Flip strings, suggesting that explicitly aligning source and target positions provides a more reliable inductive bias for copying than relying on local-context matching alone. The model in the figure has 1.4B parameters and is finetuned on the imbalanced binary copy task.

Results of Python List Conversion Test. In the right subfigure in Figure 1, our fine-tuned 2D-RoPE model in Section 4 is trained only on lists with 50 to 150 data points and periods between 2 and 8. Although the training data are easy, the 2D-RoPE model maintains substantially higher accuracy than all tested LLMs. This result is consistent with the binary copy experiments: when the input contains repeated local patterns, as in periodic or oscillatory sequences, exact copying is difficult for standard position encoding, while 2D-RoPE provides an explicit positional alignment between the source data points and their copied output positions.

Results of Synthetic Experiments on Binary Copy. Figure 4 shows that 2D positional structure provides a strong inductive bias for binary copy. 2D-RoPE-based methods exhibit much stronger length generalization than standard positional encoding. In the 12-layer setting, positional encodings with 2D structure substantially improve over standard positional encodings. The advantage is even clearer in the 1-layer setting: the 2D-RoPE model maintains perfect accuracy up to more than $1000\times$ its training context length. This suggests that the row-column alignment introduced by 2D-RoPE directly supports the position-based copy rule needed for systematic extrapolation.

3.4. Theoretical Analysis of 2D-RoPE

The experiments above suggest that 2D-RoPE provides a much stronger inductive bias for copying than standard 1D positional encodings: the source token to be copied and the corresponding target position have fixed relative positions, even though their 1D distance depends on the input length. This complements the diagnosis in Section 2. In this subsection, we show that 2D-RoPE also has positive theoretical guarantees. We prove that a one-layer 2D-RoPE Transformer can explicitly represent the position-based copy rule with bounded parameters.

3.4.1. One-Layer 2D-RoPE Can Represent Binary Copy

We study the expressive power of one-layer 2D-RoPE. In contrast to the impossibility result for one-layer shift-invariant positional encodings, 2D-RoPE makes the correct source position geometrically simple: when predicting the i-th copied token, the query position in the output row and the source token s_i in the input row have the same column coordinate. Thus, a single attention layer does not need to infer the input length or compute a length-dependent 1D offset. It only needs to attend to the token in the previous row with the same column.

The following theorem formalizes this intuition. The parameter norm controls the range of lengths for which the random 2D-RoPE frequencies separate the correct same-column position from all wrong positions. The length bound is polynomial in the norm, and the exponent grows with the RoPE dimension. We still consider format $(\mathtt{A}_1, \cdots, \mathtt{A}_a, s, \mathtt{B}_1, \cdots, \mathtt{B}_b)$ with $a, b \geq 1$ as in Section 2.4.

Theorem 3.1. *Assume that the 2D-RoPE dimension $d \geq 100 + 4a + 4b$ is divisible by 4. For any $\rho > d^2$, let the 2D-RoPE frequencies $\beta_i \overset{i.i.d.}{\sim} \mathrm{Unif}[0, 2\pi]$ for all $i \in [d/2]$. Then, with probability at least $1 - \frac{d}{\rho}$ over the randomness of β, there exists a one-layer Transformer with 2D-RoPE whose ℓ_2 parameter norm $\|\boldsymbol{\theta}\|_2 \leq \rho$ such that correctly copies every binary string s with length $|s| \leq \rho^{d/12}$.*

The construction behind Theorem 3.1 is purely position-based. We choose the query and key vectors so that the

Table 1. Large-scale pretraining results of 2D-RoPE, H-RoPE, and RoPE. Each model is pretrained on DCLM, and then finetuned on binary copy with imbalanced distribution.

Model	Param	PT Data	CSR	NIAH 512	NIAH 1K	NIAH 2K	NIAH 4K	Copy (Imbalanced) 1K	Copy (Imbalanced) 2K	Copy (Imbalanced) 4K	Copy (Imbalanced) 8K	Copy (Recursive-Flip) 1K	Copy (Recursive-Flip) 2K	Copy (Recursive-Flip) 4K	Copy (Recursive-Flip) 8K
RoPE	350M	7B	**41.4**	**93.7**	92.1	87.0	1.9	97.2	14.8	0.0	0.0	14.4	7.8	0.0	0.0
H-RoPE (ours)	350M	7B	41.2	93.1	**98.8**	**94.4**	57.9	**99.6**	57.8	0.0	0.0	**96.9**	**92.2**	35.1	0.0
2D-RoPE (ours)	350M	7B	40.4	91.9	97.9	93.9	**92.8**	97.4	**69.2**	**33.4**	**2.4**	**96.9**	89.3	**56.4**	**36.4**
RoPE	730M	15B	44.0	**100.0**	**98.4**	**96.7**	26.4	97.0	11.0	0.0	0.0	15.5	7.8	1.1	0.0
H-RoPE (ours)	730M	15B	**45.5**	95.9	97.9	90.0	4.5	95.4	11.4	0.0	0.0	14.4	5.8	0.0	0.0
2D-RoPE (ours)	730M	15B	45.2	93.0	98.2	92.4	**91.3**	**99.4**	**74.6**	**12.8**	0.0	**92.8**	**76.7**	**45.7**	**10.9**
RoPE	1.4B	28B	45.9	84.5	87.2	82.3	10.0	99.6	26.8	0.0	0.0	21.6	8.7	0.0	0.0
H-RoPE (ours)	1.4B	28B	**47.2**	93.4	**98.5**	**94.5**	62.3	**100.0**	73.6	3.6	0.0	**100.0**	**98.1**	55.3	5.5
2D-RoPE (ours)	1.4B	28B	46.8	**98.3**	96.4	92.6	**86.0**	**100.0**	**100.0**	**92.0**	**61.8**	**100.0**	**98.1**	**92.6**	**87.3**
RoPE	730M	100B	46.0	76.9	80.9	70.5	0.4	99.6	29.2	0.0	0.0	24.7	8.7	1.1	1.8
H-RoPE (ours)	730M	100B	**46.2**	83.7	89.9	**89.3**	38.3	99.2	52.1	0.0	0.0	**100.0**	98.1	28.7	1.8
2D-RoPE (ours)	730M	100B	**46.2**	**87.3**	**93.1**	88.6	**86.5**	**100.0**	**96.3**	**72.8**	**15.8**	**100.0**	**99.0**	**94.7**	**74.5**

attention score is determined by the 2D relative coordinate, and choose the value vectors so that binary tokens 0 and 1 produce opposite logits. The correct source token has a relative coordinate "same column, previous row", while every wrong source position either has a nonzero column offset or lies in the wrong row. Random frequencies ensure that, simultaneously over all column offsets up to the target length, these wrong positions do not accidentally match the correct relative coordinate. Therefore, the attention mass on the correct source token exceeds $1/2$, which is enough for the binary value vectors to predict the correct copied token.

The detailed theoretical setup of this subsection is in Appendix K.1. A more detailed version of Theorem 3.1 and its proof are in Appendix K.2.

4. Experiments on LLM Pretraining

To validate the effectiveness of 2D-RoPE in empirical language modeling, we first pretrain Transformer language models from scratch and compare 2D-RoPE against RoPE. Then, we finetune our different models on the synthetic binary copy dataset and the Python List Conversion datasets.

Model Configuration. We experiment with the Qwen3 (Yang et al., 2025a) architecture, which is one of the most widely used open-source LLMs. To ensure fair comparison, we keep everything unchanged and only replace RoPE with 2D-RoPE. We experiment with different models sizes (350M, 730M, and 1.4B parameters). See Appendix F.1 for more details.

Pretraining Configuration. The models are pretrained with the DCLM corpus (Li et al., 2024) with 2K context length using the typical optimizer, LR scheduler, and hyperparameters (see Appendix F.2 for more details). We use a data-to-parameter ratio of 20 (i.e., Chinchilla law (Hoffmann et al., 2022)). Furthermore, we experiment with an overtrained setting that is common in state-of-the-art LLMs, where a 730M model is pretrained on 100B token.

Finetuning Configuration. Each model is finetuned on the imbalanced binary copy task (described in Section 2), using 200K examples and 2K context length (1K maximum input/output tokens). For each model, we swept learning rate (LR) values of $\{3 \times 10^{-5}, 5 \times 10^{-5}, 1 \times 10^{-4}\}$ and arrive at 5×10^{-5} because it has best overall performance. See Appendix F.4 for more details.

Hybrid RoPE. To show the effectiveness of 2D-RoPE, we construct a Hybrid-RoPE (H-RoPE) model between RoPE and 2D-RoPE, in which RoPE and 2D-RoPE are used alternately across Transformer blocks. H-RoPE serves as an investigation on whether partial use of 2D positional structure is sufficient to improve length generalization in copying.

4.1. LLM Evaluation Details.

We evaluate the models on the following benchmarks.

Common-Sense Reasoning. (CSR) To evaluate the common-sense reasoning abilities of LLMs, we use the average accuracy in the following set of tasks: ARC-easy, ARC-challenge (Clark et al., 2018), HellaSwag (Zellers et al., 2019), WinoGrande (Sakaguchi et al., 2019), MMLU (Hendrycks et al., 2021), LAMBADA (Paperno et al., 2016), PIQA (Bisk et al., 2019), and BoolQ (Clark et al., 2019). We use LM-Evaluation-Harness (Gao et al., 2024) for evaluation.

Needle-in-a-Haystack. Our needle-in-a-haystack task is the same as the NIAH-Single tasks from RULER, but with one change: we add newline characters before and after the needle (which is a sentence). This change enables 2D-RoPE to better copy the needle from the context.

Binary Copy. We evaluate on the Imbalanced and Recursive-Flip binary copy tasks as described in Section 2.1. The first task is an in-domain copy task (the models are finetuned on it), and the second task is an out-of-domain task that tests the transferability of our method on other copying tasks.

4.2. LLM Results

As shown in Table 1, LLMs with 2D-RoPE have superior length generalization in the binary copy tasks, in both in-domain (Imbalanced) and out-of-domain (Recursive-Flip) settings. This result shows that 2D-RoPE can exhibit its advantage in copying in practical large-scale LLM training. Models with 2D-RoPE (2D-RoPE and H-RoPE) also show better NIAH performance when the context length exceeds the maximum training context length, implying that copying abilities is beneficial to real-world in-context retrieval tasks. Finally, models with 2D-RoPE outperform RoPE models in CSR except for the smallest-scale model, implying that learning copying does not result in degraded general language modeling capabilities at scale.

Python List Conversion Task. To demonstrate the practical importance of the ability to copy, we finetune the 350M model on the Python List Conversion task (described in Section 2.1). After finetuning, 2D-RoPE achieves perfect generalization to sequences up to 3 times longer than those seen during training. In comparison, H-RoPE also generalizes to about three times the training length, but underperforms 2D-RoPE. Meanwhile, RoPE shows limited length generalization.

Auto-2D-RoPE: Beyond the Line-Break Separator. 2D-RoPE uses line-break token `<\n>` to construct 2D positional coordinates, which means that it almost reduces to RoPE when the input text contains no, or very few, line breaks. To address this concern, we introduce an alternative called Auto-2D-RoPE. Unlike 2D-RoPE, Auto-2D-RoPE does not rely on an explicitly specified line-break token to define rows and columns. Instead, it learns to construct 2D position IDs directly from the input sequence.

Concretely, Auto-2D-RoPE represents the 2D position ID of each token through the cumulative product of a learnable 2D transformation matrix along the sequence. Intuitively, this allows the model to automatically decide how positional information should evolve as it scans the input, rather than using a fixed rule such as "increase the row index after every newline and reset the column index." Similar cumulative-product constructions have appeared in prior attention designs (Tan et al., 2025; Yang et al., 2026), and the idea of making position IDs learnable has also been explored in recent works (Li et al., 2025; Lahoti et al., 2026). We provide the full architectural details of Auto-2D-RoPE in Appendix G.

Empirically, Auto-2D-RoPE achieves strong length generalization on copy tasks where we replace the newline token `<\n>` with another character $*$. In this setting, the original 2D-RoPE can no longer identify row boundaries and effectively degenerates to standard RoPE, showing little or no length generalization. In contrast, Auto-2D-RoPE still learns a useful 2D positional structure and generalizes to longer sequences as shown in Figure 8.

5. Conclusion

In this paper, we reveal a surprising fact that current LLMs still fail on copy: given a string as input, reproduce the same string as output, possibly with minor adaptations to match a required output format, even when the inputs are well within their context lengths. Our experiments and theoretical analysis suggest that this failure is closely related to repeated structures in the input, and copying requires a position-based alignment between source and target tokens. Motivated by this, we introduce 2D-RoPE, which views text as a 2D grid and turns copying into same-column retrieval. This simple change leads to strong length generalization in binary copy experiments and improves long-context retrieval in pretrained LLMs. We further introduce Auto-2D-RoPE to reduce the reliance on explicit line breaks.

Overall, our results show that the way positional information is represented can strongly affect even the most basic algorithmic behaviors of language models. We hope this work encourages further exploration of positional encodings that better capture the latent structure of text and support more reliable length generalization.

Several directions remain open for future study. (i) A thorough theoretical understanding of why multi-layer RoPE-based transformers fail to copy is lacking. (ii) How to generalize and scale up Auto-2D-RoPE to a more practical LLM training regime may be of significant value. (iii) Also, a further theoretical understanding of why 2D-RoPE can learn binary copy in the training process is underexplored.

References

Anil, C., Wu, Y., Andreassen, A., Lewkowycz, A., Misra, V., Ramasesh, V., Slone, A., Gur-Ari, G., Dyer, E., and Neyshabur, B. Exploring length generalization in large language models. *Advances in Neural Information Processing Systems*, 35:38546–38556, 2022.

Awasthi, P. and Gupta, A. Improving length-generalization in transformers via task hinting. *arXiv preprint arXiv:2310.00726*, 2023.

Bhattamishra, S., Ahuja, K., and Goyal, N. On the ability and limitations of transformers to recognize formal languages. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 7096–7116, 2020.

Bisk, Y., Zellers, R., Bras, R. L., Gao, J., and Choi, Y. Piqa: Reasoning about physical commonsense in natural language, 2019. URL https://arxiv.org/abs/1911.11641.

Bricken, T. and Pehlevan, C. Attention approximates sparse distributed memory. *Advances in Neural Information Processing Systems*, 34:15301–15315, 2021.

Chang, Y. and Bisk, Y. Language models need inductive biases to count inductively. In *The Thirteenth International Conference on Learning Representations*, 2025.

Chen, L., Peng, B., and Wu, H. Theoretical limitations of multi-layer transformer. *arXiv preprint arXiv:2412.02975*, 2024a.

Chen, S., Sheen, H., Wang, T., and Yang, Z. Unveiling induction heads: Provable training dynamics and feature learning in transformers. In Globerson, A., Mackey, L., Belgrave, D., Fan, A., Paquet, U., Tomczak, J., and Zhang, C. (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 66479–66567. Curran Associates, Inc., 2024b. doi: 10.52202/079017-2127.

Chen, W., Lin, Y., Zhou, Z., Huang, H., Jia, Y., Cao, Z., and Wen, J.-R. Icleval: evaluating in-context learning ability of large language models. In *Proceedings of the 31st International Conference on Computational Linguistics*, pp. 10398–10422, 2025.

Chiang, D. and Cholak, P. Overcoming a theoretical limitation of self-attention. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7654–7664, 2022.

Chu, X., Su, J., Zhang, B., and Shen, C. Visionllama: A unified llama backbone for vision tasks. In *European Conference on Computer Vision*, pp. 1–18. Springer, 2024.

Clark, C., Lee, K., Chang, M.-W., Kwiatkowski, T., Collins, M., and Toutanova, K. Boolq: Exploring the surprising difficulty of natural yes/no questions, 2019. URL https://arxiv.org/abs/1905.10044.

Clark, P., Cowhey, I., Etzioni, O., Khot, T., Sabharwal, A., Schoenick, C., and Tafjord, O. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv:1803.05457v1*, 2018.

Feng, G., Zhang, B., Gu, Y., Ye, H., He, D., and Wang, L. Towards revealing the mystery behind chain of thought: a theoretical perspective. *Advances in Neural Information Processing Systems*, 36:70757–70798, 2023.

Gao, L., Tow, J., Abbasi, B., Biderman, S., Black, S., DiPofi, A., Foster, C., Golding, L., Hsu, J., Le Noac'h, A., Li, H., McDonell, K., Muennighoff, N., Ociepa, C., Phang, J., Reynolds, L., Schoelkopf, H., Skowron, A., Sutawika, L., Tang, E., Thite, A., Wang, B., Wang, K., and Zou, A. The language model evaluation harness, 07 2024. URL https://zenodo.org/records/12608602.

Grattafiori, A., Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Vaughan, A., Yang, A., Fan, A., Goyal, A., Hartshorn, A., Yang, A., Mitra, A., Sravankumar, A., Korenev, A., Hinsvark, A., Rao, A., Zhang, A., Rodriguez, A., Gregerson, A., Spataru, A., Roziere, B., Biron, B., Tang, B., Chern, B., Caucheteux, C., Nayak, C., Bi, C., Marra, C., McConnell, C., Keller, C., Touret, C., Wu, C., Wong, C., Ferrer, C. C., et al. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

Hendrycks, D., Burns, C., Basart, S., Zou, A., Mazeika, M., Song, D., and Steinhardt, J. Measuring massive multitask language understanding, 2021. URL https://arxiv.org/abs/2009.03300.

Heo, B., Park, S., Han, D., and Yun, S. Rotary position embedding for vision transformer. In *European Conference on Computer Vision*, pp. 289–305. Springer, 2024.

Hoffmann, J., Borgeaud, S., Mensch, A., Buchatskaya, E., Cai, T., Rutherford, E., Casas, D. d. L., Hendricks, L. A., Welbl, J., Cantor, A., et al. Training compute-optimal large language models. In *Advances in Neural Information Processing Systems*, volume 35, pp. 30040–30053, 2022.

Hsieh, C.-P., Sun, S., Kriman, S., Acharya, S., Rekesh, D., Jia, F., Zhang, Y., and Ginsburg, B. Ruler: What's the real context size of your long-context language models? *arXiv preprint arXiv:2404.06654*, 2024.

Hu, S., Tu, Y., Han, X., Cui, G., He, C., Zhao, W., Long, X., Zheng, Z., Fang, Y., Huang, Y., Zhang, X., Thai, Z. L., Wang, C., Yao, Y., Zhao, C., Zhou, J., Cai, J., Zhai, Z., Ding, N., Jia, C., Zeng, G., dahai li, Liu, Z., and Sun, M. MiniCPM: Unveiling the potential of small language models with scalable training strategies. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=3X2L2TFr0f.

Huang, X., Yang, A., Bhattamishra, S., Sarrof, Y., Krebs, A., Zhou, H., Nakkiran, P., and Hahn, M. A formal framework for understanding length generalization in transformers. In *The Thirteenth International Conference on Learning Representations*, 2024.

Jeevan, P. and Sethi, A. Resource-efficient hybrid x-formers for vision. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pp. 2982–2990, 2022.

Jelassi, S., d'Ascoli, S., Domingo-Enrich, C., Wu, Y., Li, Y., and Charton, F. Length generalization in arithmetic transformers. *arXiv preprint arXiv:2306.15400*, 2023.

Jelassi, S., Brandfonbrener, D., Kakade, S. M., and Malach, E. Repeat after me: Transformers are better than state space models at copying. In *International Conference on Machine Learning*, pp. 21502–21521. PMLR, 2024.

Kazemnejad, A., Padhi, I., Natesan Ramamurthy, K., Das, P., and Reddy, S. The impact of positional encoding on length generalization in transformers. *Advances in Neural Information Processing Systems*, 36:24892–24928, 2023.

Lahoti, A., Li, K., Chen, B., Wang, C., Bick, A., Kolter, J. Z., Dao, T., and Gu, A. Mamba-3: Improved sequence modeling using state space principles. In *The Fourteenth International Conference on Learning Representations*, 2026.

Li, H., Zhao, T., Cai, D., and Sproat, R. Repo: Language models with context re-positioning. *arXiv preprint arXiv:2512.14391*, 2025.

Li, J., Fang, A., Smyrnis, G., Ivgi, M., Jordan, M., Gadre, S., Bansal, H., Guha, E., Keh, S., Arora, K., et al. Datacomp-lm: In search of the next generation of training sets for language models. *Advances in Neural Information Processing Systems*, 37:14200–14282, 2024.

Lin, H. M. and Cheng, H.-T. Gemini achieves gold-medal level at the International Collegiate Programming Contest World Finals. 2025.

Loshchilov, I. and Hutter, F. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

Luong, T. and Lockhart, E. Advanced version of Gemini with Deep Think officially achieves gold-medal standard at the International Mathematical Olympiad. 2025.

Merrill, W. and Sabharwal, A. The expressive power of transformers with chain of thought. In *The Twelfth International Conference on Learning Representations*, 2024.

Morwani, D., Edelman, B. L., Oncescu, C.-A., Zhao, R., and Kakade, S. M. Feature emergence via margin maximization: case studies in algebraic tasks. In *The Twelfth International Conference on Learning Representations*, 2024.

Olsson, C., Elhage, N., Nanda, N., Joseph, N., DasSarma, N., Henighan, T., Mann, B., Askell, A., Bai, Y., Chen, A., et al. In-context learning and induction heads. *arXiv preprint arXiv:2209.11895*, 2022.

OpenAI. Competitive programming with large reasoning models. *arXiv preprint arXiv:2502.06807*, 2025.

Paperno, D., Kruszewski, G., Lazaridou, A., Pham, Q. N., Bernardi, R., Pezzelle, S., Baroni, M., Boleda, G., and Fernández, R. The lambada dataset: Word prediction requiring a broad discourse context, 2016. URL https://arxiv.org/abs/1606.06031.

Pérez, J., Barceló, P., and Marinkovic, J. Attention is turing-complete. *Journal of Machine Learning Research*, 22(75): 1–35, 2021.

Press, O., Smith, N., and Lewis, M. Train short, test long: Attention with linear biases enables input length extrapolation. In *International Conference on Learning Representations*, 2021.

Ramsauer, H., Schäfl, B., Lehner, J., Seidl, P., Widrich, M., Gruber, L., Holzleitner, M., Adler, T., Kreil, D., Kopp, M. K., et al. Hopfield networks is all you need. In *International Conference on Learning Representations*, 2021.

Sakaguchi, K., Bras, R. L., Bhagavatula, C., and Choi, Y. Winogrande: An adversarial winograd schema challenge at scale, 2019. URL https://arxiv.org/abs/1907.10641.

Sanford, C., Hsu, D. J., and Telgarsky, M. Representational strengths and limitations of transformers. *Advances in Neural Information Processing Systems*, 36:36677–36707, 2023.

Singh, A. K., Moskovitz, T., Hill, F., Chan, S. C., and Saxe, A. M. What needs to go right for an induction head? a mechanistic study of in-context learning circuits and their formation. In *Forty-first International Conference on Machine Learning*, 2024.

Smart, M., Bietti, A., and Sengupta, A. M. In-context denoising with one-layer transformers: Connections between attention and associative memory retrieval. *Proceedings of Machine Learning Research*, 267:55950–55971, 2025.

Su, J., Ahmed, M., Lu, Y., Pan, S., Bo, W., and Liu, Y. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

Tan, S., Yang, S., Courville, A., Panda, R., and Shen, Y. Scaling stick-breaking attention: An efficient implementation and in-depth study. In *International Conference on Learning Representations*, 2025.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Wang, J., Ji, T., Wu, Y., Yan, H., Gui, T., Zhang, Q., Huang, X.-J., and Wang, X. Length generalization of causal transformers without position encoding. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 14024–14040, 2024a.

Wang, W., Zhang, S., Ren, Y., Duan, Y., Li, T., Liu, S., Hu, M., Chen, Z., Zhang, K., Lu, L., et al. Needle in a multimodal haystack. *Advances in Neural Information Processing Systems*, 37:20540–20565, 2024b.

Weiss, G., Goldberg, Y., and Yahav, E. Thinking like transformers. In *International Conference on Machine Learning*, pp. 11080–11090. PMLR, 2021.

Wen, K., Dang, X., and Lyu, K. Rnns are not transformers (yet): The key bottleneck on in-context retrieval. In *The Thirteenth International Conference on Learning Representations*, 2025.

Yang, A. and Chiang, D. Counting like transformers: Compiling temporal counting logic into softmax transformers. In *First Conference on Language Modeling*, 2024.

Yang, A., Li, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Gao, C., Huang, C., Lv, C., et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025a.

Yang, B., Venkitesh, B., Gnaneshwar, D., Lin, H., Cairuz, D., Blunsom, P., and Locatelli, A. Rope to nope and back again: A new hybrid attention strategy. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025b.

Yang, S., Shen, Y., Wen, K., Tan, S., Mishra, M., Ren, L., Panda, R., and Kim, Y. PaTH attention: Position encoding via accumulating householder transformations. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026. URL https://openreview.net/forum?id=ZBlHEeSvKd.

Yao, S., Peng, B., Papadimitriou, C., and Narasimhan, K. Self-attention networks can process bounded hierarchical languages. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 3770–3785, 2021.

Zellers, R., Holtzman, A., Bisk, Y., Farhadi, A., and Choi, Y. Hellaswag: Can a machine really finish your sentence?, 2019. URL https://arxiv.org/abs/1905.07830.

Zhou, H., Bradley, A., Littwin, E., Razin, N., Saremi, O., Susskind, J. M., Bengio, S., and Nakkiran, P. What algorithms can transformers learn? a study in length generalization. In *The Twelfth International Conference on Learning Representations*, 2023.

Zhou, Y., Alon, U., Chen, X., Wang, X., Agarwal, R., and Zhou, D. Transformers can achieve length generalization but not robustly. *arXiv preprint arXiv:2402.09371*, 2024.

A. Additional Related Works

Positional Encoding. The attention mechanism relies on position encoding methods to model the positional information of tokens (Vaswani et al., 2017). Most state-of-the-art LLMs employ RoPE (Su et al., 2024), which have demonstrated strong language modeling performance and length generalization. Kazemnejad et al. (2023) shows that attention with just a causal mask can reconstruct absolute and relative position information. More recently, Yang et al. (2026) proposes a position encoding scheme based on accumulating Householder transformations and showed that it can solve NC^1-complete problems under AC^0-reductions. Critically, no previous works have explored the influence of position encoding on the copying abilities of language models.

Length Generalization. Many early studies showed that transformers sometimes succeed and sometimes fail at length-generalization, correlated with multiple factors such as positional encoding, data format, and other training hyperparameters (Bhattamishra et al., 2020; Anil et al., 2022; Kazemnejad et al., 2023; Awasthi & Gupta, 2023; Jelassi et al., 2023; Wang et al., 2024a; Zhou et al., 2023; 2024; Chang & Bisk, 2025; Jelassi et al., 2024). Specifically on the copy task, Transformers trained to copy short strings often do not generalize well to longer strings when the input string includes repeated substrings (Zhou et al., 2024; Morwani et al., 2024). Theoretically, Huang et al. (2024) introduced a formal framework, proving that Transformers can solve a class of problems expressible by the C-RASP formalism (Yang & Chiang, 2024). Unfortunately, the repeated copy is provably difficult in the sense of C-RASP expressiveness.

The Representation Power of Transformer. There is a long line of work studying the representation power of Transformer (Pérez et al., 2021; Yao et al., 2021; Chiang & Cholak, 2022; Sanford et al., 2023). Merrill & Sabharwal (2024); Feng et al. (2023) analyze the representation power of Transformer with Chain-of-Thought. Wen et al. (2025) compares the expressive ability of Transformer with Recurrent Neural Networks. Later, Chen et al. (2024a) showed the limitation of multi-layer Transformers' expressive power.

Copy Capability of Transformers. Copying from context has been studied in several related settings. Some works use synthetic copy tasks to study Transformer length generalization (Kazemnejad et al., 2023; Huang et al., 2024; Jelassi et al., 2024), while other benchmarks include copying as part of broader LLM evaluations (Chen et al., 2025; Wang et al., 2024b; Hsieh et al., 2024). However, these works do not isolate exact copying of user-provided strings as a standalone failure mode of frontier LLMs, nor do they provide a positional-encoding-based remedy. In contrast, we directly evaluate modern LLMs on exact copy tasks, connect the failure to positional alignment, and propose 2D-RoPE to better align source and target tokens.

B. Test Details of the LLMs in Binary Copy and Python List Conversion

We give the details of the closed-model Copy benchmarks reported in Figure 1 and Figure 2. All prompt templates are provided in Appendix L. All closed-model API evaluations were run between June 11 and June 14, 2026. Unless stated otherwise, each sample was attempted at most three times in total, including the initial call. In the final evaluation logs used for the reported numbers, every requested sample received a valid API response within the allowed retry budget. No sample was discarded or replaced due to an API failure; hence all reported failures in the accuracy tables correspond to incorrect model outputs rather than API failures.

B.1. Frontier LLMs Copying Test

Here we give details for Figure 1.

Models and API endpoints. We evaluate five closed frontier models: `gpt-5.5`, `claude-opus-4-7`, `gemini-3.1-pro-preview`, `deepseek-v4-pro`, and `qwen3.6-plus`. We use the OpenAI Responses API for `gpt-5.5`, the Anthropic Messages API for `claude-opus-4-7`, Gemini Vertex Express for `gemini-3.1-pro-preview`, and OpenAI-compatible chat-completion APIs for `deepseek-v4-pro` and `qwen3.6-plus`. We do not set explicit stop sequences. We use temperature 0 where supported; for `gpt-5.5`, we do not pass an explicit temperature parameter and use the provider default. For Gemini, we set the thinking level to `LOW`; for DeepSeek and Qwen, we disable thinking mode.

Recursive-Flip Binary Copy. For the first binary-copy evaluation, we test the Recursive-Flip distribution defined in Section 2.1. We use 50 samples for each $k \in \{7, 8, 9, 10, 11\}$, with global random seed 20260414. For each k, the

logical target length n satisfies

$$2^k \leq n \leq 2^{k+1} - 1.$$

Here, length refers to the number of logical binary units. For each sample, the length is uniformly and randomly selected from this region. To align tokenization across models, we use model-dependent encodings: for `gpt-5.5`, `claude-opus-4-7`, and `deepseek-v4-pro`, $\mathcal{V} = \{\texttt{000}, \texttt{111}\}$; for `gemini-3.1-pro-preview` and `qwen3.6-plus`, $\mathcal{V} = \{\texttt{0}, \texttt{1}\}$. This ensures that each binary unit is one token for the tested model.

Imbalanced Binary Copy. For the second binary-copy evaluation, we use an imbalanced distribution with p fixed to be 0.95 (see the original definition of imbalanced data in Section 2.1). Here let $\mathcal{V} = \{\texttt{" a"}, \texttt{" b"}\}$, so that

$$\Pr[\texttt{a}] = 0.95, \qquad \Pr[\texttt{b}] = 0.05.$$

Both symbols are one token in all five tested models. Thus, the model is asked to copy a space-separated symbolic sequence rather than a bare binary string. For output normalization, we extract all letters matching `[abAB]`, map `a`/`A` back to binary 1, and map `b`/`B` back to binary 0. Exact match compares this normalized binary string with the original binary target, while strict match compares the raw stripped output with the exact symbolic target string.

Binary-copy API configuration. For Recursive-Flip binary copy, the models are evaluated concurrently with one model-level worker per model. Within each model, the maximum number of concurrent sample requests is 10 for `gpt-5.5`, `claude-opus-4-7`, `deepseek-v4-pro`, and `qwen3.6-plus`, and 5 for `gemini-3.1-pro-preview`. The request timeout is 2000 seconds, and the sleep times before the second and third attempts are 2 and 4 seconds, respectively. For OpenAI-compatible clients, we set `max_retries` $= 0$ so that retries are controlled by our explicit retry loop. The output-token budgets for Recursive-Flip binary copy are

Model	`gpt`	`claude`	`gemini`	`deepseek`	`qwen`
Max output tokens	50000	8192	16384	8192	8192

For imbalanced symbolic copy, the maximum output-token budget is computed as $4 \cdot \text{target_length} + 128$.

Python List Conversion. We also evaluate the same five models on a Python List Conversion task. Each example is generated from a synthetic one-dimensional physical signal. We sample one scenario uniformly from nine signal families: simple harmonic motion, triangle wave motion, sawtooth scan motion, rectified AC signal, clipped sensor oscillation, relaxation oscillator, Fourier periodic waveform, pulse train, and pendulum-like oscillation. For each example, we sample a period p, generate one cycle of length p, and then obtain the length-n sequence by periodic indexing:

$$y_k = \text{cycle}_{k \bmod p}, \qquad k = 0, \ldots, n-1.$$

The target output is the same sequence written as a Python list,

$$[y_0, y_1, \ldots, y_{n-1}].$$

All values are rounded to two decimal places. In the API evaluation, we use $2 \leq p \leq 10$ and evaluate lengths

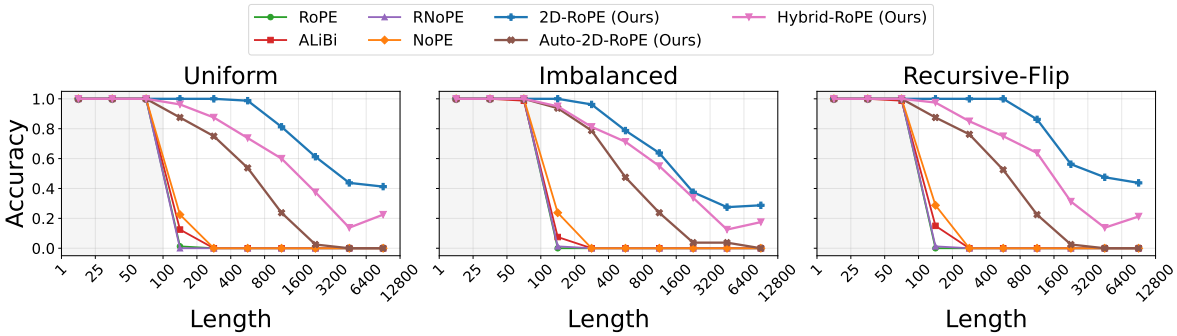
$$n \in \{200, 300, 400, 500\}.$$

For each length, we generate 50 examples using random seed 2030. The datasets are saved before model evaluation, and all models are evaluated on the same saved examples. The API endpoints and output-token budgets are the same as in Recursive-Flip binary copy, with a global request timeout of 6000 seconds. For evaluation, we strip Markdown code fences from both the model output and the gold output, then extract all numbers matching the regular expression `-?\d+(?:\.\d+)?`. Sequence exact match is true if and only if the extracted numeric sequence from the model output exactly equals that from the gold output.

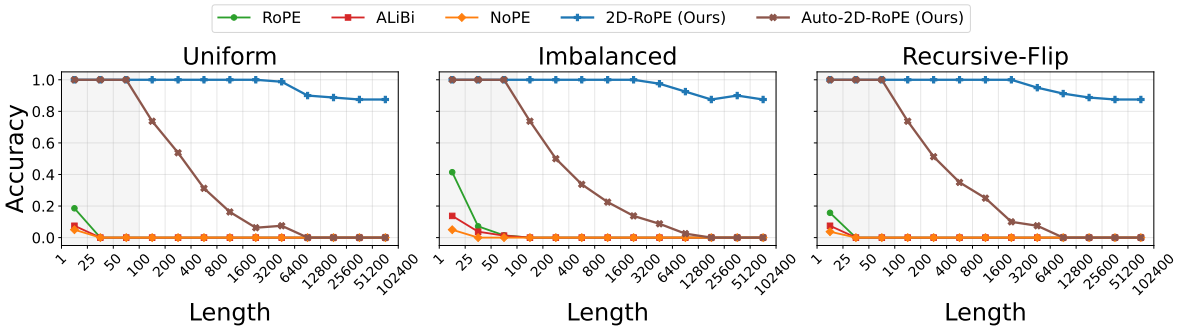
B.2. Repeat-structure Copy Test for Frontier LLMs

Here we give details for Figure 2.

Models. We evaluate `gpt-5.5`, `gemini-3.1-pro-preview`, and `qwen3.6-plus`. The resources are same as the previous subsection.



(a) Length generalization results for different positional encodings with 12 layers and 12 heads.



(b) Length generalization results for different positional encodings with 1 layer and 2 heads.

Figure 5. Length generalization for training-from-scratch experiments.

Data distribution. We use random seed 1320, target length 1000, base lengths

$$m \in \{4, 6, 8, 10\},$$

and flip probabilities

$$p \in \{0.02, 0.05, 0.10, 0.15, 0.20, 0.25, 0.30\}.$$

For each valid pair (m, p), we generate 50 samples. A pair is included only if

$$(1-p)^m \geq 0.3.$$

We first sample a base string of length m uniformly from the alphabet {`" a"`, `" b"`}, repeat it until length 1000, truncate the final repeat if necessary, and then independently flip each token with probability p. The x-axis reports $(1-p)^m$, the probability that an entire repeated block remains unflipped.

API and evaluation. GPT-5.5 uses the OpenAI Responses API with `max_output_tokens` $= 5000$ and no explicit temperature parameter. Gemini uses temperature 0.0, `max_output_tokens` $= 10000$, and thinking level `LOW`. Qwen uses temperature 0.0, `max_output_tokens` $= 2500$, request timeout 300 seconds, request gap 1 second, `enable_thinking=False`, and `max_retries` $= 0$. Final plotted data are computed from merged logs keyed by

$$(\texttt{model}, \texttt{base_length}, p).$$

Output normalization keeps only letters `a` and `b` from the raw response and maps them back to the symbolic tokens. Exact match is true only if the normalized prediction has length 1000 and agrees with the corrupted input sequence at every position.

C. Details of Synthetic Experiments on Binary Copy

We conduct training-from-scratch experiments on the synthetic binary copy dataset. For each binary string $s \in \{0,1\}^*$, the input sequence takes the form

$$\mathcal{S} := (\texttt{A}_1, \texttt{A}_2, s, \texttt{B}_1 = \texttt{<\backslash n>}, \texttt{B}_2, s),$$

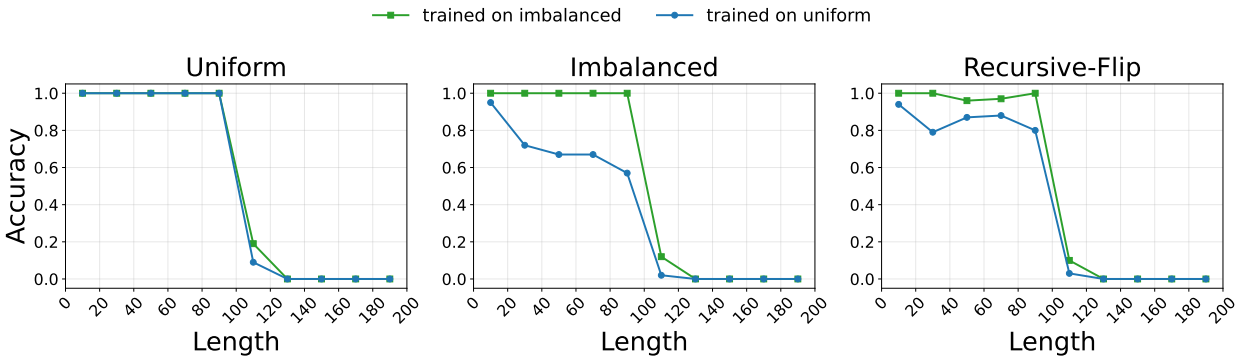


Figure 6. Length generalization for training-from-scratch experiments in 2 distributions.

where the model is trained to predict the next token autoregressively, including the copied string after $\mathtt{B}_2$ and the final <EOS> token at the end of the sequence.

Data Distributions. As introduced in Section 2.1, we consider three data distributions: *uniform*, *imbalanced*, and *recursive-flip*. All training is conducted on imbalanced data with string lengths from 1 to 100. For the imbalanced distribution, the probability of sampling the first binary token is chosen from

$$\{0.05, 0.15, 0.3, 0.5, 0.7, 0.85, 0.95\},$$

and tokens are then sampled independently according to this probability. At test time, we evaluate on all three distributions. Thus, uniform and recursive-flip test strings are out-of-distribution relative to the training distribution.

Model Architectures. We train Transformers from scratch under two model sizes. The shallow model has 1 layer and 2 attention heads, where each head has dimension 512. The deeper model has 12 layers and 12 attention heads, where each head has dimension 128. Thus, the embedding dimensions are 1024 and 1536, respectively. We set dropout to 0 and use bias-free linear layers. For each architecture and each positional encoding, we train 8 models with different random seeds and report the average accuracy. We average over seeds because training on the copy task shows noticeable variance across runs, especially near the boundary between interpolation and extrapolation.

Training Configuration. The two model sizes use different optimization settings. The different hyperparameters are in Appendix 2. For both settings, we use weight decay 0.01, $\beta_2 = 0.95$, and a warmup length of 100 iterations. The training batch size is 64 with gradient accumulation over 4 steps. The maximum context length is set to 260000 for 1-layer model and 26000 for 12-layer model, which is large enough for all training and evaluation sequences considered in these experiments.

Evaluation. During training, strings have length at most 100. At evaluation time, we test models on longer strings to measure length generalization. We evaluate lengths beyond the training range up to the longest test length used in the corresponding figure, with test lengths are uniformly selected in each interval. For each run, we sample 10 strings in each interval independently.

Results. Figure 5 shows the full length-generalization results for all test distributions. The main text reports the Recursive-Flip setting because it is both out-of-distribution and especially challenging due to repeated substrings. Across the full results, standard positional encodings fail to extrapolate reliably beyond the training length range, while 2D-RoPE-based methods show substantially stronger length generalization.

Training Configuration. As discussed before, we train transformers from scratch on the binary copy task under two model sizes. The first is a shallow model with 1 layer and 2 attention heads, where each head has dimension 512. The second is a deeper model with 12 layers and 12 attention heads, where each head has dimension 128. For each architecture and each positional encoding, we train 8 models with different random seeds and report the average accuracy. We average over seeds because training on the copy task shows noticeable variance across runs, especially near the boundary between interpolation and extrapolation.

Table 2. Training hyperparameters for each positional encoding in the length generalization experiments. Peak LR and Final LR denote the maximum and minimum learning rates used in the cosine decay schedule.

Setting	Positional Encoding	Peak LR	Final LR	Training Steps
12 layers, 12 heads				
12L	RoPE	5×10^{-5}	1×10^{-6}	10000
12L	ALiBi	5×10^{-5}	1×10^{-6}	10000
12L	NoPE	5×10^{-5}	1×10^{-6}	10000
12L	RNoPE	5×10^{-5}	1×10^{-6}	10000
12L	2D-RoPE	5×10^{-5}	1×10^{-6}	10000
12L	Auto-2D-RoPE	5×10^{-5}	1×10^{-6}	30000
12L	Hybrid RoPE	5×10^{-5}	1×10^{-6}	30000
1 layer, 2 heads				
1L	RoPE	5×10^{-4}	5×10^{-5}	60000
1L	ALiBi	5×10^{-4}	5×10^{-5}	60000
1L	NoPE	5×10^{-4}	5×10^{-5}	60000
1L	2D-RoPE	5×10^{-4}	1×10^{-6}	60000
1L	Auto-2D-RoPE	5×10^{-4}	1×10^{-4}	10000

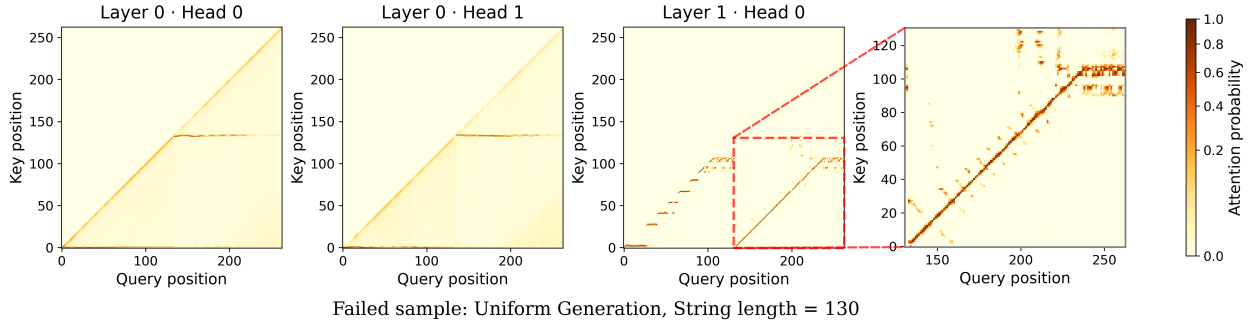


Figure 7. Attention map of a RoPE-based model on a failed sample with sequence length beyond the training length. In Layer 0, attention heads exhibit both **horizontal patterns** (attending to special tokens in between the input string and the output, and the tokens before the string as a locator) and **diagonal patterns** (semantic/induction-style behavior). In Layer 1, one can see that RoPE would attend to wrong tokens near themselves periodically.

D. Further Discussion of Multi-Layer RoPE Learns Length-Dependent Pattern

We provide a more detailed discussion of the attention patterns in Figure 3 and Figure 7. The main text shows that RoPE does not learn a clean position-based copy map even on a failed example within the training length range. Here, we further examine what happens beyond the training length.

Within the Training Length. In Figure 3, the last-layer attention should ideally align each output position with its corresponding input position. However, the zoomed attention map is not a clean diagonal alignment. The head attends not only to the correct source positions, but also periodically assigns large attention scores to nearby wrong tokens around the diagonal. This suggests that the final copy decision is affected by local-context matching: when repeated or similar substrings appear, locally similar positions can compete with the correct source position. Thus, even within the training length range, RoPE can fail because the learned copy mechanism is not purely position-based.

Beyond the Training Length. Figure 7 further shows why this learned mechanism does not length-generalize. In Layer 0, the two heads contain horizontal locator patterns that attend to special tokens around the boundary between the input string and the output string, such as the delimiters in [<\n>, <I:>, <O:>]. These patterns provide a position-related signal and can help the model infer source-target alignment within the seen length range. However, when the sequence length goes beyond the maximum training length, the locator pattern no longer remains stable: after the copied position exceeds the training range, the horizontal attention pattern becomes discontinuous and eventually breaks. This indicates that the model's position-based component is not a true length-invariant copy algorithm, but a locator-style shortcut tied to the training lengths.

At the same time, Layer 0 also contains diagonal patterns, which resemble local-context or induction-style behavior (Olsson et al., 2022; Singh et al., 2024; Jelassi et al., 2024). These patterns can attend to nearby or locally similar tokens, but they cannot uniquely identify the correct source position when repeated substrings are present. Therefore, the trained RoPE model combines two imperfect mechanisms: a locator-style positional shortcut that does not generalize in length, and a context-based matching mechanism that becomes ambiguous on repeated structures. This explains why the last-layer attention becomes noisy and why the model fails to implement the ideal position-based copy rule suggested by Theorem 2.2.

E. Implementation Details of 2D-RoPE

E.1. Rotational Frequency Hyperparameter.

In RoPE, there is a hyperparameter θ that determines the rotational frequency. For 2D-RoPE, the two dimensions can have separate rotational frequencies, θ_x, θ_y. However, for simplicity, we keep $\theta_x = \theta_y$. Moreover, we use the default hyperparameter from the original RoPE paper, which is $\theta = 10,000$, and we set $\theta_x = \theta_y = 1,000$. Our preliminary experiments show that setting $\theta_x = \theta_y = 100$ or $\theta_x = \theta_y = 10,000$ does not yield significantly different results.

E.2. Implementation Details and Comparison with Vision 2D-RoPE

RoPE-style 2D positional encodings have been explored in computer vision, where each image patch has a natural spatial coordinate. For example, prior vision models apply RoPE to patch tokens using their height and width coordinates, so that attention can encode relative spatial displacement on the image grid (Jeevan & Sethi, 2022; Heo et al., 2024; Chu et al., 2024). In this setting, the 2D coordinates are given by the image layout itself.

As we mentioned in the previous section, our implementation differs in both motivation and coordinate construction. In language modeling, the input is normally treated as a 1D token sequence, and there is no fixed 2D grid analogous to image patches. We instead use line break tokens to induce a 2D structure in text. Each token receives a row ID, determined by the number of preceding line breaks, and a column ID, determined by its offset within the current line. Thus, the 2D coordinates are not externally given but are constructed from the textual structure. Further, we introduce an alternative termed Auto 2D-RoPE for 2D-RoPE to learn the 2D structure in context adaptively in Appendix G.

In our implementation, 2D-RoPE follows an axial decomposition: part of the head dimension is rotated according to the row ID, and the remaining part is rotated according to the column ID. We keep the standard causal attention mask for language modeling. This is also different from most vision applications, which typically use bidirectional attention over image patches.

F. LLM Experiment Details

Unless otherwise specified, the model training in the Section 4 follows the following settings.

F.1. LLM Model Configuration Details

The hyperparameters for the models involved in LLM experiments (Section 4) are described in Table 3. These hyperparameter values are chosen to be similar to popular open-source LLMs such as Qwen3 and Llama3 (Grattafiori et al., 2024). Each model uses the Llama3 tokenizer.

F.2. LLM Pretraining Configuration Details

Here, we describe more details for the LLM pretraining (Section 4) for reproducibility.

Learning Rate Scheduler. Each model was trained using the WSD LR scheduler (Hu et al., 2024), with different LR depending on the model size. The 350M, 730M, and 1.4B models use 5e-4, 5e-4, and 2e-4 max LR, respectively. Each of the LRs warms up (from 0) for 100 steps, and decays for the last 10% steps to $1/10$ of their max LR.

Data Batching. The batch size is always 32, and the maximum sequence length during pretraining is always 2048, so the number of tokens per batch is 512K. Each pretraining document is concatenated with an `<end_of_sequence>` delimiter until they exceed maximum sequence length, and then truncated down to 2048. The part that was truncated is

Table 3. The hyperparameters of the three model of different sizes involved in Section 4.

Hyperparameter	**350M**	**730M**	**1.4B**
Vocab size	32000	32000	32000
Layers	24	24	28
Hidden size	1024	1536	2048
MLP intermediate dim	3072	4096	5120
MLP activation function	SwiGLU	SwiGLU	SwiGLU
Attention heads	8	12	16
KV heads	8	12	16
Attention head size	128	128	128
RoPE θ	10,000	10,000	10,000
2D-RoPE θ_x, θ_y	1,000	1,000	1,000
Training Configuration			
Pretraining batch size	256	256	256
Finetuning batch size	64	64	64
Max length	2K	2K	2K
Pretraining max LR	5e-4	5e-4	4e-4
Finetuning max LR	5e-5	5e-5	5e-5

discarded. Each token can attend to all other tokens in other documents that was concatenated into the same sequence.

Optimizer. We use the same optimizer for both pretraining and finetuning, which is AdamW (Loshchilov & Hutter, 2019) with beta values of (0.9, 0.95), a weight decay of 0.1, and gradient clipping at 1.0.

Hardware. All LLM experiments were conducted on machines equipped with NVIDIA A800-80GB GPUs, using PyTorch and HuggingFace Transformer libraries. We used BF16 precision during both training and evaluation.

F.3. LLM Finetuning Configuration Details

Learning Rate Scheduler. We use the Cosine LR scheduler for finetuning, since the cost of rerunning finetuning is much lower than pretraining. The LR warms up for 100 steps then decays down to 0.

Data Batching. For finetuning, each sequence might have different sequence length, and we use attention masking to ensure that padding tokens are not attended to. Since the training dataset is synthesized, it does not exceed 2048 tokens and there is no truncation. We always use batch size of 64 during finetuning.

Optimizer and Hardware. The optimizer for finetuning is the same as the one used for pretraining, and all experiments were conducted on the same hardware, based on the same precision and implementation libraries.

Common-Sense Reasoning To evaluate the common-sense reasoning abilities of LLMs, we use the average accuracy in the following set of tasks: ARC-easy, ARC-challenge (Clark et al., 2018), HellaSwag (Zellers et al., 2019), WinoGrande (Sakaguchi et al., 2019), MMLU (Hendrycks et al., 2021), LAMBADA (Paperno et al., 2016), PIQA (Bisk et al., 2019), BoolQ (Clark et al., 2019). We use LM-Evaluation-Harness (Gao et al., 2024) for evaluation.

Needle-in-a-Haystack. Our needle-in-a-haystack task is the same as the NIAH-Single tasks from RULER, but with one change: we add newline characters before and after the needle (which is a sentence). This change enables 2D-RoPE to better copy the needle from the context.

F.4. LR Ablation in LLM Experiments

In the LLM experiments (Section 4), we swept different max LR in finetuning and chose the best LR that balanced the performance across different tasks. In this section, we report the results for each value of LR ($\{1 \times 10^{-4}, 5 \times 10^{-5}, 3 \times 10^{-5}\}$), which is shown in Table 4. Overall, the results show that using a larger LR leads to better copying abilities, but the CSR and NIAH performance degrades due to catastrophic forgetting. Interestingly, 2D-RoPE is generally more robust to the choice of LR, leading to less catastrophic forgetting in the case of a fairly large LR.

Table 4. LR ablation results in the LLM experiments. Each model was trained with 3 different LR values, and the main content reports the LR that achieves best overall performance in the three kinds of tasks (CSR, NIAH, and Copy), which is selected to be 5e-5 and is highlighted in orange.

Model	Size	PT Data	LR	CSR	NIAH 512	NIAH 1K	NIAH 2K	NIAH 4K	Copy (Imbalanced) 1K	Copy (Imbalanced) 2K	Copy (Imbalanced) 4K	Copy (Imbalanced) 8K	Copy (Recursive-Flip) 1K	Copy (Recursive-Flip) 2K	Copy (Recursive-Flip) 4K	Copy (Recursive-Flip) 8K
Different Model Scales with Chinchilla law																
RoPE	350M	7B	1e-4	40.9	74.5	74.5	60.1	0.1	100.0	16.8	0.0	0.0	26.8	11.7	1.1	1.8
RoPE	350M	7B	5e-5	41.4	93.7	92.1	87.0	1.9	97.2	14.8	0.0	0.0	19.6	8.7	0.0	0.0
RoPE	350M	7B	3e-5	41.6	93.9	94.1	89.5	4.3	93.2	9.6	0.0	0.0	21.6	8.7	0.0	0.0
H-RoPE	350M	7B	1e-4	41.2	92.0	98.1	93.8	66.3	100.0	98.2	18.8	0.2	100.0	99.0	72.3	27.3
H-RoPE	350M	7B	5e-5	41.2	93.1	98.8	94.4	57.9	99.6	57.8	0.0	0.0	99.0	91.3	39.4	1.8
H-RoPE	350M	7B	3e-5	41.3	92.7	98.1	93.7	48.8	84.8	33.4	0.0	0.0	100.0	98.1	55.3	5.5
2D-RoPE	350M	7B	1e-4	40.1	89.8	96.7	92.7	88.7	99.8	94.8	81.8	27.2	100.0	100.0	100.0	100.0
2D-RoPE	350M	7B	5e-5	40.4	91.9	97.9	93.9	92.8	97.4	69.2	33.4	2.4	97.9	92.2	75.5	43.6
2D-RoPE	350M	7B	3e-5	40.5	92.3	98.6	94.3	93.1	81.2	57.2	14.8	0.4	100.0	98.1	92.6	87.3
RoPE	730M	15B	1e-4	43.3	93.6	97.6	91.7	21.9	100.0	20.8	0.0	0.0	20.6	6.8	0.0	1.8
RoPE	730M	15B	5e-5	44.0	100.0	98.4	96.7	26.4	97.0	11.0	0.0	0.0	12.4	5.8	1.1	0.0
RoPE	730M	15B	3e-5	44.2	97.2	97.1	93.1	27.4	88.8	4.4	0.0	0.0	14.4	7.8	0.0	0.0
H-RoPE	730M	15B	1e-4	44.1	96.4	77.7	67.1	7.1	93.6	13.6	0.0	0.0	100.0	100.0	78.7	3.6
H-RoPE	730M	15B	5e-5	45.5	95.9	97.9	90.0	4.5	95.4	11.4	0.0	0.0	76.3	72.8	17.0	0.0
H-RoPE	730M	15B	3e-5	45.4	95.9	99.5	92.1	8.0	84.0	4.0	0.0	0.0	96.9	92.2	35.1	0.0
2D-RoPE	730M	15B	1e-4	44.3	91.7	95.9	87.0	80.1	100.0	100.0	89.6	24.2	100.0	98.1	88.3	90.9
2D-RoPE	730M	15B	5e-5	45.2	93.0	98.2	92.4	91.3	99.4	74.6	12.8	0.0	78.4	73.8	47.9	5.5
2D-RoPE	730M	15B	3e-5	45.1	99.8	99.3	97.3	96.2	92.8	41.4	3.6	0.0	96.9	89.3	56.4	36.4
RoPE	1.4B	28B	1e-4	36.8	0.0	0.0	0.0	0.0	100.0	31.0	0.0	0.0	20.6	8.7	0.0	1.8
RoPE	1.4B	28B	5e-5	45.9	84.5	87.2	82.3	10.0	99.6	26.8	0.0	0.0	15.5	4.9	0.0	0.0
RoPE	1.4B	28B	3e-5	46.0	92.3	96.9	96.3	11.2	99.8	23.2	0.0	0.0	15.5	7.8	1.1	0.0
H-RoPE	1.4B	28B	1e-4	46.8	8.5	6.3	5.2	2.3	100.0	92.4	38.8	5.2	17.5	5.8	1.1	0.0
H-RoPE	1.4B	28B	5e-5	47.2	93.4	98.5	94.5	62.3	100.0	73.6	3.6	0.0	12.4	1.9	0.0	0.0
H-RoPE	1.4B	28B	3e-5	47.2	90.9	97.5	93.8	61.5	99.8	65.8	2.8	0.0	14.4	5.8	0.0	0.0
2D-RoPE	1.4B	28B	1e-4	45.9	2.7	3.7	5.4	4.6	100.0	100.0	100.0	99.0	99.0	95.1	93.6	69.1
2D-RoPE	1.4B	28B	5e-5	46.8	98.3	96.4	92.6	86.0	100.0	100.0	92.0	61.8	66.0	54.4	26.6	1.8
2D-RoPE	1.4B	28B	3e-5	46.9	98.7	97.5	92.7	92.3	100.0	92.0	51.8	14.6	92.8	76.7	45.7	10.9
Overtrained Setting																
RoPE	730M	100B	3e-5	46.9	81.9	94.1	86.5	0.2	99.5	23.3	0.0	0.0	15.5	7.8	1.1	1.8
RoPE	730M	100B	5e-5	46.0	76.9	80.9	70.5	0.4	99.6	29.2	0.0	0.0	24.7	8.7	1.1	1.8
RoPE	730M	100B	1e-4	34.3	0.0	0.0	0.0	0.0	100.0	22.6	0.0	0.0	63.9	15.5	0.0	1.8
H-RoPE	730M	100B	3e-5	46.4	99.9	95.2	96.9	31.1	100.0	41.4	0.0	0.0	92.8	86.4	6.4	0.0
H-RoPE	730M	100B	5e-5	46.2	83.7	89.9	89.3	38.3	99.2	52.1	0.0	0.0	100.0	98.1	28.7	1.8
H-RoPE	730M	100B	1e-4	43.3	0.1	0.0	0.0	0.0	100.0	74.6	1.7	0.0	100.0	100.0	50.0	1.8
2D-RoPE	730M	100B	3e-5	46.3	86.0	95.4	91.6	89.3	100.0	95.5	37.6	0.0	99.0	98.1	84.0	34.5
2D-RoPE	730M	100B	5e-5	46.2	87.3	93.1	88.6	86.5	100.0	96.3	72.8	15.8	100.0	99.0	94.7	74.5
2D-RoPE	730M	100B	1e-4	43.4	0.0	0.1	0.1	0.0	100.0	99.8	94.1	40.7	100.0	100.0	100.0	81.8

F.5. Common-Sense Reasoning Detailed Results

For completeness, we report the performance of each model on each common-sense reasoning task mentioned in Appendix F.3, in Table 5.

G. Auto-2D-RoPE Discussion

We now describe the architecture of Auto-2D-RoPE in more detail. The goal is to keep the main advantage of 2D-RoPE, namely assigning each token a 2D position ID, while removing the hard reliance on a manually specified line-break token. In standard 2D-RoPE, the position ID of each token is updated by a fixed rule: the column index increases within a line, and the row index increases after a line break. In Auto 2D-RoPE, we instead let the model learn how the 2D position IDs should evolve along the sequence.

Table 5. The results of LLMs on each individual common-sense reasoning tasks. Orange rows highlights the LR value that is chosen to maximize the performance across common-sense reasoning, NIAH, and binary copy.

Model	Size	PT Data	LR	MMLU	LMBD	HS	PIQA	ARC-e	ARC-c	WG	BoolQ	Avg.
Different Model Scales with Chinchilla law												
RoPE	350M	7B	1e-4	23.6	27.4	34.4	64.1	42.9	24.2	51.8	58.7	40.9
RoPE	350M	7B	5e-5	23.7	28.8	35.2	64.8	43.9	24.2	51.7	59.3	41.4
RoPE	350M	7B	3e-5	23.8	28.7	35.3	64.9	44.0	24.6	51.9	59.4	**41.6**
H-RoPE	350M	7B	1e-4	25.8	28.7	35.2	64.3	43.7	24.1	50.3	57.4	41.2
H-RoPE	350M	7B	5e-5	24.5	28.4	35.5	65.0	43.6	24.9	50.9	56.5	41.2
H-RoPE	350M	7B	3e-5	24.4	28.1	35.6	65.4	43.9	25.3	51.1	56.5	41.3
2D-RoPE	350M	7B	1e-4	23.1	28.8	34.9	63.4	42.6	23.9	53.3	50.6	40.1
2D-RoPE	350M	7B	5e-5	23.2	27.8	35.3	64.2	43.1	23.5	52.5	53.5	40.4
2D-RoPE	350M	7B	3e-5	23.5	27.4	35.3	64.2	43.4	23.5	53.2	53.5	40.5
RoPE	730M	15B	1e-4	23.1	38.4	42.0	67.5	45.7	25.4	53.7	50.4	43.3
RoPE	730M	15B	5e-5	23.3	38.3	43.1	69.0	48.3	26.5	53.3	50.5	44.0
RoPE	730M	15B	3e-5	23.2	38.5	43.2	69.4	48.9	26.9	53.9	49.6	44.2
H-RoPE	730M	15B	1e-4	23.6	37.6	41.5	66.9	48.7	26.3	51.6	56.7	44.1
H-RoPE	730M	15B	5e-5	24.3	38.1	43.5	68.7	51.1	27.3	52.4	58.3	**45.5**
H-RoPE	730M	15B	3e-5	24.2	38.1	43.8	68.3	50.9	27.8	52.1	58.1	45.4
2D-RoPE	730M	15B	1e-4	26.6	38.9	42.0	67.8	47.8	26.5	52.0	53.3	44.3
2D-RoPE	730M	15B	5e-5	26.6	38.1	43.3	68.4	47.8	27.1	52.6	57.4	45.2
2D-RoPE	730M	15B	3e-5	26.7	37.9	43.4	68.7	48.0	26.5	52.3	57.7	45.1
RoPE	1.4B	28B	1e-4	23.0	18.1	31.5	51.6	31.5	24.7	52.1	61.8	36.8
RoPE	1.4B	28B	5e-5	23.4	42.7	46.7	69.5	51.9	28.8	54.0	50.7	45.9
RoPE	1.4B	28B	3e-5	24.1	42.4	46.9	70.2	52.7	29.2	52.9	49.8	46.0
H-RoPE	1.4B	28B	1e-4	24.6	42.8	45.3	68.2	50.6	27.0	53.7	61.9	46.8
H-RoPE	1.4B	28B	5e-5	25.3	44.5	47.1	69.3	51.3	28.0	53.9	57.9	**47.2**
H-RoPE	1.4B	28B	3e-5	25.3	44.7	47.0	69.3	52.0	27.0	54.2	58.3	**47.2**
2D-RoPE	1.4B	28B	1e-4	24.4	40.5	45.3	69.5	50.8	26.4	51.7	58.9	45.9
2D-RoPE	1.4B	28B	5e-5	23.7	41.5	46.8	70.4	51.8	26.7	55.2	58.6	46.8
2D-RoPE	1.4B	28B	3e-5	23.7	41.5	47.1	70.2	52.3	27.2	54.2	58.8	46.9
Overtrained Setting												
RoPE	730M	100B	1e-4	22.9	18.1	30.3	51.5	29.4	23.8	49.3	49.1	34.3
RoPE	730M	100B	5e-5	23.5	43.0	46.2	68.5	48.6	26.8	51.9	59.1	46.0
RoPE	730M	100B	3e-5	24.4	44.1	47.1	69.6	50.9	28.2	52.4	58.8	**46.9**
H-RoPE	730M	100B	1e-4	23.3	37.8	41.7	66.4	46.2	26.6	51.9	52.2	43.3
H-RoPE	730M	100B	5e-5	24.4	43.0	46.9	69.7	52.3	28.0	52.7	52.6	46.2
H-RoPE	730M	100B	3e-5	24.3	42.3	47.6	69.9	52.5	27.6	52.8	54.2	46.4
2D-RoPE	730M	100B	1e-4	23.8	40.0	41.8	66.1	44.4	22.7	52.5	56.2	43.4
2D-RoPE	730M	100B	5e-5	23.9	42.8	46.5	68.9	49.6	26.0	54.6	57.1	46.2
2D-RoPE	730M	100B	3e-5	24.3	42.4	47.0	69.0	50.1	25.5	54.1	57.8	46.3

For each token i, let its 2D position ID be

$$P_i = \big(p_i^{(1)}, p_i^{(2)}\big).$$

Intuitively, $p_i^{(1)}$ plays the role of a column-like coordinate, while $p_i^{(2)}$ plays the role of a row-like coordinate. Unlike rule-based 2D-RoPE, these coordinates are not computed from line breaks. Instead, at each layer, we generate two update values from the hidden representation of the current token.

Concretely, let $\boldsymbol{h}_i^{(\ell)}$ be the hidden representation of token i at layer ℓ. We apply a learnable linear projection to obtain

$$(a_i^{(\ell)}, b_i^{(\ell)}) = \boldsymbol{W}_{\text{pos}}^{(\ell)} \boldsymbol{h}_i^{(\ell)} + \boldsymbol{b}_{\text{pos}}^{(\ell)}.$$

We then pass these two scalars through a sigmoid function and a rescaling factor α:

$$u_i^{(\ell)} = \alpha \cdot S(a_i^{(\ell)}), \qquad v_i^{(\ell)} = \alpha \cdot S(b_i^{(\ell)}),$$

where $S(\cdot)$ denotes the sigmoid function. In our experiments, we set $\alpha = 1$ by default. This makes the update coefficients bounded and positive.

Given these updated values, the first coordinate is updated by an affine transformation

$$p_{i+1}^{(1)} = u_i^{(\ell)} p_i^{(1)} + v_i^{(\ell)}.$$

Equivalently,

$$\begin{bmatrix} p_{i+1}^{(1)} \\ 1 \end{bmatrix} = \begin{bmatrix} u_i^{(\ell)} & v_i^{(\ell)} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} p_i^{(1)} \\ 1 \end{bmatrix}.$$

The second coordinate is updated by the complementary increment

$$p_{i+1}^{(2)} = p_i^{(2)} + \big(1 - u_i^{(\ell)}\big).$$

Putting the two coordinates together, we can write the update as

$$\begin{bmatrix} p_{i+1}^{(1)} \\ p_{i+1}^{(2)} \\ 1 \end{bmatrix} = \begin{bmatrix} u_i^{(\ell)} & 0 & v_i^{(\ell)} \\ 0 & 1 & 1 - u_i^{(\ell)} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_i^{(1)} \\ p_i^{(2)} \\ 1 \end{bmatrix}.$$

Thus, the position ID of token i is determined by the cumulative product of these learned affine transformations along the sequence.

This update rule can be viewed as a learnable relaxation of the line-break rule used in 2D-RoPE. For example, if $u_i^{(\ell)} \approx 1$ and $v_i^{(\ell)} \approx 1$, then

$$p_{i+1}^{(1)} \approx p_i^{(1)} + 1, \qquad p_{i+1}^{(2)} \approx p_i^{(2)},$$

which resembles moving to the next column within the same row. On the other hand, if $u_i^{(\ell)} \approx 0$ and $v_i^{(\ell)} \approx 0$, then

$$p_{i+1}^{(1)} \approx 0, \qquad p_{i+1}^{(2)} \approx p_i^{(2)} + 1,$$

which resembles resetting the column coordinate and moving to a new row. Therefore, Auto 2D-RoPE can in principle learn behavior similar to rule-based 2D-RoPE, but it is not restricted to using the newline token as the only row separator.

After the 2D position IDs are computed, we apply the same 2D-RoPE rotation as in Section 3. Specifically, for each layer ℓ, the query and key vectors are rotated according to the learned coordinates

$$\big(p_i^{(1)}, p_i^{(2)}\big),$$

where one part of the head dimension encodes relative differences in the first coordinate, and the other part encodes relative differences in the second coordinate. The attention layer is then computed in the usual causal way:

$$\mathsf{Attn}(\boldsymbol{H}^{(\ell)}) = \mathcal{S}\left(\text{MASK}\left(R_{\text{auto}}(\boldsymbol{H}^{(\ell)}\boldsymbol{Q})\big(R_{\text{auto}}(\boldsymbol{H}^{(\ell)}\boldsymbol{K})\big)^\top\right)\right)\boldsymbol{H}^{(\ell)}\boldsymbol{V},$$

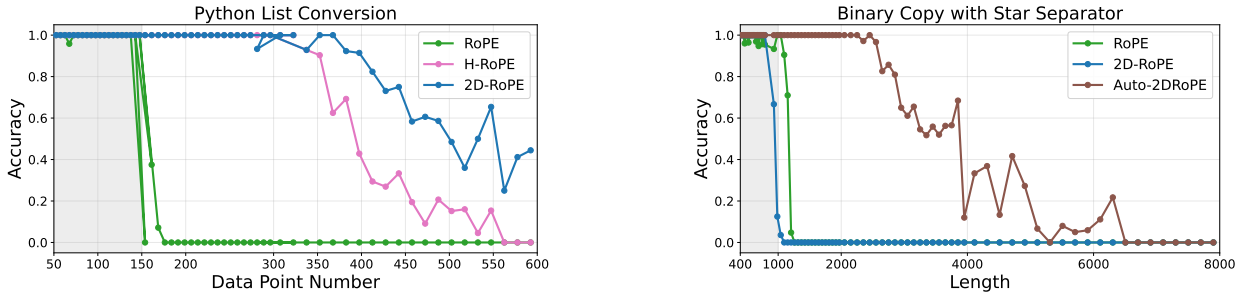


Figure 8. Fine-tuning results on copy-related tasks using 350M models pretrained on 7B DCLM tokens. The gray region indicates the training context length range during finetuning.

where R_{auto} denotes the 2D-RoPE rotation using the learned Auto 2D position IDs.

Importantly, the quantities $a_i^{(\ell)}$ and $b_i^{(\ell)}$ are not independent learnable parameters tied to absolute positions. They are generated by a shared linear map from the hidden representation of each token. Therefore, the number of parameters does not grow with the input length, and the coordinate-generation rule can be applied to longer sequences at test time. This is the main difference between Auto 2D-RoPE and simply learning a fixed position ID table.

Empirically, this learned coordinate update allows the model to recover useful 2D structure even when the input does not contain explicit newline separators. As discussed in the main text, when we replace the newline token <\n> with another character $*$, rule-based 2D-RoPE can no longer identify row boundaries and effectively loses its 2D structure. In contrast, Auto 2D-RoPE can still learn data-dependent position IDs and shows stronger length generalization on the copy task, as shown in Figure 8.

H. Preliminary for Theoretical Results

In this section, we formalize our general setting for our theory.

Theoretical Definition of Copying. First, we define our copying task used in theory. We always have a $0/1$ string s with length at least 1 and a copying position $1 \leq i \leq |s|$. We fix the positive prompt lengths (a, b). The input of the model is a string generated by s, i, denoted by $\mathsf{Str}(s, i) := (\mathtt{A}_1, \mathtt{A}_2, \cdots, \mathtt{A}_a, s, \mathtt{<\backslash n>}, \mathtt{B}_1, \mathtt{B}_2, \cdots, \mathtt{B}_b, \cdots, \mathtt{B}_b, \mathtt{<O:>}, s_{:i})$, where $\mathtt{A}_i$, $\mathtt{B}_j$ are the different tokens other than 0,1. Here $s_{:i}$ means the length $(i-1)$ prefix of s. For this input, the last token is s_{i-1} (and when $i = 1$, the last token is <O:>). The vocabulary set is then defined as $\mathcal{V} = \{\mathtt{A}_1, \mathtt{A}_2, \cdots, \mathtt{A}_a, \mathtt{B}_1, \mathtt{B}_2, \cdots, \mathtt{B}_b, \cdots, \mathtt{B}_b, 0, 1, \mathtt{<\backslash n>}, \mathtt{<O:>}\}$. We model the predictor as a function $P_{\boldsymbol{\theta}} : \mathcal{V}^+ \to [0,1]^2$, parameterized by $\boldsymbol{\theta}$, which maps an input string $x \in \mathcal{V}^+$ to a distribution over the next token in $\{0, 1\}$. Specifically, for any input x, the output $P_{\boldsymbol{\theta}}(x) = \left([P_{\boldsymbol{\theta}}(x)]_0, [P_{\boldsymbol{\theta}}(x)]_1\right)$ satisfies

$$[P_{\boldsymbol{\theta}}(x)]_0 + [P_{\boldsymbol{\theta}}(x)]_1 = 1, \quad \text{and} \quad 0 \leq [P_{\boldsymbol{\theta}}(x)]_i \leq 1 \text{ for } i \in \{0, 1\}.$$

Thus, $P_{\boldsymbol{\theta}}(x)$ defines a valid probability distribution over the next token. For each pair of s, i, the corresponding loss is defined as

$$\mathcal{L}_{\boldsymbol{\theta}}(s, i) = -\log \left[P_{\boldsymbol{\theta}}(\mathsf{Str}(s, i))\right]_{s_i}.$$

Now given a distribution $\mathcal{D}$, the total loss is defined as

$$\mathcal{L}(\boldsymbol{\theta}) = \mathbb{E}_{(s,i)\sim\mathcal{D}}[\mathcal{L}_{\boldsymbol{\theta}}(s, i)].$$

Definition H.1. We say a model can successfully do copying in length L if for any $|s| \leq L$ and $1 \leq i \leq |s|$, we have $\left[P_{\boldsymbol{\theta}}(\mathsf{Str}(s, i))\right]_{s_i} > \frac{1}{2}$.

Formalization of RoPE. Given a dimension d, RoPE attention is defined as a function $\mathsf{Attn} : \mathbb{R}^{T\times d} \to \mathbb{R}^{T\times d}$ for any T parameterized by $\boldsymbol{K}, \boldsymbol{Q}, \boldsymbol{V} \in \mathbb{R}^{d\times d}$. For any input $\boldsymbol{X}$, the RoPE attention is defined as

$$\mathsf{Attn}(\boldsymbol{X}) = \mathcal{S}\left(\text{MASK}\left(R\left(\boldsymbol{X}\boldsymbol{Q}\right)\left(R\left(\boldsymbol{X}\boldsymbol{K}\right)\right)^\top\right)\right)\boldsymbol{X}\boldsymbol{V}^\top \in \mathbb{R}^{T\times d}.$$

Here, $R : \mathbb{R}^{T\times d} \to \mathbb{R}^{T\times d}$ is the RoPE function. Specifically, for an input $\boldsymbol{X} \in \mathbb{R}^{T\times d}$, the i-th row X_i for $1 \le i \le L$ is mapped to $X_i R_i^\top$, where

$$R_i \;=\; \begin{bmatrix} \cos(i\beta_1) & -\sin(i\beta_1) & & & \\ \sin(i\beta_1) & \cos(i\beta_1) & & & \\ & & \cos(i\beta_2) & -\sin(i\beta_2) & \\ & & \sin(i\beta_2) & \cos(i\beta_2) & \\ & & & & \ddots \end{bmatrix} \in \mathbb{R}^{d\times d}.$$

Here, β_j is the base frequency associated with the $(2j-1, 2j)$ coordinate pair. In our setting, these β_j's are arbitrary hyperparameters. Thus, R_i acts block-diagonally on X_i, rotating each two-dimensional subspace $(X_{i,2j-1}, X_{i,2j})$ by angle $i\beta_j$. The causal mask operator $\mathrm{MASK}(\cdot)$ enforces the autoregressive constraint: for $\boldsymbol{Z} \in \mathbb{R}^{T\times T}$,

$$\mathrm{MASK}(\boldsymbol{Z})_{ij} \;=\; \begin{cases} \boldsymbol{Z}_{ij}, & j \le i, \\ -\infty, & j > i, \end{cases}$$

so that position i cannot attend to any future position $j > i$. The operator $\mathcal{S}(\cdot)$ applies row-wise softmax normalization: for a matrix $\boldsymbol{A} \in \mathbb{R}^{T\times T}$,

$$\mathcal{S}(\boldsymbol{A})_{ij} \;=\; \frac{\exp(\boldsymbol{A}_{ij})}{\sum_{k=1}^{L}\exp(\boldsymbol{A}_{ik})}, \quad 1 \le i, j \le T.$$

I. Shift Invariant Positional Encoding Cannot Solve Binary Copy

In this section, we prove that one-layer shift-invariant positional encoding cannot do copying as stated in Theorem 2.1. Remind that our general theoretical setting is in Appendix H.

I.1. Shift Invariant PE Formalization

First, we mathematically define what shift-invariant positional encoding is.

Similar to the RoPE definition in Appendix H, shift-invariant attention is also parameterized by $\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V} \in \mathbb{R}^d$. For each $\boldsymbol{X} \in \mathbb{R}^{T\times d}$, the attention first computes the key vectors and query vectors

$$\boldsymbol{k}_i = \boldsymbol{K}^\top \boldsymbol{X}_{i,:}^\top, \boldsymbol{q}_i = \boldsymbol{Q}^\top \boldsymbol{X}_{i,:}^\top.$$

Then, it computes a matrix $\boldsymbol{A} \in \mathbb{R}^{T\times T}$ such that

$$\boldsymbol{A}_{i,j} = \begin{cases} -\infty & i < j \\ f(i-j, \boldsymbol{q}_i, \boldsymbol{k}_j) & i \ge j \end{cases}$$

for some function $f : \mathbb{R} \times \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$. Then, the output of the attention is

$$\mathsf{Attn}(\boldsymbol{X}) = \mathcal{S}(\boldsymbol{A})\boldsymbol{X}\boldsymbol{V}^\top.$$

When we treat this attention as a model for binary copy (as defined in Appendix H), we first have a token embedding $\boldsymbol{w}_{\mathtt{C}} \in \mathbb{R}^d$ for each $\mathtt{C} \in \mathcal{V}$. For any input s, i, we convert the string $\mathsf{Str}(s,i)$ into the input to the attention as $\boldsymbol{X}_{j,:} = \boldsymbol{w}_{\mathsf{Str}(s,i)_j}^\top$ (denote this matrix as $\boldsymbol{X}(s,i) \in \mathbb{R}^{(|s|+i-1+a+b)\times d}$, where $(|s|+i-1+a+b)$ is the length of $\mathsf{Str}(s,i)$). We also add another matrix parameter $\boldsymbol{W}_O \in \mathbb{R}^{d\times 2}$, and the model is defined as a map from (s,i) to a 2-dimensional vector:

$$P_{\boldsymbol{\theta}}(s,i) = \mathrm{softmax}\left((\mathsf{Attn}(\boldsymbol{X}(s,i))\boldsymbol{W}_O)_{|s|+i-1+a+b,:}\right). \tag{1}$$

I.2. Failure of Shift Invariant PE for Binary Copy

In this subsection, we organize and provide proof for our arguments in Theorem 2.1. First, to make our results more organized and easy to understand, we give a more detailed statement of Theorem 2.1.

Theorem I.1 (Detailed Statement of Theorem 2.1)**.** *For binary copy with vocabulary set* $\mathcal{V} = \{\mathtt{A}_1, \mathtt{A}_2, \cdots, \mathtt{A}_a, \mathtt{B}_1, \mathtt{B}_2, \cdots, \mathtt{B}_b, 0, 1, \mathtt{<\backslash n>}, \mathtt{<O:>}\}$*, no one-layer Transformer with shift-invariant positional encodings in Equation* (2) *can successfully copy all input strings of length* $n \leq 5$*. The definition of successful copy is following Definition H.1.*

Notations. We first recall some notations used in Appendix H and introduce some other necessary notations. We denote the input prefix string $(\mathtt{A}_1, \mathtt{A}_2, \cdots, \mathtt{A}_a, s, \mathtt{<\backslash n>}, \mathtt{B}_1, \mathtt{B}_2, \cdots, \mathtt{B}_b, \cdots, \mathtt{B}_b, s_{:i})$ as $\mathsf{Str}(s, i)$. We consider the last token of $\mathsf{Str}(s, i)$ and denote it by $C(s, i)$. For each word $\mathtt{C} \in \mathcal{V}$, notice that $\boldsymbol{w}_{\mathtt{C}}\boldsymbol{V}^\top\boldsymbol{W}_O = (x_1, x_2)^\top$ is a 2-dimensional vector, we denote $\Delta_{\mathtt{C}} := x_1 - x_2$ as the first coordinate minus the second coordinate of it. Moreover, we define $\boldsymbol{q}_{\mathtt{C}} := \boldsymbol{w}_{\mathtt{C}}\boldsymbol{Q}^\top \in \mathbb{R}^{1\times d}$, $\boldsymbol{k}_{\mathtt{C}} := \boldsymbol{w}_{\mathtt{C}}\boldsymbol{K}^\top \in \mathbb{R}^{1\times d}$. To avoid confusion, when we refer to $\boldsymbol{q}_{\mathtt{C}}$ and $\boldsymbol{k}_{\mathtt{C}}$ for $\mathtt{C} = 0, 1$, we write $\boldsymbol{q}_{(0)}$ and $\boldsymbol{q}_{(1)}$, whose subscripts are not 0, 1 but (0), (1) to indicate that the index is a token rather than a position.

For any input s, i, we convert the string $\mathsf{Str}(s, i)$ into the input to the attention as $\boldsymbol{X}_{j,:} = \boldsymbol{w}_{\mathsf{Str}(s,i)_j}^\top$. We denote this matrix as $\boldsymbol{X}(s, i) \in \mathbb{R}^{(|s|+i-1+a+b)\times d}$, where $(|s| + i - 1 + a + b)$ is the length of $\mathsf{Str}(s, i)$. We also add another matrix parameter $\boldsymbol{W}_O \in \mathbb{R}^{d\times 2}$, and the model is defined as a map from (s, i) to a 2-dimensional vector:

$$P_{\boldsymbol{\theta}}(s, i) = \mathrm{softmax}\left((\mathsf{Attn}(\boldsymbol{X}(s, i))\boldsymbol{W}_O)_{(|s|+i-1+a+b),:}\right). \tag{2}$$

Therefore, a model can successfully do binary copy at the i-th index as $[P_{\boldsymbol{\theta}}(\mathsf{Str}(s, i))]_{s_i} > \frac{1}{2}$ has an equivalent criterion as the following lemma states.

Lemma I.2. *The inequality*

$$[P_{\boldsymbol{\theta}}(\mathsf{Str}(s, i))]_{s_i} > \frac{1}{2}$$

is equivalent to that:

$$(-1)^{s_i} \sum_{j=1}^{|s|+i-1+a+b} \exp(\boldsymbol{A}_{|s|+i-1+a+b,j})\Delta_{\mathsf{Str}(s,i)_j} > 0, \tag{3}$$

where $\boldsymbol{A}$ *is the attention matrix for input* $\boldsymbol{X}(s, i)$ *and* $s_i \in \{0, 1\}$*.*

Proof of Lemma I.2. Let

$$T := |\mathsf{Str}(s, i)| = |s| + i - 1 + a + b, \qquad x := \mathsf{Str}(s, i).$$

Write the two logits at the last position as

$$\boldsymbol{z} = (z_0, z_1) := (\mathsf{Attn}(\boldsymbol{X}(s, i))\boldsymbol{W}_O)_{T,:}.$$

By the definition of the model in Equation (2),

$$P_{\boldsymbol{\theta}}(\mathsf{Str}(s, i)) = \mathrm{softmax}(\boldsymbol{z}).$$

Since the output space only contains the two binary tokens 0 and 1, for any $b \in \{0, 1\}$ we have

$$[\mathrm{softmax}(\boldsymbol{z})]_b = \frac{\exp(z_b)}{\exp(z_0) + \exp(z_1)}.$$

Therefore, if $s_i = 0$, then

$$\begin{aligned} [P_{\boldsymbol{\theta}}(\mathsf{Str}(s, i))]_0 > \frac{1}{2} \quad &\Longleftrightarrow \quad \frac{\exp(z_0)}{\exp(z_0) + \exp(z_1)} > \frac{1}{2} \\ &\Longleftrightarrow \quad \exp(z_0) > \exp(z_1) \\ &\Longleftrightarrow \quad z_0 - z_1 > 0. \end{aligned}$$

Similarly, if $s_i = 1$, then

$$[P_{\boldsymbol{\theta}}(\mathsf{Str}(s, i))]_1 > \frac{1}{2} \quad \Longleftrightarrow \quad z_1 - z_0 > 0 \quad \Longleftrightarrow \quad z_0 - z_1 < 0.$$

Combining the two cases, and using $s_i \in \{0, 1\}$, we obtain the equivalent condition

$$(-1)^{s_i}(z_0 - z_1) > 0.$$

It remains to rewrite $z_0 - z_1$ in terms of the attention scores. By the definition of the attention layer,

$$\mathsf{Attn}(\boldsymbol{X}(s,i))_{T,:} = \sum_{j=1}^{T} \mathcal{S}(\boldsymbol{A})_{T,j}\, \boldsymbol{w}_{x_j} \boldsymbol{V}^\top,$$

where $x_j = \mathsf{Str}(s,i)_j$. Since the last position T can causally attend to all positions $1, \ldots, T$, the softmax-normalized attention weight is

$$\mathcal{S}(\boldsymbol{A})_{T,j} = \frac{\exp(\boldsymbol{A}_{T,j})}{\sum_{\ell=1}^{T} \exp(\boldsymbol{A}_{T,\ell})}.$$

Moreover, under the shift-invariant attention model, each attention score in this last row has the form

$$\boldsymbol{A}_{T,j} = f(T - j, \boldsymbol{q}_{x_T}, \boldsymbol{k}_{x_j}) = f(T - j, \boldsymbol{q}_{C(s,i)}, \boldsymbol{k}_{\mathsf{Str}(s,i)_j}),$$

where $C(s,i) = x_T$ is the last token of $\mathsf{Str}(s,i)$.

Now multiplying the last hidden vector by $\boldsymbol{W}_O$, we get

$$\boldsymbol{z} = \sum_{j=1}^{T} \mathcal{S}(\boldsymbol{A})_{T,j}\, \boldsymbol{w}_{x_j} \boldsymbol{V}^\top \boldsymbol{W}_O.$$

For each token $\mathtt{C} \in \mathcal{V}$, recall that

$$\Delta_{\mathtt{C}} = \big(\boldsymbol{w}_{\mathtt{C}} \boldsymbol{V}^\top \boldsymbol{W}_O\big)_0 - \big(\boldsymbol{w}_{\mathtt{C}} \boldsymbol{V}^\top \boldsymbol{W}_O\big)_1.$$

Thus,

$$z_0 - z_1 = \sum_{j=1}^{T} \mathcal{S}(\boldsymbol{A})_{T,j} \Delta_{x_j} = \frac{\sum_{j=1}^{T} \exp(\boldsymbol{A}_{T,j}) \Delta_{x_j}}{\sum_{\ell=1}^{T} \exp(\boldsymbol{A}_{T,\ell})}.$$

The denominator is strictly positive, so it does not affect the sign. Hence

$$(-1)^{s_i}(z_0 - z_1) > 0 \quad \Longleftrightarrow \quad (-1)^{s_i} \sum_{j=1}^{T} \exp(\boldsymbol{A}_{T,j}) \Delta_{x_j} > 0.$$

Substituting $T = |s| + i - 1 + a + b$ and $x_j = \mathsf{Str}(s,i)_j$ gives exactly

$$(-1)^{s_i} \sum_{j=1}^{|s|+i-1+a+b} \exp(\boldsymbol{A}_{|s|+i-1+a+b,j}) \Delta_{\mathsf{Str}(s,i)_j} > 0.$$

This proves the equivalence. □

With the above lemma, now we are ready to give the proof of Theorem I.1 by contradiction.

Proof of Theorem I.1. Now, assume that the model can successfully solve binary copy with any string of length at most 5. We prove it cannot be true by contradiction. We consider the following four specific pairs of (s, i):

- $s_1 = (00010), i_1 = 2$
- $s_2 = (01000), i_2 = 2$
- $s_3 = (100), i_3 = 3$
- $s_4 = (001), i_4 = 3.$

We first consider

$$u := \mathsf{Str}(s_1, i_1) = (\mathtt{A}_1, \mathtt{A}_2, \cdots, \mathtt{A}_a, 0, 0, 0, 1, 0, \mathtt{B}_1, \mathtt{B}_2, \cdots, \mathtt{B}_b, 0),$$
$$v := \mathsf{Str}(s_2, i_2) = (\mathtt{A}_1, \mathtt{A}_2, \cdots, \mathtt{A}_a, 0, 1, 0, 0, 0, \mathtt{B}_1, \mathtt{B}_2, \cdots, \mathtt{B}_b, 0).$$

By our assumption that the model can successfully solve binary copy with any string of length at most 5 and Lemma I.2, Equation (3) holds for u and v. We further rewrite Equation (3) for these two strings as

$$\sum_{\substack{1\le j\le 6+a+b\\ j\ne a+2,a+4}} \exp(f(6+a+b-j, \boldsymbol{q}_{(0)}, \boldsymbol{k}_{u_j}))\Delta_{u_j} + \exp(f(4+b, \boldsymbol{q}_{(0)}, \boldsymbol{k}_{(0)}))\Delta_{(0)} + \exp(f(2+b, \boldsymbol{q}_{(0)}, \boldsymbol{k}_{(1)}))\Delta_{(1)} > 0$$
$$\sum_{\substack{1\le j\le 6+a+b\\ j\ne a+2,a+4}} \exp(f(6+a+b-j, \boldsymbol{q}_{(0)}, \boldsymbol{k}_{v_j}))\Delta_{v_j} + \exp(f(4+b, \boldsymbol{q}_{(0)}, \boldsymbol{k}_{(1)}))\Delta_{(1)} + \exp(f(2+b, \boldsymbol{q}_{(0)}, \boldsymbol{k}_{(0)}))\Delta_{(0)} < 0.$$

The above inequalities use the property of our shift-invariant model that

$$\boldsymbol{A}_{|s|+i-1+a+b,j} = f(|s|+i-1+a+b-j, \boldsymbol{q}_{|s|+i-1+a+b}, \boldsymbol{k}_j).$$

By comparing the above two formulas, and noticing that $u_j = v_j$ for $j \ne 4, 6$, we have

$$\begin{aligned} &\exp(f(4+b, \boldsymbol{q}_{(0)}, \boldsymbol{k}_{(0)}))\Delta_{(0)} + \exp(f(2+b, \boldsymbol{q}_{(0)}, \boldsymbol{k}_{(1)}))\Delta_{(1)} \\ >&\exp(f(4+b, \boldsymbol{q}_{(0)}, \boldsymbol{k}_{(1)}))\Delta_{(1)} + \exp(f(2+b, \boldsymbol{q}_{(0)}, \boldsymbol{k}_{(0)}))\Delta_{(0)}. \end{aligned} \tag{4}$$

Now, we similarly compare Equation (3), and denote

$$u' := \mathsf{Str}(s_3, i_3) = (\mathtt{A}_1, \mathtt{A}_2, \cdots, \mathtt{A}_a, 1, 0, 0, \mathtt{B}_1, \mathtt{B}_2, \cdots, \mathtt{B}_b, 1, 0),$$
$$v' := \mathsf{Str}(s_4, i_4) = (\mathtt{A}_1, \mathtt{A}_2, \cdots, \mathtt{A}_a, 0, 0, 1, \mathtt{B}_1, \mathtt{B}_2, \cdots, \mathtt{B}_b, 0, 0).$$

We have

$$\sum_{\substack{1\le j\le 5+a+b\\ j\ne a+1,a+3}} \exp(f(5+a+b-j, \boldsymbol{q}_{(0)}, \boldsymbol{k}_{u'_j}))\Delta_{u'_j} + \exp(f(4+b, \boldsymbol{q}_{(0)}, \boldsymbol{k}_{(1)}))\Delta_{(1)} + \exp(f(2+b, \boldsymbol{q}_{(0)}, \boldsymbol{k}_{(0)}))\Delta_{(0)} > 0$$
$$\sum_{\substack{1\le j\le 5+a+b\\ j\ne a+1,a+3}} \exp(f(5+a+b-j, \boldsymbol{q}_{(0)}, \boldsymbol{k}_{v'_j}))\Delta_{v'_j} + \exp(f(4+b, \boldsymbol{q}_{(0)}, \boldsymbol{k}_{(0)}))\Delta_{(0)} + \exp(f(4+b, \boldsymbol{q}_{(0)}, \boldsymbol{k}_{(1)}))\Delta_{(1)} < 0.$$

Therefore,

$$\begin{aligned} &\exp(f(b+4, \boldsymbol{q}_{(0)}, \boldsymbol{k}_{(0)}))\Delta_{(0)} + \exp(f(b+2, \boldsymbol{q}_{(0)}, \boldsymbol{k}_{(1)}))\Delta_{(1)} \\ <&\exp(f(b+4, \boldsymbol{q}_{(0)}, \boldsymbol{k}_{(1)}))\Delta_{(1)} + \exp(f(b+2, \boldsymbol{q}_{(0)}, \boldsymbol{k}_{(0)}))\Delta_{(0)}, \end{aligned}$$

which contradicts to Equation (4). Thus, we proved Theorem I.1. □

J. 2-Layer RoPE Transformer can Represent Binary Copy

In this section, we show the existence of a Transformer that can exactly perform binary copy for sequence lengths significantly larger than the norm constraint.

J.1. Prompt Template

We recall the binary copy task in Appendix H defined over the vocabulary set

$$\mathcal{V} = \{\mathtt{A}_1, \mathtt{A}_2, \cdots, \mathtt{A}_a, \mathtt{B}_1, \mathtt{B}_2, \cdots, \mathtt{B}_b, 0, 1, \texttt{<\textbackslash n>}, \texttt{<O:>}\}.$$

The full copy task is specified by a binary string $s \in \{0,1\}^*$. In the next-token prediction formulation, copying s is decomposed into predicting each copied token one by one. More precisely, when we analyze the prediction of the i-th

copied token, where $1 \leq i \leq |s|$, the model has only seen the source string s and the already generated prefix of the output string. We denote this prefix by $s_{:i}$, which is the length $(i-1)$ prefix of s. Thus, the input sequence seen by the model at this prediction step is

$$\mathsf{Str}(s,i) := (\texttt{A}_1, \texttt{A}_2, \cdots, \texttt{A}_a,\ s,\ \texttt{<\textbackslash n>},\ \texttt{B}_1, \texttt{B}_2, \cdots, \texttt{B}_b,\ \texttt{<O:>},\ s_{:i}).$$

The target token for this input is s_i. Therefore, when $i = 1$, the output prefix $s_{:i}$ is empty and the last token in $\mathsf{Str}(s,i)$ is $\texttt{A}_a$; when $i > 1$, the last token is the previously copied token s_{i-1}.

Equivalently, for any strict prefix s' of s, the model input has the form

$$(\texttt{A}_1, \texttt{A}_2, \cdots, \texttt{A}_a,\ s,\ \texttt{B}_1, \texttt{B}_2, \cdots, \texttt{B}_b, s'),$$

and the objective is to predict the next token of s after the prefix s'. In the notation above, $s' = s_{:i}$ and the target output is s_i.

J.2. Transformer Architecture

We consider the following simplified transformer architecture.

Let the RoPE dimension $d > 50$ be an even number. The embeddings $\boldsymbol{w}_0, \boldsymbol{w}_1, \boldsymbol{w}_{\texttt{A1}}, \boldsymbol{w}_{\texttt{A2}}, \boldsymbol{w}_{\texttt{B1}}, \boldsymbol{w}_{\texttt{B2}}, \boldsymbol{w}_{\texttt{<O:>}}, \boldsymbol{w}_{\texttt{<\textbackslash n>}} \in \mathbb{R}^d$ are unit vectors and orthogonal to each other. The input is converted to $\boldsymbol{X}^{(0)} \in \mathbb{R}^{T \times d}$, where T is the total input length and d is the embedding size. Then we have one RoPE attention layer

$$\boldsymbol{H}^{(1)} \ = \ \mathsf{Attn}_1\big(\boldsymbol{X}^{(0)}\big) = \mathcal{S}\left(\mathrm{MASK}\left(R\left(\boldsymbol{X}^{(0)}\boldsymbol{Q}^{(0)}\right)\left(R\left(\boldsymbol{X}^{(0)}\boldsymbol{K}^{(0)}\right)\right)^\top\right)\right)\boldsymbol{X}^{(0)}(\boldsymbol{V}^{(0)})^\top,$$

where $\boldsymbol{H}^{(1)} \in \mathbb{R}^{T \times d}$. Here, $\boldsymbol{K}^{(0)}, \boldsymbol{Q}^{(0)}, \boldsymbol{V}^{(0)} \in \mathbb{R}^{d \times d}$, and $R : \mathbb{R}^{T \times d} \to \mathbb{R}^{T \times d}$ is the RoPE function. Specifically, for an input $\boldsymbol{X} \in \mathbb{R}^{T \times d}$, the i-th row X_i for $1 \leq i \leq T$ is mapped to $X_i R_i^\top$, where

$$R_i \ = \ \begin{bmatrix} \cos(i\beta_1) & -\sin(i\beta_1) & & & \\ \sin(i\beta_1) & \cos(i\beta_1) & & & \\ & & \cos(i\beta_2) & -\sin(i\beta_2) & \\ & & \sin(i\beta_2) & \cos(i\beta_2) & \\ & & & & \ddots \end{bmatrix} \ \in \ \mathbb{R}^{d \times d}.$$

Here, β_j is the base frequency associated with the $(2j-1, 2j)$ coordinate pair. In our setting, these β_j's are arbitrary hyperparameters. Thus, R_i acts block-diagonally on X_i, rotating each two-dimensional subspace $(X_{i,2j-1}, X_{i,2j})$ by angle $i\beta_j$. The causal mask operator $\mathrm{MASK}(\cdot)$ enforces the autoregressive constraint: for $\boldsymbol{Z} \in \mathbb{R}^{T \times T}$,

$$\mathrm{MASK}(\boldsymbol{Z})_{ij} \ = \ \begin{cases} \boldsymbol{Z}_{ij}, & j \leq i, \\ -\infty, & j > i, \end{cases}$$

so that position i cannot attend to any future position $j > i$. The operator $\mathcal{S}(\cdot)$ applies row-wise softmax normalization: for a matrix $\boldsymbol{A} \in \mathbb{R}^{T \times T}$,

$$\mathcal{S}(\boldsymbol{A})_{ij} \ = \ \frac{\exp(\boldsymbol{A}_{ij})}{\sum_{k=1}^{L} \exp(\boldsymbol{A}_{ik})}, \quad 1 \leq i, j \leq T.$$

Then, the attention layer output come through a normalization layer (here we don't consider MLP layer for simplicity), and then added to $\boldsymbol{X}^{(0)}$ by a residual link. Formally, we have

$$\bar{\boldsymbol{H}}^{(1)} = \mathrm{RMSNorm}(\boldsymbol{H}^{(1)})$$
$$\boldsymbol{X}^{(1)} = \boldsymbol{X}^{(0)} + \bar{\boldsymbol{H}}^{(1)}$$

where

$$\mathrm{RMSNorm}([\boldsymbol{h}_1, \ldots, \boldsymbol{h}_L]^\top) = \left[\frac{\lambda\, \boldsymbol{h}_1}{\|\boldsymbol{h}_1\|_2}, \ldots, \frac{\lambda\, \boldsymbol{h}_T}{\|\boldsymbol{h}_T\|_2}\right]^\top.$$

The second attention layer doesn't use positional encoding. Formally,

$$\boldsymbol{H}^{(2)} \;=\; \mathsf{Attn}_2\big(\boldsymbol{X}^{(1)}\big) = \mathcal{S}\left(\mathrm{MASK}\left((\boldsymbol{X}^{(1)}(\boldsymbol{K}^{(1)})^\top)\left((\boldsymbol{X}^{(1)}(\boldsymbol{Q}^{(1)})^\top)\right)^\top\right)\right)\boldsymbol{X}^{(1)}(\boldsymbol{V}^{(1)})^\top \;\in\; \mathbb{R}^{T\times d}.$$

Finally, the hidden representation $\boldsymbol{H}^{(2)} \in \mathbb{R}^{T\times d}$ is projected to the vocabulary space via a linear map $\boldsymbol{W}_O \in \mathbb{R}^{d\times|\mathcal{V}|}$:

$$\boldsymbol{Y} \;=\; \boldsymbol{H}^{(2)}\boldsymbol{W}_O \;\in\; \mathbb{R}^{T\times|\mathcal{V}|},$$

where $\boldsymbol{Y}$ contains the output logits.

Let $\mathcal{V}^+ = \mathcal{V}\cup\mathcal{V}^2\cup\mathcal{V}^3\cup\cdots$ denote the set of all nonempty finite sequences over $\mathcal{V}$. We denote the Transformer architecture as a function

$$F_{\boldsymbol{\theta}}^{(d,\boldsymbol{\beta})} : \mathcal{V}^+ \to \mathbb{R}^{L\times|\mathcal{V}|}, \quad F_{\boldsymbol{\theta}}^{(d,\boldsymbol{\beta})}(x) = \boldsymbol{Y},$$

where $x \in \mathcal{V}^+$ is the input sequence, $\boldsymbol{\theta}$ are the learnable parameters, d is the embedding dimension, and $\boldsymbol{\beta}$ denotes the hyperparameters $\beta_i, 1 \le i \le d/2$.

The prediction distribution is obtained by applying the row-wise softmax:

$$P_{\boldsymbol{\theta}}^{(d,\boldsymbol{\beta})}(x) = \mathrm{softmax}\Big(F_{\boldsymbol{\theta}}^{(d,\boldsymbol{\beta})}(x)\Big),$$

$$\Big[P_{\boldsymbol{\theta}}^{(d,\boldsymbol{\beta})}(x)\Big]_{i,j} = \frac{\exp\Big(F_{\boldsymbol{\theta}}^{(d,\boldsymbol{\beta})}(x)_{i,j}\Big)}{\sum_{k=1}^{|\mathcal{V}|}\exp\Big(F_{\boldsymbol{\theta}}^{(d,\boldsymbol{\beta})}(x)_{i,k}\Big)},$$

where $x \in \mathcal{V}^+$ is the input sequence, i indexes positions, and j indexes vocabulary symbols.

J.3. Proof of Theorem 2.2: Existence of a Length-Generalizing Solution

In this section, we show the existence of a transformer architecture that can solve binary copy for length larger than some polynomial of the parameter norm as stated in Theorem 2.2. We first give a detailed statement of Theorem 2.2, and we prove it directly.

Theorem J.1 (Detailed Statement of Theorem 2.2)**.** *Let $a \ge 2, b \ge 2$ be constant. There exists a constant embedding dimension $d \ge 10a + 10b$ such that for any norm constraint $\rho > 10000$, there exists RoPE freqencies $\boldsymbol{\beta}$ and a set of parameters $\boldsymbol{\theta} \in \{\|\boldsymbol{\theta}\|_2 \le \rho\}$ such that Transformer with embedding dimension d, RoPE freq $\boldsymbol{\beta}$, parameters $\boldsymbol{\theta}$ can perfectly perform copying for all lengths $1 \le L \le L_0 := \rho^2/(1000\log\rho)$. More precisely, for every such s and every $i \in [|s|]$, we have*

$$\arg\max_{j\in\mathcal{V}}\Big[P_{\boldsymbol{\theta}}^{(d,\boldsymbol{\beta})}(\mathtt{A}_1, \mathtt{A}_2, \cdots, \mathtt{A}_a,\ s,\ \mathtt{<\backslash n>},\ \mathtt{B}_1, \mathtt{B}_2, \cdots, \mathtt{B}_b,\ \mathtt{<O:>},\ s)\Big]_{|s|+a+b+i-1,j} = s_i.$$

Proof of Theorem J.1. We fix $d = 10a + 10b + 100$. Let $L_0 = \frac{\rho^2}{1000\log\rho}, \alpha = \frac{\rho}{10}$. We prove the existence of the parameters that can perfectly do copy for every binary string s with length $1 \le n := |s| \le L_0$.

We choose the RoPE frequencies as follows. Let $\beta_1 = 0$ and $\beta_2 = \frac{\pi}{4L_0}$. The remaining frequencies can be chosen arbitrarily. The proof below only uses β_2.

As there are only $(a + b + 2)$ kinds of token, there exists $\boldsymbol{K}^{(0)}, \boldsymbol{Q}^{(0)}$ such that:

$$\boldsymbol{K}^{(0)}\boldsymbol{w}_{\mathtt{A}_{a-1}} = \alpha\,\mathbf{e}_1, \boldsymbol{K}^{(0)}\boldsymbol{w}_{\mathtt{A}_a} = \alpha\mathbf{e}_1 + \mathbf{e}_4, \boldsymbol{K}^{(0)}\boldsymbol{w}_{\mathtt{B}_{b-1}} = 2\alpha\,\mathbf{e}_1, \boldsymbol{K}^{(0)}\boldsymbol{w}_{\mathtt{B}_b} = 2\alpha\,\mathbf{e}_1 + \mathbf{e}_4,$$
$$\boldsymbol{K}^{(0)}\boldsymbol{w}_{\mathtt{C}} = \mathbf{0} \text{ for any other token } \mathtt{C},$$
$$\boldsymbol{Q}^{(0)}\boldsymbol{w}_0 = \boldsymbol{Q}^{(0)}\boldsymbol{w}_1 = \boldsymbol{Q}^{(0)}\boldsymbol{w}_{\mathtt{B}_b} = \mathbf{e}_1 + \mathbf{e}_3,$$
$$\boldsymbol{Q}^{(0)}\boldsymbol{w}_{\mathtt{C}} = \mathbf{0} \text{ for any other token } \mathtt{C}.$$

Let $\boldsymbol{v}_{\mathtt{A}_{a-1}}, \boldsymbol{v}_{\mathtt{A}_a}, \boldsymbol{v}_{\mathtt{B}_{b-1}}, \boldsymbol{v}_{\mathtt{B}_b}$ be unit vectors that are orthogonal to $\boldsymbol{w}_{\mathtt{C}}$ for any token $\mathtt{C}$. Let

$$\boldsymbol{V}^{(0)}\boldsymbol{w}_{\mathtt{C}} = \boldsymbol{v}_{\mathtt{C}}, \mathtt{C} = \mathtt{A}_{a-1}, \mathtt{A}_a, \mathtt{B}_{b-1}, \mathtt{B}_b$$
$$\boldsymbol{V}^{(0)}\boldsymbol{w}_{\mathtt{C}} = \mathbf{0} \text{ for any other token } \mathtt{C}.$$

Since the token embeddings are orthonormal and the above linear maps are set to zero on the orthogonal complement of their span, we have

$$\|\boldsymbol{K}^{(0)}\|_F^2 = \alpha^2 + (\alpha^2+1) + 4\alpha^2 + (4\alpha^2+1) = 10\alpha^2 + 2,$$
$$\|\boldsymbol{Q}^{(0)}\|_F^2 = 4\|\mathbf{e}_1 + \mathbf{e}_3\|_2^2 = 8,$$
$$\|\boldsymbol{V}^{(0)}\|_F^2 = 4.$$

Therefore,

$$\|\boldsymbol{K}^{(0)}\|_F^2 + \|\boldsymbol{Q}^{(0)}\|_F^2 + \|\boldsymbol{V}^{(0)}\|_F^2 = 10\alpha^2 + 14.$$

Let s_0' be $\mathtt{B}_b$. In this construction, for $1 \le i \le n$, we have

$$\boldsymbol{w}_{\mathtt{A}_{a-1}}^\top \left(\boldsymbol{K}^{(0)}\right)^\top R_{i+1}\boldsymbol{Q}^{(0)}\boldsymbol{w}_{s_i} = \alpha,$$
$$\boldsymbol{w}_{\mathtt{A}_a}^\top \left(\boldsymbol{K}^{(0)}\right)^\top R_{i}\boldsymbol{Q}^{(0)}\boldsymbol{w}_{s_i} = \alpha + \sin(i\beta_2),$$
$$\boldsymbol{w}_{\mathtt{B}_b}^\top \left(\boldsymbol{K}^{(0)}\right)^\top R_{i+1}\boldsymbol{Q}^{(0)}\boldsymbol{w}_{s'_{i-1}} = 2\alpha,$$
$$\boldsymbol{w}_{\mathtt{B}_{b-1}}^\top \left(\boldsymbol{K}^{(0)}\right)^\top R_{i}\boldsymbol{Q}^{(0)}\boldsymbol{w}_{s'_{i-1}} = 2\alpha + \sin(i\beta_2).$$

We define

$$\boldsymbol{h}_i = \frac{1}{\sqrt{1+\exp^2(\sin(i\beta_2))}}\boldsymbol{v}_{\mathtt{A}_{a-1}} + \frac{\exp(\sin(i\beta_2))}{\sqrt{1+\exp^2(\sin(i\beta_2))}}\boldsymbol{v}_{\mathtt{A}_a}, 1 \le i \le n$$
$$\tilde{\boldsymbol{h}}_i = \frac{1}{\sqrt{1+\exp^2(\sin(i\beta_2))}}\boldsymbol{v}_{\mathtt{B}_b} + \frac{\exp(\sin(i\beta_2))}{\sqrt{1+\exp^2(\sin(i\beta_2))}}\boldsymbol{v}_{\mathtt{B}_{b-1}}, 1 \le i \le n.$$

As $\boldsymbol{V}^{(0)}\boldsymbol{w}_{\mathtt{0},1} = \mathbf{0}$, we have

$$\bar{\boldsymbol{H}}_{i+a}^{(1)} = \lambda \boldsymbol{h}_i, 1 \le i \le n.$$

Now we consider $\bar{\boldsymbol{H}}_{n+a+b+i-1}^{(1)}$. Notice that

$$\begin{aligned}\frac{1}{\lambda}\bar{\boldsymbol{H}}_{n+a+b+i-1}^{(1)} =& \frac{\exp(-\alpha)}{\sqrt{1+\exp^2(\sin(i\beta_2)) + \exp^2(-\alpha) + \exp^2(\sin((i+3+n)\beta_2)-\alpha))}}\boldsymbol{v}_{\mathtt{A}_{a-1}} \\ &+ \frac{\exp(\sin((i+3+n)\beta_2)-\alpha)}{\sqrt{1+\exp^2(\sin(i\beta_2)) + \exp^2(-\alpha) + \exp^2(\sin((i+3+n)\beta_2)-\alpha))}}\boldsymbol{v}_{\mathtt{A}_a} \\ &+ \frac{1}{\sqrt{1+\exp^2(\sin(i\beta_2)) + \exp^2(-\alpha) + \exp^2(\sin((i+3+n)\beta_2)-\alpha))}}\boldsymbol{v}_{\mathtt{B}_b} \\ &+ \frac{\exp(\sin(i\beta_2))}{\sqrt{1+\exp^2(\sin(i\beta_2)) + \exp^2(-\alpha) + \exp^2(\sin((i+3+n)\beta_2)-\alpha))}}\boldsymbol{v}_{\mathtt{B}_{b-1}}.\end{aligned}$$

Therefore,

$$\|\bar{\boldsymbol{H}}_{n+5+i}^{(1)} - \lambda\tilde{\boldsymbol{h}}_i\|_2^2 \le \lambda^2\exp(-2\alpha)(1+e^2+2e^2(e^2+1)^2) \le 10000\lambda^2\exp(-2\alpha).$$

We need to prove the following lemma:

Lemma J.2. *Let $\boldsymbol{h}_0 = \frac{1}{\sqrt{2}}\boldsymbol{v}_{\mathtt{A}1} + \frac{1}{\sqrt{2}}\boldsymbol{v}_{\mathtt{A}2}, \boldsymbol{h}_{-1} = \boldsymbol{v}_{\mathtt{A}1}$. There exists $\boldsymbol{\beta}$ such that for any $n < L_0$ and $-1 \le i < j \le n+1$, we have $\langle \boldsymbol{h}_i, \boldsymbol{h}_j\rangle \le 1 - \frac{1}{72L_0^2}$.*

Proof. Let $\beta_2 = \frac{\pi}{4L_0}$. For $0 \le i \le n+1$, we let $a_i = \exp(\sin(i\beta))$. Therefore, for any $0 \le i < j \le n+1$, as $n < L_0$,

$$\sin(j\beta) - \sin(i\beta) = \int_{i\beta}^{j\beta}\cos(x)dx \ge \frac{\beta}{\sqrt{2}}(j-i) \ge \frac{1}{2L_0}.$$

Therefore, for any $0 \le i < j \le n+1$,

$$a_j - a_i = \int_{\sin(i\beta)}^{\sin(j\beta)} \exp(x)dx \ge \frac{1}{e}(\sin(j\beta) - \sin(i\beta)) \ge \frac{1}{2eL_0} \ge \frac{1}{6L_0}.$$

This shows

$$|\arctan a_i - \arctan a_j| \ge \frac{1}{6L_0}.$$

As $\frac{1}{e} \le a_i \le e$, we have

$$\langle \boldsymbol{h}_i, \boldsymbol{h}_j \rangle = \cos(\arctan a_i - \arctan a_j) \le 1 - \frac{1}{72L_0^2}.$$

Moreover, for each $0 \le i \le n+1$,

$$\langle \boldsymbol{h}_i, \boldsymbol{h}_{-1} \rangle = \frac{1}{\sqrt{1+a_i^2}} \le 1 - \frac{1}{72L_0^2}.$$

Thus, the lemma follows. □

Let $\lambda = 10\sqrt{L_0 \log L_0}$. In the second layer, we let

$$(\boldsymbol{K}^{(1)})^\top = 10\sqrt{L_0}\left(\boldsymbol{v}_{\mathbb{A}_{a-1}}\boldsymbol{v}_{\mathbb{B}_b}^\top + \boldsymbol{v}_{\mathbb{A}_a}\boldsymbol{v}_{\mathbb{B}_{b-1}}^\top\right), \quad \boldsymbol{Q}^{(1)} = 10\sqrt{L_0}\left(\boldsymbol{v}_{\mathbb{B}_b}\boldsymbol{v}_{\mathbb{B}_b}^\top + \boldsymbol{v}_{\mathbb{B}_{b-1}}\boldsymbol{v}_{\mathbb{B}_{b-1}}^\top\right),$$

so

$$(\boldsymbol{K}^{(1)})^\top \boldsymbol{Q}^{(1)} = 100L_0\left(\boldsymbol{v}_{\mathbb{A}_{a-1}}\boldsymbol{v}_{\mathbb{B}_b}^\top + \boldsymbol{v}_{\mathbb{A}_a}\boldsymbol{v}_{\mathbb{B}_{b-1}}^\top\right).$$

By the previous lemma, for any $1 \le i \le n+1$ and any $-1 \le j \le n+1$ with $j \ne i$, we have

$$\langle \boldsymbol{h}_i, \boldsymbol{h}_i - \boldsymbol{h}_j \rangle = 1 - \langle \boldsymbol{h}_i, \boldsymbol{h}_j \rangle \ge \frac{1}{72L_0^2}.$$

Moreover, $\bar{\boldsymbol{H}}_{i+a}^{(1)} = \lambda \boldsymbol{h}_i$ and $\bar{\boldsymbol{H}}_{n+a+b-1+i}^{(1)}$ is within $100\lambda\exp(-\alpha)$ of $\lambda\tilde{\boldsymbol{h}}_i$. Therefore, when ρ is large enough, the second-layer attention score of the correct source position $i+a$ is larger than the score of any incorrect source position by at least $100\log L_0$:

$$\bar{\boldsymbol{H}}_{i+a}^{(1)}(\boldsymbol{K}^{(1)})^\top \boldsymbol{Q}^{(1)}(\bar{\boldsymbol{H}}_{n+a+b-1+i}^{(1)})^\top \ge \bar{\boldsymbol{H}}_{j+a}^{(1)}(\boldsymbol{K}^{(1)})^\top \boldsymbol{Q}^{(1)}(\bar{\boldsymbol{H}}_{n+a+b-1+i}^{(1)})^\top + 100\log L_0.$$

Thus, in the second layer, the attention of the query position $n+a+b-1+i$ to the correct source token s_i is larger than $1/2$ for every $1 \le i \le n$.

We let $\boldsymbol{v}_0, \boldsymbol{v}_1, \boldsymbol{v}_{<\backslash n>}$ be unit vectors that are orthogonal to the $\boldsymbol{v}_C, \boldsymbol{w}_C$ defined before. We let

$$\boldsymbol{V}^{(1)}\boldsymbol{w}_C = \boldsymbol{v}_C, \quad C \in \{0, 1, <\backslash n>\}.$$

At last, we let $\boldsymbol{W}_O$ be

$$\boldsymbol{W}_O = \boldsymbol{v}_0\boldsymbol{e}_0^\top + \boldsymbol{v}_1\boldsymbol{e}_1^\top.$$

Here $\boldsymbol{e}_C$ denotes the one-hot vector corresponding to token $C \in \{0, 1, <\backslash n>\}$. Since the second layer puts more than half of its attention mass on the correct source token, the logit of the correct next token is strictly larger than the logit of the incorrect binary token. Therefore, the prediction is always correct.

For our construction, we have

$$\begin{aligned}\|\boldsymbol{\theta}\|^2 &= \|\boldsymbol{K}^{(0)}\|_2^2 + \|\boldsymbol{Q}^{(0)}\|_2^2 + \|\boldsymbol{V}^{(0)}\|_2^2 + \|\boldsymbol{K}^{(1)}\|_2^2 + \|\boldsymbol{Q}^{(1)}\|_2^2 + \|\boldsymbol{V}^{(1)}\|_2^2 + \|\boldsymbol{W}_O\|_2^2 + \lambda^2 \\ &\le \frac{\rho^2}{10} + 100 + 400L_0 + 100L_0\log L_0 \le \rho^2.\end{aligned}$$

Thus the construction satisfies the norm constraint, so the theorem follows. □

Remark J.3. One may argue that the above construction does not provide a solution that works for all sequence lengths. Indeed, such a solution cannot exist under bounded parameters: if all weights are uniformly bounded, then the attention assigned to any individual token is at most $O(1/L)$, which vanishes as the sequence length L grows. Consequently, binary copy cannot be perfectly solved for arbitrarily long sequences.

K. One-Layer 2D-RoPE Provably Solve Binary Copy with Length Generalization

K.1. Theoretical Setup

To make this section self-contained, in this subsection, we define our setup in detail before we give further theoretical results in the following subsections.

Binary Copy. For binary copy task, aligned with the previous theoretical results, we still consider the vocabulary set

$$\mathcal{V} = \{\mathtt{A}_1, \mathtt{A}_2, \cdots, \mathtt{A}_a, \mathtt{B}_1, \mathtt{B}_2, \cdots, \mathtt{B}_b, 0, 1\}.$$

Here, to use the linebreaking in our 2D-RoPE, we assume that $\mathtt{B}_1 = \texttt{<\textbackslash n>}$, and other tokens are not linebreaking tokens. For a binary string s and an index $1 \le i \le |s|$, we denote by $s_{:i}$ the length $(i-1)$ prefix of s. In the next-token prediction formulation, when the model predicts the i-th copied token, the input sequence seen by the model is

$$\mathsf{Str}(s, i) := (\mathtt{A}_1, \mathtt{A}_2, \cdots, \mathtt{A}_a,\ s,\ \mathtt{B}_1, \mathtt{B}_2, \cdots, \mathtt{B}_b, s_{:i}).$$

The target token for this input is s_i. Notice that the length of $\mathsf{Str}(s, i)$ is $|\mathsf{Str}(s, i)| = |s| + a + b - 1 + i$. Equivalently, the full copy sample corresponds to the sequence

$$(\mathtt{A}_1, \mathtt{A}_2, \cdots, \mathtt{A}_a,\ s,\ \mathtt{B}_1, \mathtt{B}_2, \cdots, \mathtt{B}_b, s),$$

but in the autoregressive loss, the logit at position $|s| + a + b - 1 + i$ is used to predict s_i. We further denote the set of all nonempty finite sequences over $\mathcal{V}$ as $\mathcal{V}^+ := \mathcal{V} \cup \mathcal{V}^2 \cup \mathcal{V}^3 \cup \cdots$.

2D-RoPE Architecture. We now define the one-layer 2D-RoPE Transformer used in theory. Let

$$u = (C_1, \ldots, C_T) \in \mathcal{V}^+$$

be an input sequence with length T, where each $C_i \in \mathcal{V}$. We first convert each token into its embedding and obtain an input matrix $\boldsymbol{X} \in \mathbb{R}^{T \times d}$ by setting

$$\boldsymbol{X}_{i,:} = \boldsymbol{w}_{C_i}^\top, \qquad 1 \le i \le T.$$

Here, d is the embedding dimension. We set d to be divisible by 4, so that the 2D-RoPE rotation can split the coordinates into a column part and a row part. We consider the one-hot embeddings of the tokens in $\mathcal{V}$ as

$$\boldsymbol{w}_{\mathtt{0}}, \boldsymbol{w}_{\mathtt{1}}, \boldsymbol{w}_{\texttt{<0:>}}, \boldsymbol{w}_{\texttt{<\textbackslash n>}}, \boldsymbol{w}_{A_1}, \boldsymbol{w}_{A_2}, \cdots, \boldsymbol{w}_{A_a}, \boldsymbol{w}_{B_1}, \boldsymbol{w}_{B_2}, \cdots, \boldsymbol{w}_{B_b} \in \mathbb{R}^d.$$

Next, we assign a 2D position to each token using the line-break rule discussed in Section 3.1. That is, the column index increases within the same row, and after the line-break token $\texttt{<\textbackslash n>}$, the row index increases and the column index is reset. Let the 2D position of the i-th token be (x_i, y_i), where x_i is the column index and y_i is the row index.

We now define the 2D-RoPE rotation. For any matrix $\boldsymbol{Z} \in \mathbb{R}^{T \times d}$, the 2D-RoPE map $R : \mathbb{R}^{T \times d} \to \mathbb{R}^{T \times d}$ maps the i-th row $\boldsymbol{Z}_{i,:}$ to

$$\boldsymbol{Z}_{i,:} \boldsymbol{R}_{x_i, y_i}^\top,$$

where

$$\boldsymbol{R}_{x_i, y_i} = \operatorname{diag}\big(\boldsymbol{R}_1(x_i), \ldots, \boldsymbol{R}_{d/4}(x_i),\ \boldsymbol{R}_{d/4+1}(y_i), \ldots, \boldsymbol{R}_{d/2}(y_i)\big) \in \mathbb{R}^{d \times d}.$$

Here, the first $d/4$ rotation blocks encode the column coordinate, and the last $d/4$ rotation blocks encode the row coordinate. For each $1 \le j \le d/2$, the block $\boldsymbol{R}_j(\cdot) \in \mathbb{R}^{2 \times 2}$ is the standard RoPE rotation matrix

$$\boldsymbol{R}_j(t) \ := \ \begin{bmatrix} \cos(t\beta_j) & -\sin(t\beta_j) \\ \sin(t\beta_j) & \cos(t\beta_j) \end{bmatrix}.$$

Here, β_j are the base frequencies associated with the $(2j-1, 2j)$ coordinate pair. In our setting, these frequencies are arbitrary hyperparameters unless otherwise specified.

Given the above embedding and 2D-RoPE rotation, we define the one-layer 2D-RoPE attention layer as

$$\boldsymbol{Y} = \mathsf{Attn}(\boldsymbol{X}) = \mathcal{S}\left(\mathrm{MASK}\left(R\left(\boldsymbol{X}\boldsymbol{Q}\right)\left(R\left(\boldsymbol{X}\boldsymbol{K}\right)\right)^\top\right)\right)\boldsymbol{X}\boldsymbol{V}^\top \in \mathbb{R}^{T \times 2}. \tag{5}$$

Here, $\boldsymbol{K}, \boldsymbol{Q} \in \mathbb{R}^{d\times d}$ and $\boldsymbol{V} \in \mathbb{R}^{2\times d}$ are the learnable parameters of this attention layer. The output $\boldsymbol{Y}$ has two coordinates at each position, corresponding to the logits for the two binary output symbols 0 and 1.

Equivalently, let

$$\boldsymbol{A} := \mathrm{MASK}\left(R(\boldsymbol{X}\boldsymbol{Q})\,(R(\boldsymbol{X}\boldsymbol{K}))^\top\right)$$

be the masked attention score matrix. If we denote the unrotated query and key vectors at position i by

$$\boldsymbol{q}_i = (\boldsymbol{X}\boldsymbol{Q})^\top_{i,:}, \qquad \boldsymbol{k}_i = (\boldsymbol{X}\boldsymbol{K})^\top_{i,:},$$

then the entries of $\boldsymbol{A}$ can be written as

$$\boldsymbol{A}_{i,j} = \begin{cases} -\infty & i < j, \\ f((x_i, y_i), (x_j, y_j), \boldsymbol{q}_i, \boldsymbol{k}_j) & i \geq j, \end{cases}$$

where

$$f((x_i, y_i), (x_j, y_j), \boldsymbol{q}_i, \boldsymbol{k}_j) = \boldsymbol{q}_i^\top \boldsymbol{R}_{x_j - x_i, y_j - y_i} \boldsymbol{k}_j.$$

The causal mask operator $\mathrm{MASK}(\cdot)$ enforces the autoregressive constraint: for $\boldsymbol{Z} \in \mathbb{R}^{T\times T}$,

$$\mathrm{MASK}(\boldsymbol{Z})_{ij} \;=\; \begin{cases} \boldsymbol{Z}_{ij}, & j \leq i, \\ -\infty, & j > i. \end{cases}$$

For any input sequence $u \in \mathcal{V}^+$ with length T, we denote the above one-layer 2D-RoPE Transformer architecture as

$$F^{(d,\boldsymbol{\beta})}_{\boldsymbol{\theta}}(u) = \boldsymbol{Y} \in \mathbb{R}^{T\times 2}, \tag{6}$$

where $\boldsymbol{\theta} = (\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V})$ are the learnable parameters, d is the embedding dimension, and $\boldsymbol{\beta}$ denotes the hyperparameters $\beta_j, 1 \leq j \leq d/2$.

The prediction distribution is obtained by applying the row-wise softmax to these two logits:

$$\begin{aligned} P^{(d,\boldsymbol{\beta})}_{\boldsymbol{\theta}}(u) &= \mathrm{softmax}\left(F^{(d,\boldsymbol{\beta})}_{\boldsymbol{\theta}}(u)\right), \\ \left[P^{(d,\boldsymbol{\beta})}_{\boldsymbol{\theta}}(u)\right]_{i,b} &= \frac{\exp\left(F^{(d,\boldsymbol{\beta})}_{\boldsymbol{\theta}}(u)_{i,b}\right)}{\sum_{b'\in\{0,1\}} \exp\left(F^{(d,\boldsymbol{\beta})}_{\boldsymbol{\theta}}(u)_{i,b'}\right)}, \end{aligned} \tag{7}$$

where i indexes positions and $b \in \{0, 1\}$ indexes binary output symbols.

K.2. Expressive Power of One-Layer 2D-RoPE

Next, we prove the detailed version of Theorem 3.1 as the following theorem states.

Theorem K.1 (Detailed Version of Theorem 3.1)**.** *Assume that the 2D-RoPE dimension $d \geq 100 + 4a + 4b$ is divisible by 4. For any $\rho > d^2$, let $\beta_j \overset{\text{i.i.d.}}{\sim} \mathrm{Unif}[0, 2\pi]$ for every $1 \leq j \leq d/2$. Then with probability at least $\left(1 - \frac{d}{\rho}\right)$ over the randomness of $\boldsymbol{\beta}$, there exists a 1-layer 2D-RoPE transformer $P^{(d,\boldsymbol{\beta})}_{\boldsymbol{\theta}}$ defined in Equations (5) to (7) such that:*

1. *$\|\boldsymbol{\theta}\|_2 \leq \rho$;*
2. *For any binary string s with length $|s| \leq \rho^{d/12}$ and any $1 \leq i \leq |s|$, the transformer correctly predicts the next copied token:*

$$\operatorname*{arg\,max}_{b\in\{0,1\}} \left[P^{(d,\boldsymbol{\beta})}_{\boldsymbol{\theta}}(\mathsf{Str}(s, i))\right]_{|s|+a+b-1+i,b} = s_i.$$

Proof of Theorem K.1. Let $M = \frac{\rho}{d}$ and $L = M^{d/6}$. Since $\boldsymbol{w}_{\mathtt{C}}$ are one-hot embeddings, there exists $\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}$ such that, for every $\mathtt{C} \in \mathcal{V}$,

$$\boldsymbol{q}_{\mathtt{C}} := \boldsymbol{Q}^\top \boldsymbol{w}_{\mathtt{C}} = \boldsymbol{R}_{a+1-b,-1} \begin{bmatrix} M \\ M \\ \vdots \\ M \end{bmatrix}, \qquad \boldsymbol{k}_{\mathtt{C}} := \boldsymbol{K}^\top \boldsymbol{w}_{\mathtt{C}} = \boldsymbol{R}_{0,0} \begin{bmatrix} M \\ M \\ \vdots \\ M \end{bmatrix}.$$

Here $\boldsymbol{q}_{\mathtt{C}}$ has d entries equal to M, while $\boldsymbol{k}_{\mathtt{C}}$ has $d/2$ entries equal to M and $d/2$ entries equal to $-M$. We also set

$$\boldsymbol{v}_{(0)} := \boldsymbol{V}\boldsymbol{w}_0 = \begin{bmatrix} M \\ -M \end{bmatrix}, \qquad \boldsymbol{v}_{(1)} := \boldsymbol{V}\boldsymbol{w}_1 = \begin{bmatrix} -M \\ M \end{bmatrix},$$

and

$$\boldsymbol{v}_{\mathtt{C}} := \boldsymbol{V}\boldsymbol{w}_{\mathtt{C}} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad \text{for any } \mathtt{C} \in \mathcal{V}\backslash\{0,1\}.$$

By construction, $\|\boldsymbol{\theta}\|_2 = \sqrt{2(a+b+3)M^2 d + 4M^2} \le \rho$.

We first need the following technical lemma.

Lemma K.2. *Let $\theta_1, \ldots, \theta_m \sim \mathrm{Unif}[0, 2\pi]$ be independent. Then for any $\delta \in (0, 0.1)$, with probability at most $\delta^{m/2}$, we have*

$$\sum_{i=1}^{m} \cos(\theta_i) \ge m - \delta.$$

Proof. First consider one $\theta \sim \mathrm{Unif}[0, 2\pi]$. Notice that $\cos\theta = 1 - 2\sin^2\frac{\theta}{2}$. Therefore, if $\cos\theta \ge 1 - \delta$, then $\sin^2\frac{\theta}{2} \le \frac{\delta}{2}$. Since $\delta < 0.1$, we must have $0 \le \theta \le \sqrt{2\delta}$ or $2\pi - \sqrt{2\delta} \le \theta \le 2\pi$. Therefore, the probability of $\sin^2\frac{\theta}{2} \le \frac{\delta}{2}$ is at most $\sqrt{\delta}$.

Notice that $\sum_{i=1}^{m}\cos(\theta_i) \ge m - \delta$ indicates that $\cos(\theta_i) \ge 1 - \delta$ for each i. Therefore, the result follows by the independency of these θ_i's. □

Now we consider back our construction of $\boldsymbol{q}, \boldsymbol{k}$. We define the following good event:

$$\mathcal{E} := \left\{ \sum_{k=1}^{d/4} \cos(r\beta_k) \le \frac{d}{4} - \delta \text{ for every } 1 \le |r| \le L + a + b + 3 \right\} \cap \left\{ \sum_{k=d/4+1}^{d/2} \cos(\beta_k) \le \frac{d}{4} - \delta \right\}.$$

For any non-zero integer r, $\cos(r\beta_k)$ has the same distribution as $\cos(\theta)$ for $\theta \sim \mathrm{Unif}[0, 2\pi]$. Therefore, by Lemma K.2 and a union bound over all possible relative column offsets, we have

$$\Pr(\mathcal{E}) \ge 1 - (L + a + b + 4)\delta^{d/8}.$$

We now work on the event $\mathcal{E}$. Fix any binary string s with $n := |s| \le L$ and any $1 \le i \le n$. Let

$$T_i := |\mathsf{Str}(s, i)| = n + a + b - 1 + i$$

be the query position for predicting s_i, and let

$$p_i := i + a$$

be the position of the corresponding source token s_i in the input string. Under the 2D position assignment, the source position p_i and the query position T_i have the same column, and their row difference is -1. Therefore, by construction,

$$\boldsymbol{A}_{T_i, p_i} = d \cdot M^2.$$

Next, we compare this score with all other positions. First consider a first-row position $1 \le j \le a + L + 1$ with $j \ne p_i$. Then j has row difference -1 from the query position, but its column is different from the column of T_i. Let r_j be this non-zero column difference. Then

$$\boldsymbol{A}_{T_i, j} = 2M^2 \sum_{k=1}^{d/4} \cos(r_j \beta_k) + \frac{d}{2} M^2.$$

Thus, on the event $\mathcal{E}$,

$$\boldsymbol{A}_{T_i, p_i} - \boldsymbol{A}_{T_i, j} = \frac{d}{2} M^2 - 2M^2 \sum_{k=1}^{d/4} \cos(r_j \beta_k) \ge 2M^2 \delta.$$

Now consider a second-row position $a + L + 2 \leq j \leq T_i$. In this case, we consider $\boldsymbol{A}_{T_i,j}$. The first $d/2$ items contributes at most $\frac{d}{2}M^2$, and the last $d/2$ items contributes $2M^2 \sum_{k=d/4+1}^{d/2} \cos(r_j \beta_k)$. Thus

$$\boldsymbol{A}_{T_i,j} \leq 2M^2 \sum_{k=d/4+1}^{d/2} \cos(\beta_k) + \frac{d}{2}M^2.$$

On the event $\mathcal{E}$, this implies

$$\boldsymbol{A}_{T_i,p_i} - \boldsymbol{A}_{T_i,j} \geq 2M^2\delta.$$

Therefore, for every $j \neq p_i$, we have the attention score gap

$$\boldsymbol{A}_{T_i,p_i} \geq \boldsymbol{A}_{T_i,j} + 2M^2\delta.$$

It follows that the softmax attention weight on the correct source position is at least

$$\mathcal{S}(\boldsymbol{A})_{T_i,p_i} \geq \frac{1}{1 + (T_i - 1)\exp(-2M^2\delta)} \geq \frac{1}{1 + 3L\exp(-2M^2\delta)}.$$

Thus, as long as

$$3L\exp(-2M^2\delta) \leq 1,$$

we have

$$\mathcal{S}(\boldsymbol{A})_{T_i,p_i} > \frac{1}{2}.$$

Finally, we translate this attention property into the output prediction. If $s_i = 0$, then the value vector at the correct source position is $\boldsymbol{v}_{(0)} = (M, -M)^\top$. Since all non-binary tokens have zero value vectors and the attention mass on the correct source position is larger than $1/2$, the first output logit is strictly larger than the second output logit. Therefore,

$$\operatorname*{arg\,max}_{b \in \{0,1\}} \left[P_{\boldsymbol{\theta}}^{(d,\boldsymbol{\beta})}(\mathsf{Str}(s,i)) \right]_{T_i,b} = 0 = s_i.$$

The case $s_i = 1$ is symmetric, because $\boldsymbol{v}_{(1)} = (-M, M)^\top$. Hence the model correctly predicts s_i for every s with $|s| \leq L$ and every $1 \leq i \leq |s|$.

Now let $\delta = M^{-3/2}$ and $L = M^{d/6}$. Since $d \geq 100 + 4a + 4b$, we have

$$(L + a + b + 4)\delta^{d/8} \leq 2M^{d/6 - 3d/16} = 2M^{-d/48} \leq \frac{1}{M}.$$

Moreover,

$$3L\exp(-2M^2\delta) = 3M^{d/6}\exp(-2M^{1/2}) \leq 1$$

when M is sufficiently large. Therefore, with probability at least $1 - \frac{1}{M} = 1 - \frac{d}{\rho}$, the constructed one-layer 2D-RoPE Transformer can correctly copy every binary string with length at most

$$L = M^{d/6} = (\rho/d)^{d/6} \geq \rho^{d/12}$$

since $\rho > d^2$. This proves the theorem. □

L. Prompt Templates of Copy Test for Frontier LLMs

Table 6. Prompt template for the **Recursive-Flip binary copy task** used for Gemini and Qwen. The input is a recursively generated binary string, and the Output is the exact same sequence.

System Prompt	User Prompt with Example Input	Output
You are taking a copying test. Your task is to copy the binary sequence exactly. Output only the copied binary sequence. Do not add any explanation, quotes, punctuation, or formatting. Do not add spaces or newlines inside the sequence.	Copy the following binary sequence exactly: `0101010101010101010101...`	`0101010101010101010101...`

Table 7. Prompt template for the **Recursive-Flip binary copy task** used for GPT and DeepSeek. We use a different input template because GPT and DeepSeek tokenizers tend to merge three consecutive digits into one token, as discussed in Appendix B.2. The target output is still the exact same binary sequence.

System Prompt	User Prompt with Example Input	Output
You are taking a copying test. Your task is to copy the binary sequence exactly. Output only the copied binary sequence. Do not add any explanation, quotes, punctuation, or formatting. Do not add spaces or newlines inside the sequence.	Copy the following binary sequence exactly: `000111000111000111000...`	`000111000111000111000...`

Table 8. Prompt template for the **binary copy** task under **Imbalanced Generation**. The example input is an imbalanced sequence over two symbols, where one symbol appears much more frequently than the other. The target output is the exact same sequence.

Prompt Template	Example Input	Output
Copy the following sequence exactly. Output only the copied sequence. Do not add any explanation, quotes, punctuation, or formatting.<\n>	a b a a a a a b a a a a a a b a a a a ...	a b a a a a a b a a a a a a b a a a a ...

Table 9. Prompt templates for the **Python List Conversion** task. Each prompt is followed by a comma-separated sequence of sensor values, and the target output is the same sequence formatted as a single-line Python list.

ID	Prompt Template	Example Input	Example Output
1	Copy the following sensor sequence exactly. Output only one single-line Python list, with no line breaks, no spaces, and no extra text:<\n>	1.55,1.71,...	[1.55,1.71,...,]
2	Rewrite the following sensor measurements as exactly one single-line Python list. Do not add any line breaks, spaces, explanation, or extra characters:<\n>	1.55,1.71,...	[1.55,1.71,...,]
3	Return the following numbers as one single-line Python list only. Keep the values and order exactly the same. No line breaks, no spaces, no extra text:<\n>	1.55,1.71,...	[1.55,1.71,...,]
4	Convert the following sensor readings into a Python list on a single line. Output only the list, with no spaces, no line breaks, and no other text:<\n>	1.55,1.71,...	[1.55,1.71,...,]
5	Repeat the following sequence exactly as a one-line Python list. Do not change any number. Do not insert line breaks, spaces, or commentary:<\n>	1.55,1.71,...	[1.55,1.71,...,]
6	Write the following measurements as exactly one Python list in a single line. Output only the list itself, with no spaces, no newlines, and no explanation:<\n>	1.55,1.71,...	[1.55,1.71,...,]
7	Copy the following numeric sequence exactly into a single-line Python list. Preserve every value and its order. No line breaks, no spaces, no extra text:<\n>	1.55,1.71,...	[1.55,1.71,...,]
8	Output exactly one single-line Python list containing the following numbers in the same order. Do not add spaces, newlines, or any surrounding text:<\n>	1.55,1.71,...	[1.55,1.71,...,]
9	Turn the following sensor values into a Python list written on one line only. Keep the sequence exactly unchanged. No spaces, no line breaks, no extra text:<\n>	1.55,1.71,...	[1.55,1.71,...,]
10	Produce exactly the following sequence as a single-line Python list. Output only the list, with no spaces, no line breaks, and no additional content:<\n>	1.55,1.71,...	[1.55,1.71,...,]

Prompt Template	Example Input	Gold Output
Copy the following sequence exactly. Output only the copied sequence and nothing else. Do not add explanations, quotation marks, or newlines.	`Sequence: a b a b a b a b b b a ...`	`a b a b a b a b b b a ...`

Table 10. Prompt template for the **repeat-structure copy test**. We first sample a short base sequence over symbols `a` and `b`, repeat it to length 1000, and then corrupt tokens with a specified probability. The model is asked to output exactly the corrupted sequence.

M. Prompt Templates for Finetuning Data

Following Python list shows the prompts templates that was uniformly and randomly sampled from when generating finetuning data. The `{s}` indicates the portion that is to be replaced with the binary string to be copied.

```
[
    # With newline
    "Please echo the next sequence verbatim (no extra text). Here it is:\n{s}",
    "Copy the text below exactly as-is (keep every character):\n{s}",
    "Task: return the string shown below. Do not add or remove anything.\n{s}",
    "Print the following content exactly, unchanged. Output only the content:\n{s}",
    "Write the following string exactly as it is:\n{s}",
    "I want you to say the following string and nothing else:\n{s}",
    "Say the following string exactly as it is:\n{s}",
    "Here is a string:\n{s}\n\nPlease print the string exactly as it is.",
    "I will give you a string. You need to repeat it exactly as it is. Here it is:\n{s}",
    "{s}\n\nPlease print the above string exactly as it is.",
    "{s}\n\nCan you write out the above string and nothing else.",
    "Can you copy a string of characters? Here is the string I want you to copy:\n{s}",
    "Repeat this string and say nothing else:\n{s}",

    # Without newline
    "Repeat the following string after me (output only the string): {s}",
    "Echo this sequence exactly (no extra characters): {s}",
    "Return the exact same string and nothing else -> {s}",
    "Copy-paste the following payload exactly as written: {s}",
    "Output EXACTLY the string below, unchanged: {s}",
    "Just say \"{s}\" and nothing else.",
    "Say '{s}' and say nothing else.",
]
```

Following is the chat template used for finetuning:

```
### user:
{input}

### assistant:
{output}
```