

Attend or Perish: Benchmarking Attention on Algorithmic Reasoning

Michal Spiegel^{♣♦} Michal Štefánik^{♣♠} Marek Kadlčík[♣] Valeria Ruscio^{★♦}
Martin Fajčík[▲] Josef Kuchař[♣]

[♣]TransformersClub @ Faculty of Informatics, Masaryk University

[♠]R&D Centre for Large Language Models, National Institute of Informatics, Japan

[★]Sapienza University, Rome

[♦]Intuition Machines

[▲]Brno University of Technology

[◇]Filevine

Abstract

While Transformer models can learn algorithmic tasks and generalize reliably to unseen, in-distribution data, they often fail catastrophically when required to extrapolate to sequence lengths beyond their training regime. Using AttentionSpan, a suite of synthetic diagnostic tasks designed to enable inspection of internal model computations, we analyze the model’s behavior on length extrapolation. Our findings indicate that this failure is not fully explained by an inability to induce general rules for the task. Instead, we identify inconsistent attention patterns as a major contributing factor: information retrieval strategies learned by individual attention heads, which fail to remain stable as sequence length increases. Reinforcing a reference attention pattern at inference time, using no out-of-distribution data, largely recovers accuracy on String Reversal and partially on Value Assignment, while leaving multi-digit arithmetic unresolved. We also study *length adaptation*, how a model trained on one length adjusts to another by fine-tuning its Key and Query parameters, and find that a single dimension in every attention head is enough, across 4 algorithmic tasks and the RULER benchmark. Adaptation does not carry past the fine-tuned lengths, and our interventions are diagnostic rather than a proposed remedy. We make our benchmark and code publicly available.

1 Introduction

As Transformer models become more popular and are applied to complex, assistant-like tasks, there is a growing interest in their reasoning capabilities. We focus on algorithmic reasoning, which isolates specific skills a general reasoner should have, such as arithmetic or sorting. This controlled setting allows us to study the fundamental limitations of the Transformer architecture, particularly its ability to extrapolate beyond the data it was trained on.

While Transformers achieve high accuracy on tasks within their training distribution, their per-

formance often collapses on longer sequences not seen during training. This raises a fundamental question: what is the underlying cause of this failure, and can it be repaired? To answer this, we move beyond black-box evaluations to investigate the model’s internal computations. We hypothesize that the self-attention mechanism, the component responsible for mixing and combining information, is a principal source of the problem. Specifically, our main hypothesis is that these generalization failures happen because the model learns attention patterns that are not consistent across different input lengths.

To investigate this, we test our hypothesis by introducing a methodology to diagnose and repair these attention patterns. Our contributions are as follows:

- We introduce AttentionSpan, a suite of synthetic diagnostic tasks with interpretability scaffolding with reference attention patterns and a methodology for evaluating attention pattern consistency in models.
- We show that models trained on algorithmic tasks fail to generalize to longer, out-of-distribution sequences, despite high in-distribution accuracy.
- We identify inconsistent attention patterns—arising from the Key (W_K) and Query (W_Q) projection matrices—as a major source of length-extrapolation failure. Our strongest, data-free causal evidence covers two tasks, one positional and one content-based; separately, we show that fine-tuning just a small subset of parameters in these matrices for all attention heads on longer sequences adapts the model to those new input sequence lengths.
- We evaluate this minimal repair across four diverse algorithmic tasks from AttentionSpan

and tasks from the RULER benchmark (Hsieh et al., 2024).

2 Related Work

A number of recent benchmarks have been developed to systematically evaluate the algorithmic reasoning and length generalization abilities of neural models. CRLS-Text (Markeeva et al., 2024) implements traditional algorithms for training and evaluating LLMs, and we extend its methodology by introducing reference attention masks for interpretability. McLeish et al. benchmark ChatGPT on the CLRS suite, finding that it outperforms specialist GNNs by leveraging Python code execution, raising questions about the nature of out-of-distribution generalization in LLMs. BIG-Bench (Srivastava et al., 2023) offers a wide range of algorithmic tasks, but as a fixed test set, it is less suitable for robust extrapolation evaluation; we address this by creating configurable generators to avoid data contamination. Flip-Flop Language Modeling (Liu et al., 2023) is a synthetic task for analyzing attention glitches, which we extend to a broader set of algorithmic tasks. Other benchmarks such as GSM-Symbolic (Mirzadeh et al., 2025) and the DeepMind mathematics suite (Saxton et al., 2019) further highlight the fragility and limitations of current models in systematic mathematical reasoning.

Many works have proposed architectural changes or new positional encoding schemes to improve length generalization. Threshold Relative Attention (TRA) (Oppen et al., 2025) introduces selective sparsity and contextualized relative distance in attention, while PRISM (Lee, 2025) and Abacus (McLeish et al., 2024a) propose specialized positional embeddings for arithmetic tasks. Jelassi et al. show that relative position embeddings enable generalization for addition, but not for multiplication, and introduce "train set priming" as a remedy. Shen et al. demonstrate that modifying positional encodings or task representations can dramatically improve arithmetic performance and length generalization.

Despite these advances, none of the above methods explain why standard Transformers fail to learn generalizable attention patterns or positional logic from data alone. (Dziri et al., 2023) highlights that even state-of-the-art models struggle with basic algorithmic tasks like addition and multiplication, suggesting a persistent and fundamental prob-

lem. Veličković et al. shows that softmax attention leads to dispersed, high-entropy attention at longer lengths, but our findings show that extrapolation failures can occur even with attention remaining sharp (low-entropy) but misallocated (Section 5.3). Behrens et al. show that Transformers can learn fundamentally different algorithms for the same task depending on random initialization, motivating our learned interventions to stabilize internal solutions. Zhou et al. demonstrate that length generalization is possible but fragile, with large variance across seeds and data order. Schwarzschild et al. show that recurrent networks can generalize algorithmic reasoning by increasing recurrent steps, motivating the search for similar scalable reasoning in Transformers.

Several empirical studies have explored the impact of data and training strategies on length generalization. Lee et al. show that small Transformers can efficiently learn arithmetic operations with carefully formatted or chain-of-thought data, but still face challenges in length generalization. Yang et al. demonstrate that, with sufficient targeted data, language models can achieve near-perfect accuracy on multi-digit arithmetic tasks, surpassing even GPT-4 in some cases.

While prior work has focused on architectural changes, engineered embeddings, or data-centric solutions, few directly diagnose the root cause of extrapolation failures in attention. Our work differs by showing that minimal, targeted interventions can restore out-of-distribution performance: reinforcing a reference attention pattern at inference time, or fine-tuning just a single column of the key and query projection matrices. This suggests that there might be a systematic issue within attention that could be alleviated by an architectural change.

3 Methodology

To systematically investigate length extrapolation failures, we introduce AttentionSpan, a suite of synthetic algorithmic tasks for controlled evaluation. Each task is procedurally generated, which allows us to create distinct in-distribution (ID) training sets and out-of-distribution (OOD) test sets. This setup enables a rigorous assessment of how models extrapolate learned algorithms to novel scenarios, such as longer sequence lengths, which is the primary focus of this study.

The benchmark includes a diverse set of algorithmic challenges: String Reversal, Long Addition,

Long Multiplication, Flip-Flop Language Modeling (FFLM) (Liu et al., 2023), and Value Assignment. These tasks are designed to probe fundamental reasoning capabilities, from positional lookups (String Reversal) to content-based retrieval (Value Assignment) and multi-step arithmetic computation (Addition and Multiplication). Example inputs and outputs for each task are provided in Table 1, with detailed descriptions available in Appendix A.

Our core experimental methodology relies on the distinction between in-distribution and out-of-distribution data. For each task, we train models on an ID dataset characterized by a constrained set of parameters, such as sequences of length 1-10 for String Reversal. We then evaluate the trained models on an OOD dataset where a key parameter is shifted beyond the training range, for instance, testing on sequences of length 11-50. This sharp split allows us to precisely measure the model’s ability to generalize the learned algorithm versus simply memorizing samples from the training data. The specific ID/OOD configurations for each task are detailed in Appendix G.

Throughout this work, we use the term *attention patterns* to refer to the specific information retrieval strategies, i.e. algorithms learned by individual attention heads. Each pattern can be understood as a specialized sub-task, such as a positional lookup (e.g., "always attend to the token two positions back") or a content-based lookup (e.g., "find the key token that has a specific property related to the current query"). For a model to generalize an algorithm, we posit that these learned patterns must be consistent—the underlying logic of the retrieval strategy must remain stable as sequence length increases. We describe a pattern as inconsistent when it fails to maintain its retrieval strategy at novel lengths, which we hypothesize is a primary cause of failures in algorithmic reasoning.

4 Analyzing Generalization Failures in Attention

We fine-tune models on each algorithmic task using only in-distribution (ID) data, training until they achieve near-perfect (> 0.99) accuracy on the ID test set. (Table 2). This confirms that the models can successfully acquire a robust in-distribution heuristic within the training distribution. However, their performance collapses when evaluated on out-of-distribution (OOD) sequences of novel lengths. This failure is not absolute; the models still exhibit

some limited extrapolation ability, suggesting they have acquired a robust in-distribution heuristic but are hindered in executing it reliably at scale. This gap between partial understanding and complete failure motivates our deeper investigation into what limits models’ length extrapolation.

5 Inconsistent Attention Patterns

To diagnose whether a model’s learned attention patterns are consistent across different sequence lengths, we introduce the concept of a *reference attention pattern*. These masks are created manually and represent our expectation of the ideal attention pattern a model should implement to robustly solve a given task. For simple tasks, the required pattern is often provably necessary. For example, to correctly solve String Reversal, the model must execute a specific retrieval algorithm: to predict the first output character, it must attend to the last input character; to predict the second, it must attend to the second-to-last, and so on. This creates a distinct diagonal attention pattern where the query at each output position ‘i’ looks back to the input position ‘N-i’, where ‘N’ is the input length.

Formally, the reference attention pattern is a discrete matrix, i.e. a boolean mask. For each target token to be predicted, it identifies the critical (reference) past tokens that the model should attend to. An element in the matrix is set to 1 if the corresponding past token carries information essential for predicting the target token; otherwise, it is 0. In Figure 1, these reference masks are visualized as red overlays on the model’s aggregated attention maps, providing a clear ground truth to identify misalignments between the expected and the learned patterns.

We use the reference attention patterns to track which tokens the model considers at each reasoning step. We rely on **attention rollout** (Abnar and Zuidema, 2020), a standard method for aggregating attention across heads and layers. This allows us to assess whether the model acquired the expected attention patterns in a generalized fashion. The attention rollout is expressed as a recursive product of attention scores across layers:

$$R = \prod_{l=1}^L \left(\frac{1}{H} \sum_{h=1}^H A_{l,h} + I \right) \quad (1)$$

where R is the final attention rollout matrix, L is the total number of self-attention layers, H is the number of attention heads in each layer, $A_{l,h}$ is the

Task	Example Input	Corresponding Output
String Reversal	d h 1 3 h 8 2 h j 2 8 3 j 2 3 H =	H 3 2 j 3 8 2 j h 2 8 h 3 1 h d
Long Addition	1240 + 4335 + 3440 =	8916
Long Multiplication	9900 * 9900 =	1980 + 0198 + 0000 + 0000 = 1089
FFLM	w 1 1 i 1 1 f 1 0 r 1 0 f 1 0 r 1	1
Value Assignment	B1 E0 D1 A1 C0 ABBEDACABCD	11101101101

Table 1: Example instances of our tasks. The spacing is adjusted for clarity and does not denote a separator of tokens. Digits in Long Addition and Long Multiplication are written *least-significant first*, so the carry propagates left to right (Appendix A); row 2 reads $421 + 5334 + 443 = 6198$ and row 3 $99 \times 99 = 9801$. How the tasks handle tokenization is described in greater detail in Appendix E.

Model	Task	ID Acc.	OOD Acc.	OOD Token-level Acc.
Llama-3.2-1B-Instruct	String Reversal	99.85	64.00	97.70
	Long Addition	100.00	15.00	79.53
	Long Multiplication	100.00	83.00	99.84
	FFLM	100.00	99.20	99.79
	Value Assignment	96.87	1.04	71.00
Qwen2.5-1.5B-Instruct	String Reversal	93.75	17.70	72.74
	Long Addition	96.87	22.91	88.83
	Long Multiplication	97.00	0.00	80.88
	FFLM	100.00	88.50	96.71
	Value Assignment	91.66	3.125	88.31
gemma-3-1b-it	String Reversal	100.00	14.00	39.52
	Long Addition	100.00	2.62	67.85
	Long Multiplication	97.91	0.00	76.65
	FFLM	100.00	90.12	96.71
	Value Assignment	98.95	0.00	19.35

Table 2: Exact match accuracy of finetuned models on AttentionSpan tasks with consistent in-distribution and out-of-distribution splits. Token-level accuracy denotes the ratio of correct tokens to total target tokens. Despite a sharp decline in OOD Accuracy in almost all cases, OOD Token-level Accuracy reveals that models correctly predict a large proportion of target tokens, indicating some extrapolation abilities are present.

(post-softmax) attention score matrix for head h in layer l and I is the identity matrix, which accounts for the residual connections.

What rollout does and does not measure. Rollout is a proxy for information flow rather than a measurement of it, and Equation (1) averages uniformly over heads, so a few heads implementing the reference pattern can be washed out by the rest. This is one reason the absolute AttnScore values in Table 3 are low. We therefore treat rollout as a coarse, population-level diagnostic and rely on per-head analyses for claims about specific mechanisms (Section 6.1); see Limitations.

Quantitatively, we leverage our dataset’s reference attention masks and aggregated attention

scores to calculate the proportion of attention scores assigned to tokens identified as essential for correct prediction, which we refer to as **Attn Score** in Table 3 (see Appendix D for details). Formally, Attn Score is defined as:

$$\text{Attn Score} = \frac{1}{N} \sum_{i=1}^N \left(\frac{1}{|T_i|} \sum_{j=1}^{|T_i|} \left(\sum_{k \in S_i} a_{i,j,k} \right) \right) \quad (2)$$

where N is the total number of samples in the dataset, T_i is the sequence of target (output) tokens for sample i , and $|T_i|$ is its length, S_i is the set of indices corresponding to the reference input tokens for sample i , $a_{i,j,k}$ is the normalized attention score aggregated in $R(1)$ from the j -th output token to the k -th input token in sample i . We will represent Attn Score as percentages.

Through the lens of Attn Score, we explore differences in models’ behavior on correct and incorrect (token-level) predictions to identify systematic attention patterns associated with errors. For String Reversal, Value Assignment and Multiplication, and for all three models, incorrect token predictions are associated with the model allocating *less* attention to the necessary input tokens (Table 3), providing initial evidence that misallocated attention could be a key failure mode. Addition is a consistent exception: there the effect is significant but runs in the *opposite* direction for every model. We do not fold this into the general trend, and discuss it separately in Section 5.2.

5.1 Inconsistent Patterns are Associated with Prediction Errors

Our analyses indicate that for String Reversal and Value Assignment, out-of-distribution (OOD) errors are linked to reduced attention on reference tokens (Table 3). The appearance of this phenomenon in these specific tasks is significant, as they are representative of copying tasks, a recently highlighted benchmark class (Arjovsky et al., 2016;

Task	Llama		Qwen		Gemma	
	Corr.	Err.	Corr.	Err.	Corr.	Err.
Addition	30.12	44.54	34.35	40.27	42.57	70.57
Multiplication	1.90	0.42	3.02	1.02	2.92	1.85
String Reversal	4.55	2.36	3.07	2.17	3.97	2.30
Value Assignment	3.33	1.26	1.27	1.20	5.04	4.91

Table 3: **Mean AttnScore (%) on reference tokens, for correct and erroneous target-token predictions.** All differences are statistically significant (Welch’s t-test). Attention shifts away from reference tokens on errors for every task except Addition, where the effect reverses for all three models (bold; Section 5.2). AttnScore is computed from attention rollout, which averages uniformly over heads and ignores the OV pathway, so absolute values are low and should be read as a coarse diagnostic. Models are the instruction-tuned variants.

Jelassi et al., 2024), and embody two distinct, fundamental retrieval modes: positional (e.g., sorting, reordering) and content-based (e.g., using variables or a knowledge base). This pattern suggests that insufficient attention to reference tokens contributes to faulty predictions. We also identified minimal necessary subpatterns for the more complex arithmetic tasks. For Multiplication the effect matches the retrieval tasks. For Addition it does not: the difference is significant but reversed, which we take up in Section 5.2. The association therefore holds across three pre-trained models and architectures on three of our four tasks (Table 3), indicating a problem that emerges commonly, though not universally.

To better understand the nature of these OOD failures, we visualize the attention patterns of individual heads that specialize in the required lookup algorithm. Figure 1 demonstrates how attention scores scatter away from the reference diagonal as input sequence length increases. Table 3 quantifies this, showing that prediction errors occur precisely when attention deviates from the reference diagonal. Interestingly, we observe that attention does not simply diffuse across neighboring tokens in OOD scenarios. Instead, it remains sharp (Figure 1) but incorrectly shifts to a distant, irrelevant token. This suggests a potential failure in how positional embeddings like RoPE (Su et al., 2023) generalize to long-range positional relationships not seen during training. We investigate the role of RoPE in length extrapolation in more detail in Appendix B.1, but find no conclusive evidence as we find only limited reliance of the model on tasks with extrapolation failures.

5.2 A Systematic Reversal on Addition

For Addition the AttnScore difference is significant but runs in the opposite direction to our hypothesis, and it does so for all three models: 44.54 vs. 30.12 (Llama), 40.27 vs. 34.35 (Qwen) and 70.57 vs. 42.57 (Gemma) on erroneous vs. correct predictions. Being consistent in sign across three architectures, the effect is unlikely to be noise, and we exclude Addition from the association reported above. We did not investigate it further, and report two candidate explanations without adjudicating between them. The measurement may be at fault: Addition is the one task whose reference tokens lie close to the query, which makes AttnScore hard to separate from ordinary locality. Alternatively our reference mask may be too coarse, since it marks the operand digits and the result but not carry propagation, which a model may implement by holding state across generated tokens rather than re-reading the input. Consistent with the latter, reference-pattern reinforcement (Section 6.1), which also presupposes a well-specified mask, is the intervention that fails on Addition.

5.3 Sharp but Misallocated Attention

Even on OOD sequences, the model often confidently (sharply) attends to incorrect tokens. This is illustrated in Figure 1, where attention in a string reversal task remains highly concentrated but targets the wrong positions. Quantitatively, as shown in Table 4, mean attention entropy stays low for OOD data, confirming that the dispersion of attention scores on longer sequences as shown by (Veličković et al., 2025) is not a key issue in our case.

This is supported quantitatively by the mean attention entropy in Table 4, which stays low for OOD data (1.90 for 60-digit addition vs. 1.80 in-distribution), even as accuracy drops. After our minimal Key/Query adjustment, both accuracy and entropy return to near in-distribution levels, which is consistent with the attention pattern being retargeted rather than merely sharpened. As noted in Section 6.5, entropy characterises the spread of attention and not the computation performed on the attended values.

6 Intervening on Attention Recovers Out-of-Distribution Performance

Our analyses so far reveal an association between inconsistent attention patterns and length extrapolation

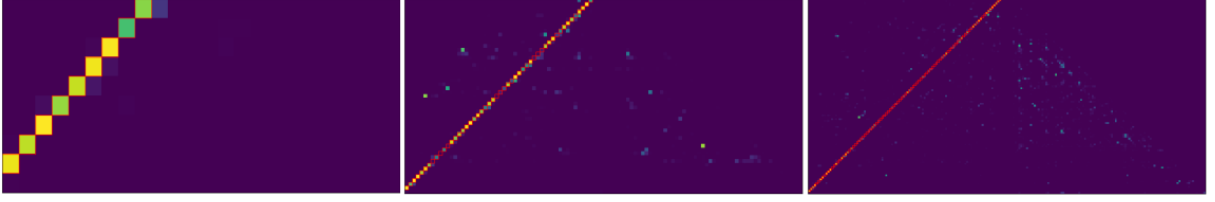


Figure 1: Attention from a single head for String Reversal, identified as performing our reference diagonal lookup algorithm. As sequence length increases from in-distribution (left) to out-of-distribution (right), attention scatters away from the correct diagonal. Red overlays mark the reference positions the head should attend to; the scattered mass outside them coincides with prediction errors.

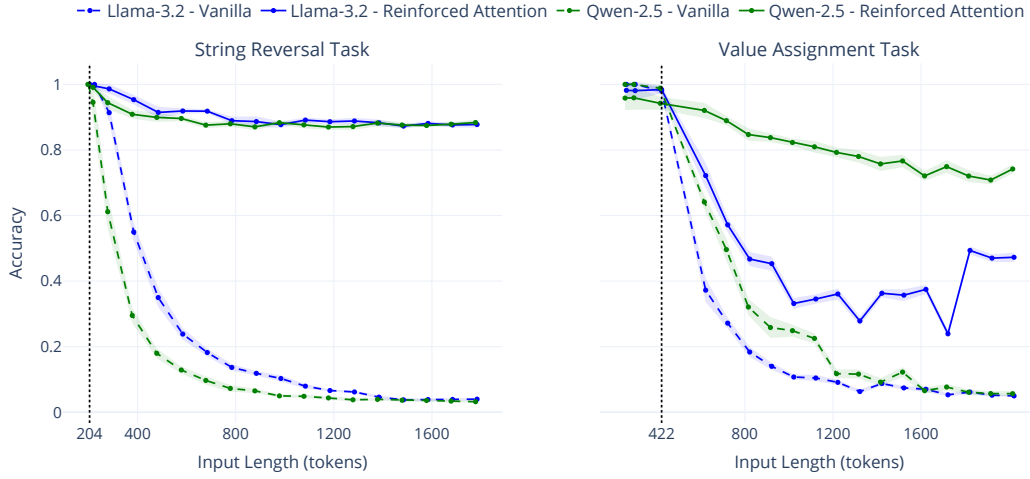


Figure 2: Effect of reinforcing the reference attention pattern at inference time. For *Llama-3.2-1B-Instruct* and *Qwen2.5-1.5B-Instruct*, we adjust post-softmax attention scores in a small set of selected heads so that reference positions receive additional weight. This substantially improves accuracy at out-of-distribution lengths relative to the unmodified models, and it does so without any out-of-distribution training data, which implicates the attention mechanism in the extrapolation failure on these two tasks. The same intervention does not recover performance on multi-digit addition or multiplication, for which no single reference pattern is well defined (Section 5.2). Dotted vertical lines indicate the maximum input sequence length seen during fine-tuning.

tion failures, and an association is not a cause. We therefore intervene on attention directly. Our first intervention, *inference-time reinforcement*, adds a constant to post-softmax attention scores at reference positions and **uses no out-of-distribution data**; where it succeeds, attention placement was the binding constraint. It works for String Reversal, partially for Value Assignment, and not at all for multi-digit arithmetic, so our cleanest causal evidence covers those two tasks. We do not claim that fixing attention inconsistency is sufficient for length extrapolation in general.

6.1 Attention Reinforcement using Reference Patterns

Having established reference attention patterns for each task, we leverage them to directly intervene in the model’s attention mechanism and test

whether reinforcing these patterns can adapt to out-of-distribution (OOD) lengths. Our approach is designed to causally link attention deficits to OOD failures by selectively amplifying attention to reference tokens in a minimal set of attention heads.

We begin by identifying the attention heads most responsible for implementing the desired reference pattern. For each head h , we compute the sum of its post-softmax attention scores on reference tokens given in-distribution (ID) sample of length L as measured by the total attention scores allocated to the set of reference token indices $\mathcal{R}$:

$$\text{score}(h) = \sum_{i=1}^L \sum_{j \in \mathcal{R}} A_{ij}^{(h)}. \quad (3)$$

This score is averaged across multiple input samples. Heads are ranked by this score, and we select the top N heads for intervention, where N is cho-

Setting	Acc.	Entropy	Std	Min	Max
5-digit (ID)	1.00	1.80	0.52	0.53	3.65
60-digit (base)	0.18	1.90	0.67	0.57	4.26
60-digit (adapted)	0.87	1.68	0.79	0.69	4.31

Table 4: Accuracy and mean attention entropy for 5-digit (in-distribution, ID) and 60-digit (out-of-distribution, OOD) addition tasks using *Llama-3.2-1B-Instruct* (512 heads). Both 60-digit rows are out-of-distribution: “base” is trained only on ID data, while “adapted” is further fine-tuned on OOD data by updating a single dimension in each W_Q and W_K projection matrix. The results show that attention entropy remains low for OOD data, even as accuracy drops, and that the intervention adapts both accuracy and entropy.

sen to maximize end-to-end task performance after intervention. For String Reversal, we identified that intervening on a minimal set of 3 attention heads was sufficient. For Value Assignment we identified 40 such attention heads (out of 512 heads across all layers in *Llama-3.2-1B-Instruct*).

During OOD inference, we apply our intervention only to these selected heads. For simple tasks like String Reversal, we add a constant value $\delta > 0$ to the post-softmax attention scores at reference token positions, effectively reinforcing the ideal pattern. The value of δ is manually optimized for each task to maximize performance and applied uniformly across samples. This direct modification leads to dramatic improvements in OOD accuracy, as shown in Figure 2.

For more complex tasks such as Value Assignment, the reference pattern is distributed across multiple heads, and a global reinforcement of the whole reference pattern in each head is not effective. Here, we use a conditional strategy: the constant δ is added only if the original attention score at a reference token position exceeds a threshold C , reinforcing existing sub-patterns rather than imposing the full reference pattern. While this improves performance, the accuracy boost is sometimes lower, as weak activations below the threshold are not reinforced.

An attention score $A_{ij}^{(h)}$ of head h at the i -th row and j -th column is modified as follows:

$$A'_{ij} = \begin{cases} A_{ij}^{(h)} + \delta & \text{if } A_{ij}^{(h)} > C \\ A_{ij}^{(h)} & \text{otherwise} \end{cases} \quad (4)$$

where C is a threshold optimized for maximal performance in Value Assignment, or set to 0 for String Reversal, in which case the attention score is always modified.

As shown in Figure 2, this intervention leads to an absolute increase in accuracy on OOD samples of approximately 90 % across all input lengths for String Reversal, with a weaker improvement for Value Assignment oscillating around 40–50 %. Because the intervention uses no OOD training data and touches nothing but post-softmax attention scores in the small set of selected heads identified above, this provides **causal evidence that insufficient attention to reference tokens is a principal cause of extrapolation failure on these two tasks**. We are explicit about the scope of that claim: it covers String Reversal strongly and Value Assignment partially, and, as we discuss next, it does not transfer to multi-digit arithmetic.

While this *intervention by a reference attention pattern* is effective for simple tasks like String Reversal and Value Assignment, it does not extend to more complex algorithms such as multi-digit addition or multiplication. We hypothesize two main reasons for this limitation. First, the algorithmic logic for these tasks is often distributed across multiple attention heads, with each head specializing in a different sub-task. Second, there is no unique, ground-truth attention pattern for these tasks. In multi-digit addition, for instance, the model might learn several valid strategies for handling the carry operation. To address this, in Appendix B, we introduce an approach that learns, rather than prescribes, the attention correction. On every evaluated task and model we find a bias $B^{(h)}$ that adapts the model to novel, longer sequence lengths. This bias is itself trained on OOD samples, so the result demonstrates that a sufficient correction exists *within the attention scores alone*, rather than demonstrating length extrapolation. That localisation is itself non-trivial.

6.2 Minimal Parameter Adjustments for Length Adaptation

Building on the learned attention interventions detailed in Appendix B, we investigated which specific parameters govern length adaptation. We find that fine-tuning a minimal subset of Key and Query parameters is sufficient to adapt the model to novel input lengths.

Concretely, for a per-head key or query projection $W \in \mathbb{R}^{d_{\text{model}} \times d_{\text{head}}}$ we unfreeze a *single column* $W_{:,c} \in \mathbb{R}^{d_{\text{model}}}$ and freeze the remaining $d_{\text{head}} - 1$ columns, training one of the coordinates along which queries and keys are compared inside each head. This is neither a one-dimensional subspace of the weight matrix in isolation nor a

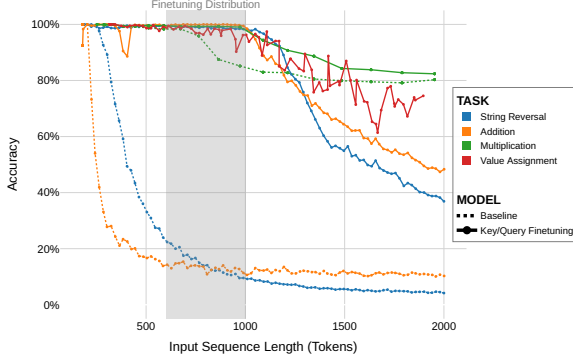


Figure 3: Accuracy evolution of the original model versus its version with a fine-tuned minimal subset of Key/Query parameters. Results are shown for Llama-3.2-1B-Instruct; full evaluations for Qwen2.5-1.5B-Instruct and Gemma-3-1B-IT across two length distributions (600–1000 and 1500–1750 tokens) are provided in Appendix I. Most configurations exhibit near-perfect adaptation within the adapted range, though Gemma-3-1b-it consistently fails to adapt as successfully as the other models (Section 6.3).

low-rank adapter. The same coordinate is trained in every head of every layer, with the rest of the model frozen; for Llama-3.2-1B ($d_{\text{model}} = 2048$, $d_{\text{head}} = 64$) this is 2048 parameters per head per projection. We take $c = d_{\text{head}}$, the slowest-rotating RoPE component, but do not claim it is privileged: in preliminary experiments every coordinate we tried adapted the model, some more readily than others (see Limitations).

Experiments We evaluate the effect of minimal Key and Query updates by examining model performance across sequence lengths up to 2,000 tokens, comparing the original task-adapted models against versions with a fine-tuned subset of attention parameters. Using Llama-3.2-1B-Instruct, Qwen2.5-1.5B-Instruct, and Gemma-3-1B-IT, we tested two adaptation regimes—600–1,000 and 1,500–1,750 tokens—requiring the latter to interpolate across a significant gap from the initial 600-token training cap.

As shown in Figure 3, minimal adaptation of a single head dimension allows Llama-3 to achieve near-perfect task accuracy across the adapted length range. In contrast, while the intervention prevents total collapse in Gemma-3-1b-it, this model fails to reach the same performance parity and exhibits inconsistent accuracy as length increases. We report this in Section 6.3 rather than setting it aside, since it delimits the generality of the method. Because the adjustment is trained on sequences of the

target length, the gains are confined to the adapted range: this is length adaptation rather than extrapolation.

6.3 Failure to Adapt on Gemma-3

Gemma-3-1b-it is the one model in our study for which the single-column W_K/W_Q adjustment does not restore performance, and it fails consistently across every task and both adaptation regimes. We did not investigate why, so we report it as a limitation on the generality of the method rather than as something we can explain. The most likely reason is that Gemma-3 interleaves local and global attention layers, whereas our procedure adapts the same coordinate in every head of every layer and so implicitly treats all layers alike. Whatever the cause, the result shows that the intervention is not architecture-independent, and we describe it as effective on the models we tested rather than in general.

6.4 Generalization to Natural Language Tasks

To evaluate our findings beyond synthetic tasks, we assess our methodology on RULER (Hsieh et al., 2024), which probes the effective context window of language models through 13 tasks across four categories (Retrieval, Multi-hop Tracing, Aggregation, and Question Answering) and shows that most models collapse well before their claimed context limits. Our algorithmic experiments required an initial supervised fine-tuning (SFT) phase to accommodate specialised input formats, which introduces potential distribution bias. Applying the minimal W_K/W_Q adjustment directly to pretrained models, with no task-specific SFT stage, lets us test whether the attention instability we identified is an artifact of that regime. The same adjustment helps here too, which suggests that it is not.

We evaluated all 13 tasks; Figure 4 presents one representative task from each category, with the full evaluation in Appendix J. The intervention targets the attention drift seen in retrieval-heavy tasks such as Multi-needle NIAH and Variable Tracking, where performance degrades sharply as sequence length increases. It is sufficient to reach near-perfect accuracy on several of these tasks where the baseline fails, narrowing the gap between claimed and effective context windows. As in the algorithmic setting, the gains are confined to the adapted length range.

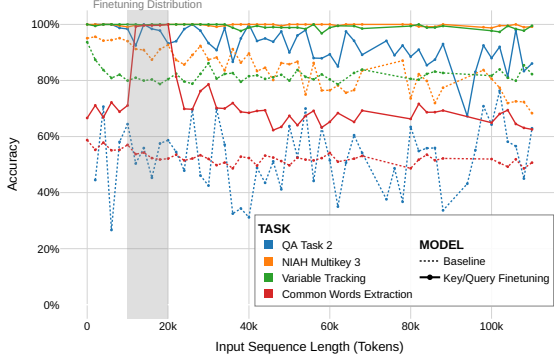


Figure 4: Performance comparison on RULER tasks of a Llama-3.2-1B-Instruct as baseline compare to the same model with minimal Key/Query adjustment. Minimal W_K/W_Q adaptation restores near-perfect retrieval accuracy.

6.5 Key/Query Adjustments Restore Attention Patterns

Qualitative analysis of the attention maps shows that the adjustment does not merely sharpen a diffuse signal. In the baseline OOD failure mode the attention maxima do not dissolve into noise; instead, as seen in Figure 5, the model’s focus flips from the reversed diagonal that String Reversal requires to a local autoregressive diagonal, attending only to the most recently generated tokens. The model appears to fall back on a generic local-copying mechanism once its learned global indexing breaks down at novel lengths. Fine-tuning W_K and W_Q relocates the highest attention weights from these local tokens back to the distant, task-correct ones identified during in-distribution training, which indicates that the adjustment moves attention mass rather than suppressing entropy or noise. We do not read more into it than that: attention maps describe where the model reads, not what it computes with what it reads, so restored placement is consistent with restored computation without demonstrating it (see Limitations). We analyze this further using effective rank in Appendix C.

7 Conclusion

We investigated why Transformers fail to generalize learned algorithms to out-of-distribution inputs, and introduced a methodology to diagnose the failure and intervene on it. Our evidence supports two claims of different strength. On String Reversal and Value Assignment, reinforcing a reference attention pattern at inference time, using no OOD data, recovers much of the lost accuracy, which

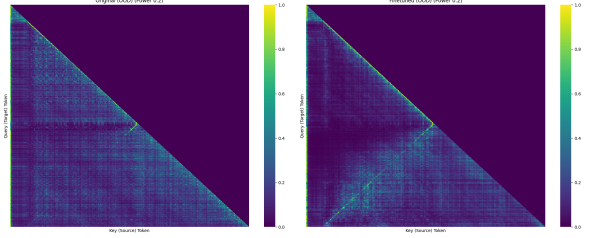


Figure 5: Restoration of attention patterns in String Reversal. The baseline OOD model (left) defaults to a local autoregressive diagonal, whereas the Key/Query adjustment restores the reversed diagonal maxima (right) consistent with the in-distribution behavior.

causally implicates attention placement for those tasks. Separately, fine-tuning a single column of the key and query projections adapts the model to longer sequences on three of four algorithmic tasks and on RULER, for two of the three architectures tested; this bounds how much of the network must change rather than showing that the underlying algorithm was intact. Our interventions are post-hoc repairs rather than a practical remedy. The question motivating the analysis is whether this instability is avoidable by design, and we offer the diagnosis in the hope of making that answerable.

Limitations

Task coverage. AttentionSpan consists of synthetic tasks, chosen so that a reference attention pattern can be specified by hand. That design bounds what we can test. Of our tasks, only String Reversal and Value Assignment have a reference pattern specific enough for the inference-time intervention to work, and our causal evidence rests on those two. Addition and multiplication admit no single correct pattern, and our masks for them are coarse enough that we cannot interpret the AttnScore results (Section 5.2). Whether the account holds for tasks outside this suite is untested.

The minimal fine-tuning experiment measures adaptation, not extrapolation. Beyond those two tasks, our results come from fine-tuning Key and Query parameters on sequences of the target length. That experiment is designed to show how a model adapts to a new length rather than to make it extrapolate, and accuracy does not improve past the fine-tuned range. It also means we cannot distinguish an algorithm that was intact and merely starved of the right inputs from one that had to learn new long-range positional behaviour. Appendix B.1 leans towards the second reading, since

the learned W_Q/W_K dot product overfits to the absolute positions seen during training, and we leave the tension unresolved.

We measure attention, not information flow. AttnScore is computed from attention rollout, which aggregates post-softmax weights, averages uniformly across heads, and ignores the OV pathway, the MLPs, and the magnitudes of the values being mixed. It tells us where attention mass sits, not what the model does with it, and the low absolute values in Table 3 partly reflect head averaging rather than weak effects. Activation patching along the output-relevant pathway would close this gap, and we have not done it.

“Minimal” is a claim about parameter count. Our result is that a single-column W_K/W_Q update *suffices* for length adaptation, which upper-bounds the amount of change required. It does not yet establish that this particular parameterisation is privileged, and we flag three controls we have not run. First, we have no full-model fine-tuning or LoRA baseline trained on the same OOD data, so we cannot quantify how much of the gain is attributable to the specific parameters chosen as opposed to any small update on longer sequences. Second, we have not ablated *which* column matters: comparing the last (slowest-rotating) RoPE coordinate against a randomly chosen coordinate would test our positional interpretation directly. Third, we have not tried the matched-budget update to W_V , which would test whether the key/query pathway is specifically implicated rather than attention parameters generally. Until those controls are run, “minimal single-column W_K/W_Q adaptation is sufficient” should be read as a statement about sufficiency and parameter count, not as a claim that no comparably small alternative would work.

Model scale and architecture. We test three models between 1B and 1.5B parameters. Larger models see far more data during pretraining and may have learned quite different internal dynamics, so we do not know whether any of this transfers to them. Even among our three, the intervention works on two and fails consistently on Gemma-3-1b-it (Section 6.3), which we did not investigate.

References

- Samira Abnar and Willem Zuidema. 2020. [Quantifying attention flow in transformers](#). *Preprint*, arXiv:2005.00928.

Martin Arjovsky, Amar Shah, and Yoshua Bengio. 2016. [Unitary evolution recurrent neural networks](#). *Preprint*, arXiv:1511.06464.

Freya Behrens, Luca Biggio, and Lenka Zdeborová. 2025. [Counting in small transformers: The delicate interplay between attention and feed-forward layers](#). *Preprint*, arXiv:2407.11542.

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

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, and Boris Ginsburg. 2024. [RULER: What’s the real context size of your long-context language models?](#) In *First Conference on Language Modeling*.

Samy Jelassi, David Brandfonbrener, Sham M. Kakade, and Eran Malach. 2024. [Repeat after me: Transformers are better than state space models at copying](#). *Preprint*, arXiv:2402.01032.

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

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

Philip Heejun Lee. 2025. [Position as probability: Self-supervised transformers that think past their training for length extrapolation](#). *Preprint*, arXiv:2506.00920.

Bingbin Liu, Jordan T. Ash, Surbhi Goel, Akshay Krishnamurthy, and Cyril Zhang. 2023. [Exposing attention glitches with flip-flop language modeling](#). *Preprint*, arXiv:2306.00946.

Larisa Markeeva, Sean McLeish, Borja Ibarz, Wilfried Bounsi, Olga Kozlova, Alex Vitvitskyi, Charles Blundell, Tom Goldstein, Avi Schwarzschild, and Petar Veličković. 2024. [The clrs-text algorithmic reasoning language benchmark](#). *Preprint*, arXiv:2406.04229.

Sean McLeish, Arpit Bansal, Alex Stein, Neel Jain, John Kirchenbauer, Brian R. Bartoldson, Bhavya Kailkhura, Abhinav Bhatele, Jonas Geiping, Avi Schwarzschild, and Tom Goldstein. 2024a. [Transformers can do arithmetic with the right embeddings](#). *Preprint*, arXiv:2405.17399.

Sean McLeish, Avi Schwarzschild, and Tom Goldstein. 2024b. [Benchmarking chatgpt on algorithmic reasoning](#). *Preprint*, arXiv:2404.03441.

- Iman Mirzadeh, Keivan Alizadeh, Hooman Shahrokhi, Oncel Tuzel, Samy Bengio, and Mehrdad Farajtabar. 2025. [Gsm-symbolic: Understanding the limitations of mathematical reasoning in large language models](#). *Preprint*, arXiv:2410.05229.
- Mattia Oppel, Roland Fernandez, Paul Smolensky, and Jianfeng Gao. 2025. [Tra: Better length generalization with threshold relative attention](#). *Preprint*, arXiv:2503.23174.
- David Saxton, Edward Grefenstette, Felix Hill, and Pushmeet Kohli. 2019. [Analysing mathematical reasoning abilities of neural models](#). *Preprint*, arXiv:1904.01557.
- Avi Schwarzschild, Eitan Borgnia, Arjun Gupta, Furong Huang, Uzi Vishkin, Micah Goldblum, and Tom Goldstein. 2021. [Can you learn an algorithm? generalizing from easy to hard problems with recurrent networks](#). *Preprint*, arXiv:2106.04537.
- Ruoqi Shen, Sébastien Bubeck, Ronen Eldan, Yin Tat Lee, Yuanzhi Li, and Yi Zhang. 2023. [Positional description matters for transformers arithmetic](#). *Preprint*, arXiv:2311.14737.
- Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, et al. 2023. [Beyond the imitation game: Quantifying and extrapolating the capabilities of language models](#). *Preprint*, arXiv:2206.04615.
- Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. 2023. [Roformer: Enhanced transformer with rotary position embedding](#). *Preprint*, arXiv:2104.09864.
- Petar Veličković, Christos Perivolaropoulos, Federico Barbero, and Razvan Pascanu. 2025. [softmax is not enough \(for sharp out-of-distribution\)](#).
- Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, and 3 others. 2020. [Transformers: State-of-the-Art Natural Language Processing](#). In *Proc. of the 2020 Conf. EMNLP: System Demonstrations*, pages 38–45. ACL.
- Zhen Yang, Ming Ding, Qingsong Lv, Zhihuan Jiang, Zehai He, Yuyi Guo, Jinfeng Bai, and Jie Tang. 2023. [Gpt can solve mathematical problems without a calculator](#). *Preprint*, arXiv:2309.03241.
- Yongchao Zhou, Uri Alon, Xinyun Chen, Xuezhi Wang, Rishabh Agarwal, and Denny Zhou. 2024. [Transformers can achieve length generalization but not robustly](#). *Preprint*, arXiv:2402.09371.

A Task Descriptions

A.1 String Reversal

This task requires the model to generate the input sequence in the reverse order. The task generator can be configured by the character set and the range of the input length.

A.2 Long Multiplication

Long multiplication is parametrized by the digit length of two operands and optional padding. The solution contains a sequence of intermediate products, which are then summed together into the final result. The digit ordering is consistent with the long addition task.

A.3 Long Addition

This task consists of adding several multi-digit numbers. The digits are ordered from the least significant to the most significant. The ordering of the digits is given by the standard addition algorithm where we compute the lower order digits first in order to be able to propagate the carry to the topmost digit. The problem generator can be parametrized by the number of operands, their length in digits, and whether short numbers are padded with zeros. As a subtask of long multiplication, it provides further insight into the inner functioning of models on these arithmetic tasks.

A.4 Value Assignment

In this task, the problem specifies a translation table from an input alphabet to an output alphabet. The model is then required to translate an input string, symbol by symbol. The character sets, and the string length can be configured. Value assignment is a subtask of many algorithmic tasks where we work with symbolic representations.

A.5 Flip Flop Language Modeling

Flip Flop Language Modeling, as introduced by (Liu et al., 2023) represents a simulation of memory composed of a single one-bit registers. We extend this into multiple registers problem, adding a new flip command that flips the value of the specific register. The input is a sequence of read, write, ignore, and flip instructions, each with the register index specified as a first operand. The sequence ends with a read instruction, and the solution is the bit value currently stored at the selected register. The parameters of the task can specify how many

registers are used, the length of the instruction sequence, and whether flip commands are used.

B Learned Interventions to Attention Scores

In this approach, we freeze the entire fine-tuned model and for each attention head h , we insert a small, trainable component that computes a bias $B^{(h)}(x)$ as a function of the layer input x . The original pre-softmax attention logits $Z^{(h)}(x)$, also a function of x , are kept frozen, and only $B^{(h)}(x)$ is learned. The corrected attention scores for a given head h are then given by:

$$A_{\text{fix}}^{(h)} = \text{softmax}(Z^{(h)}(x) + B^{(h)}(x)). \quad (5)$$

Learned Intervention Architectures We experimented with several architectures for this trainable component. **(1) Trainable bias:** we learn a single bias matrix for each attention head:

$$B_{\text{bias}}^{(h)} \in \mathbb{R}^{d_{\text{seq_len}} \times d_{\text{seq_len}}} \quad (6)$$

which is added to the pre-softmax attention logits $Z^{(h)}(x)$ for head h . Notably, this approach requires training a separate bias matrix for each sequence length. It is effective for tasks with input-independent patterns, such as the diagonal pattern in String Reversal, but fails on tasks that require content-based lookups like Value Assignment which requires to find the correct output symbol by association with the specific input symbol.

(2) Auxiliary Multi-Head Attention (MHA) module: To allow capturing more complex, content-dependent patterns, we learn new key and query projection matrices:

$$Q' = xW_Q^{(h)'} \quad K' = xW_K^{(h)'} \quad (7)$$

where $W_Q^{(h)'}, W_K^{(h)'} \in \mathbb{R}^{d_{\text{model}} \times d_{\text{head}}}$ and x is the layer input. The queries and keys are then rotated using a rotation matrix $R_{\Theta, m}, R_{\Theta, n}$ (see [Su et al. \(2023\)](#) for details), and the attention logits are computed as:

$$B_{\text{MHA}}^{(h)} = \text{softmax}\left(\frac{R_{\Theta, m} Q' R_{\Theta, n} K'^{\top}}{\sqrt{d_{\text{head}}}}\right). \quad (8)$$

The resulting bias $B(x)$ is added to the original attention logits. This auxiliary MHA module is trained exclusively on OOD samples and successfully adapts the model across all algorithmic tasks we evaluated. Because it is trained on OOD samples, it shows that a sufficient correction exists within the attention scores alone; it is not evidence of extrapolation.

B.1 The role of RoPE in Length Extrapolation

A common hypothesis for length extrapolation failures points to the limitations of positional encoding schemes like Rotary Position Embeddings (RoPE). To investigate this, we conducted a case study on a model fine-tuned for the String Reversal task. We performed a series of ablation experiments, including removing RoPE entirely during inference. Surprisingly, these interventions had no negative impact on the model’s in-distribution performance, indicating that for this specific task, the model had learned to bypass RoPE and develop its own internal mechanism for positional awareness. Crucially, despite not using RoPE, the model still suffers from the same extrapolation failures on longer sequences.

How can a model learn positional logic without relying on RoPE?

A possible reason could be that the input to an attention block at layer ℓ , denoted as $x_m^{(\ell)}$, is the output of the preceding layer. This vector has already accumulated information from all previous computations, which can implicitly encode absolute position. For example, early-layer MLPs or attention heads might learn to create features that correspond to a token’s distance from the beginning or end of the sequence. Another reason could be that the model can learn W_Q and W_K matrices that are specifically tuned to leverage these implicit positional signals. Instead of relying on RoPE’s explicit rotations, the attention mechanism learns to directly compute attention scores based on absolute positions. For a task like String Reversal of length L , the model learns projections such that the raw dot product $(x_m W_Q)^{\top} (x_n W_K)$ is maximized when $n = L - 1 - m$. This creates an “absolute position” bias for the attention head.

The reason this learned mechanism still fails to extrapolate is that it becomes overfitted to the distribution of positions seen during training. The features and projection weights are optimized for a specific range of absolute positions (e.g., 0 to 512). When presented with a longer sequence, like 600, the learned function produces unpredictable and uncalibrated attention logits, causing the softmax output to break down.

In conclusion, our findings show that the model can learn a brittle, implicit form of positional reasoning directly within its attention parameters. The failure to extrapolate is therefore not simply a failure of the positional encoding scheme, but a more

fundamental failure of the learned W_Q and W_K matrices to generalize beyond their training distribution. This reinforces our central thesis that the attention parameters themselves are the primary focus of these OOD failures.

C Effective rank analysis

To quantitatively bridge the qualitative shift to the model’s global behavior, we analyze the effective rank of the softmaxed attention scores across all heads as sequence length increases. We find that the effective rank "explodes" in OOD scenarios (Figure 6), signaling a transition from a low-rank, structured retrieval manifold to a disorganized state where queries attend to a high-diversity set of tokens. Importantly, while entropy measures the "spread" of focus, effective rank captures the "diversity" of misallocated attention targets; even if attention remains sharp (low entropy), a high effective rank indicates that the model is effectively "scattering" its focus across multiple irrelevant positions.

Our surgical adjustment suppresses this rank explosion, forcing the model to converge back to the low-rank manifold required for the task. As shown in Figure 6, the effective rank of the modified model stays near in-distribution levels even as sequence length reaches 2,000 tokens. This reduction in rank co-occurs with the restoration of the reversed diagonal pattern. We report it as a correlation: effective rank is computed from the same attention matrices whose structure we are describing, so it characterises what the intervention does to attention rather than providing independent evidence that low-rank attention is a prerequisite for length generalization.

While attention entropy remains low OOD (attention stays sharp), we can see a marked increase in the effective rank of attention matrices, suggesting that attention becomes globally less structured and relies on a larger number of modes. After our minimal W_Q/W_K repair, effective rank returns near ID levels, suggesting the intervention restores a low-dimensional, consistent retrieval mechanism.

D Attention Score on Reference Tokens

The proportion attention score attributed to reference tokens is computed per each row of the aggregated attention, that is for each predicted token, separately. This attributes to the need to investigate the proportion of information that has influenced

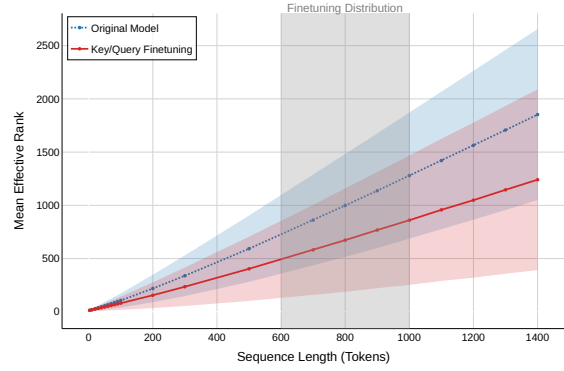


Figure 6: Evolution of the effective rank of attention scores across increasing input lengths on String Reversal task. Per each input length ranks are averaged across layers, heads and multiple samples. The baseline model shows a sharp increase in rank (scattering), whereas our W_K/W_Q adjustment maintains a low-rank structure consistent with in-distribution behavior.

a given output representation or output token. The result is then averaged across the whole sample or the whole batch to get an idea of how the model attributes attentions score on a given distribution of data.

E Tokenization of training and evaluation samples

With the exclusion of the instruction prompt, we tokenize the few-shot examples and the data points themselves into single character-level tokens. This is important to prepare the reference attention masks. Without tokenizing like this it would be possible to evaluate the attention patterns because different tokenization schemes wildly change the nature of the task and distribution of critical information between tokens. However, the fine-tuned models were able to parse this representation and fit the task as can be seen in the resulting accuracies after training.

F Training Hyperparameters

The following configuration summarizes the setup used for fine-tuning (or training from scratch) of our models.

Model:

- **Name:** meta-llama/Llama-3.2-1B-Instruct
- **Architecture Configuration:**
 - Attention Dropout Probability: 0.0
 - Hidden Dropout Probability: 0.0

Training Hyperparameters:

- **Epochs:** 1
- **Batch Size:** 4
- **Optimizer:** AdamW
- **Optimizer Parameters:**
 - Learning Rate: 5×10^{-6}
 - β_1 : 0.95
 - β_2 : 0.999
 - Weight Decay: 0.2

These hyperparameters are chosen on the basis of a hyperparameter search that was executed on String Reversal and Addition tasks, the results of the search was averaged over these two tasks. The hyperparameter search can be reproduced by running the prepared script in our codebase.

The conclusion of the hyperparameter search was that, for both tasks, smaller batch size, smaller learning and weight decay were effective in increasing accuracy in OOD. The effect of using dropout in attention or hidden layers was highly task-dependent and inconclusive, so we decided not to use it.

All our experiments were run on a single Nvidia A100 GPU card and required less than 12 hours to converge. As we document in our codebase, our experiments employ HuggingFace Transformers library (Wolf et al., 2020) v4.48.1 and PyTorch v2.5.1.

G OOD Evaluation

G.1 Long Addition Task Evaluation Parameters

The following configuration details the evaluation setup for the Long Addition task.

In-distribution:

- 2 operands
- Each number is 1-4 digits long

Out-of-distribution:

- 2 operands
- Each number is 5-10 digits long

G.2 FFLM Task Evaluation Parameters

The following configuration details the evaluation setup for the FFLM task.

In-distribution:

- Use the flip command
- Each string is composed of 10 commands
- Each instance works with 2 different registers

Out-of-distribution:

- Use the flip command
- Each string is composed of 11-100 commands
- Each instance works with 2 different registers

G.3 Long Multiplication Task Evaluation Parameters

The following configuration details the evaluation setup for the Long Multiplication task.

In-distribution:

- Each number is 1-3 digits long

Out-of-distribution:

- Each number is 4-6 digits long

G.4 String Reversal Task Evaluation Parameters

The following configuration details the evaluation setup for the String Reversal task.

In-distribution:

- Each string is 1-10 characters long
- The character set is composed of at least 50 unique characters

Out-of-distribution:

- Each string is 11-50 characters long
- The character set is composed of at least 50 unique characters

G.5 Value Assignment Evaluation Parameters

The following configuration details the evaluation setup for the Value Assignment task.

In-distribution:

- The number of unique tuples in the translation table is 5
- The length of the string to be translated is 5

Model	Task	ID Acc.	OOD Acc.
From Scratch	String Reversal	5.21	0.0578
	Long Addition	9.37	0.1713
	Long Multiplication	18	0.1302
	FFLM	68.75	0.0129
	Value Assignment	4.17	0.3060

Table 5: Performance of models trained from random initialization. As mentioned in the main paper, we initiated experiments training models from scratch to evaluate performance without the benefit of pre-training. The results show that while the models could achieve some accuracy on the in-distribution (ID) data, they consistently failed to generalize, with out-of-distribution (OOD) accuracy remaining near-zero across all tasks. Due to this poor generalization performance, we did not pursue this line of research further.

Out-of-distribution:

- The number of unique tuples in the translation table is 10-50
- The length of the string to be translated is 10-20

H Training models from random initialization

All experiments in the main paper start from pre-trained checkpoints. To check whether our findings depend on pretraining, we also trained models of the same architecture from random initialization on each task, using the hyperparameters of Appendix F. As Table 5 shows, these models did not reach usable in-distribution accuracy, so they could not serve as a baseline for the length-extrapolation analysis: a model that has not learned the task in-distribution cannot inform us about why a learned algorithm fails to extend to longer inputs. We therefore restrict our analysis to pretrained models and note this as a scope condition: our conclusions concern algorithms acquired by fine-tuning a pre-trained Transformer, not algorithms learned from scratch.

I Minimal Key/Query Updates on All Tasks and Models

This appendix reports the complete set of minimal W_K/W_Q adaptation results referenced in Section 6.2. For each of the three models we show both adaptation regimes: fine-tuning on sequences of 600–1,000 tokens (Figures 7, 9 and 11) and on 1,500–1,750 tokens (Figures 8, 10 and 12), the latter requiring the model to interpolate across a

gap from its original 600-token training cap. Each plot compares the task-adapted baseline against the same model with a single adapted W_K/W_Q column, evaluated across input lengths up to 2,000 tokens.

Two patterns are consistent across the figures. First, for Llama-3.2-1B-Instruct and Qwen2.5-1.5B-Instruct, the adaptation restores accuracy across the adapted range on most tasks, and accuracy falls off outside it. The behaviour is length adaptation, not extrapolation. Second, Gemma-3-1b-it does not reach parity in either regime on any task (Section 6.3).

J Full RULER Evaluation

Figure 13 reports the complete evaluation across all 13 RULER tasks for *Llama-3.2-1B-Instruct*, comparing the unmodified pretrained baseline against the same model after the minimal W_K/W_Q column adjustment. Section 6.4 presents four representative tasks, one from each RULER category; the remaining tasks are shown here. The gains are largest on the retrieval-heavy categories and smaller on aggregation, and in all cases they are confined to the length range on which the adjustment was fine-tuned.

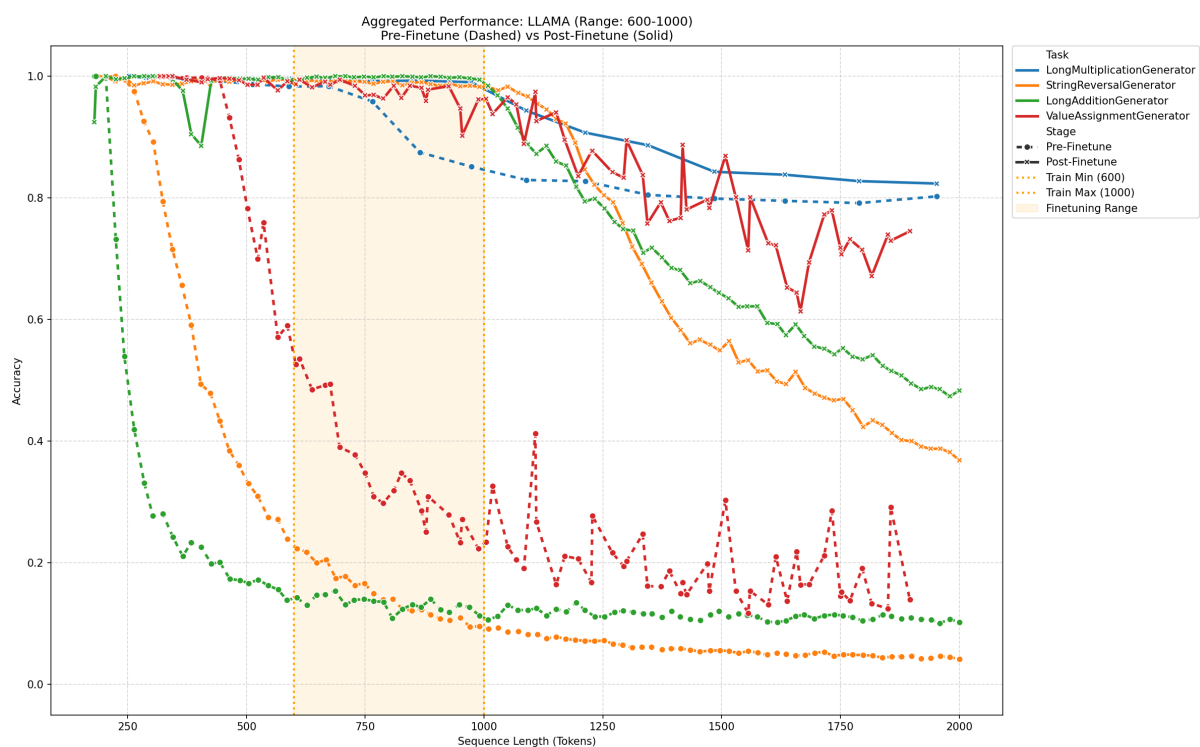


Figure 7: *Llama-3.2-1B-Instruct*, adaptation range 600–1,000 tokens.

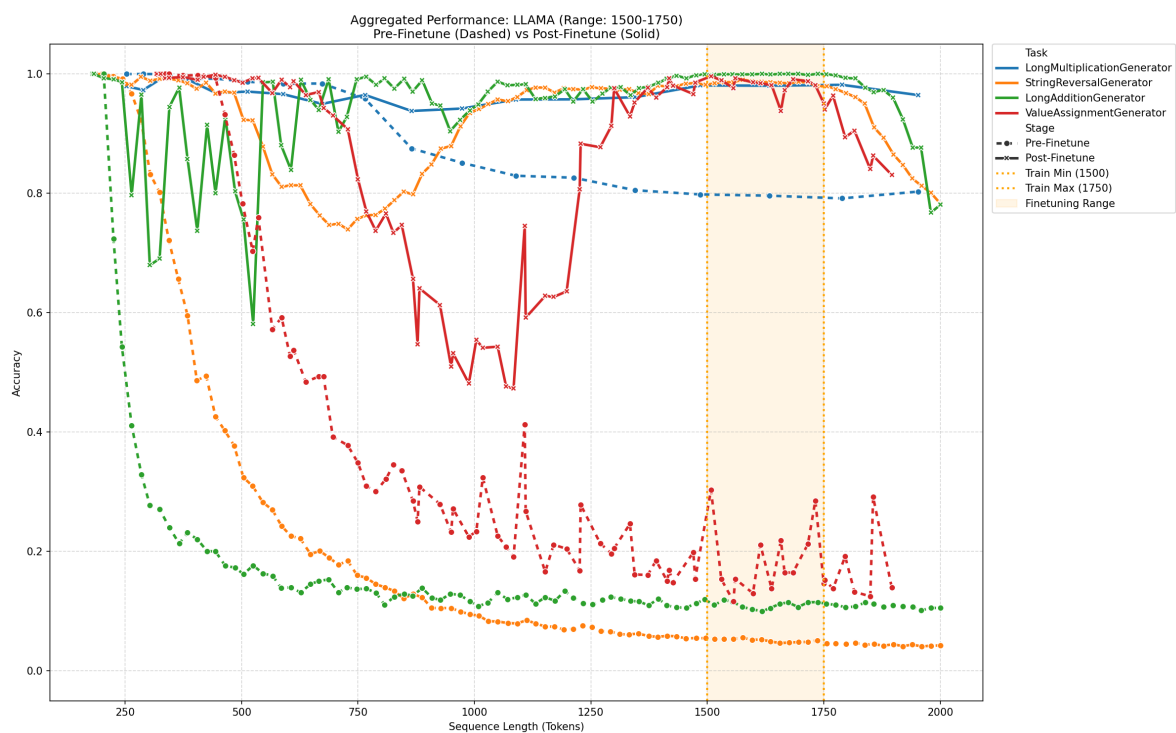


Figure 8: *Llama-3.2-1B-Instruct*, adaptation range 1,500–1,750 tokens.

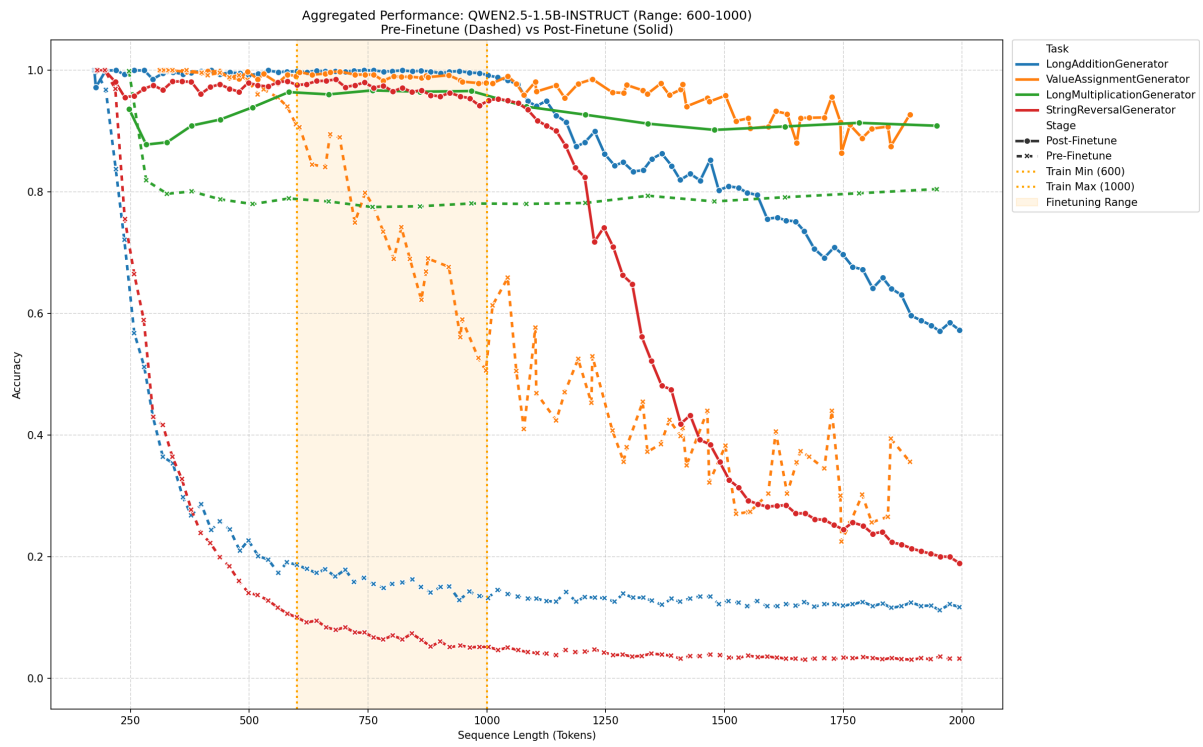


Figure 9: *Qwen2.5-1.5B-Instruct*, adaptation range 600–1,000 tokens.

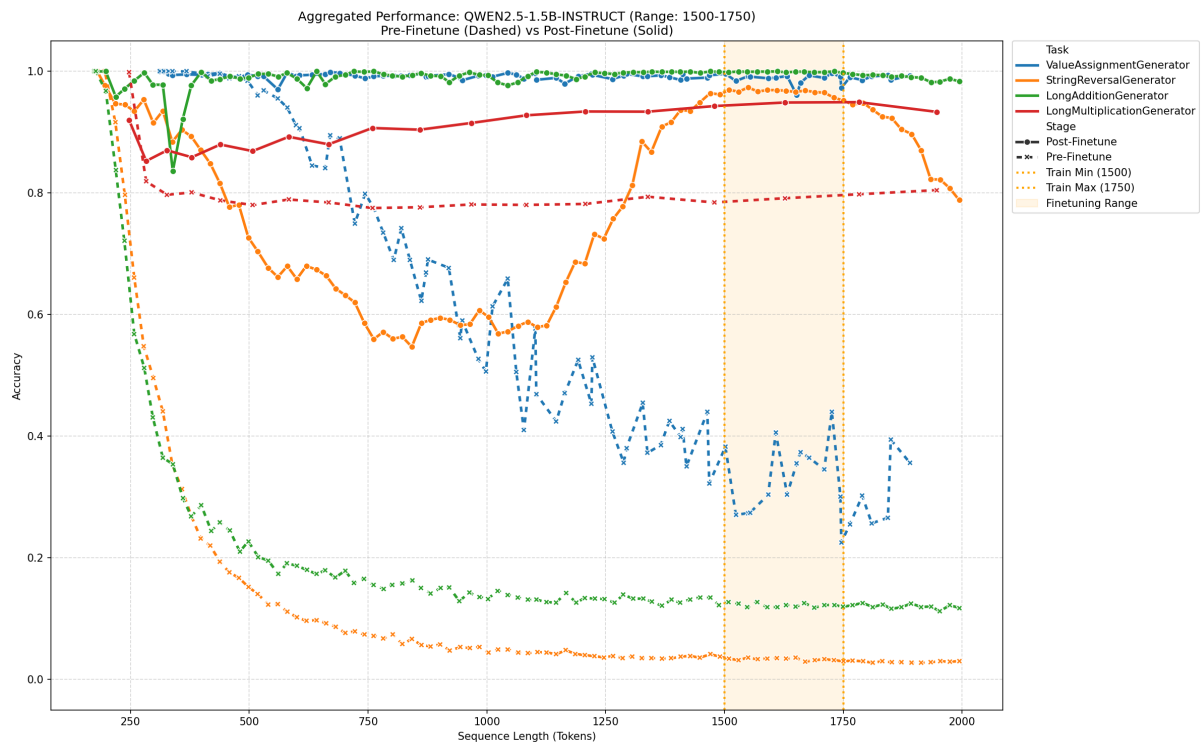


Figure 10: *Qwen2.5-1.5B-Instruct*, adaptation range 1,500–1,750 tokens.

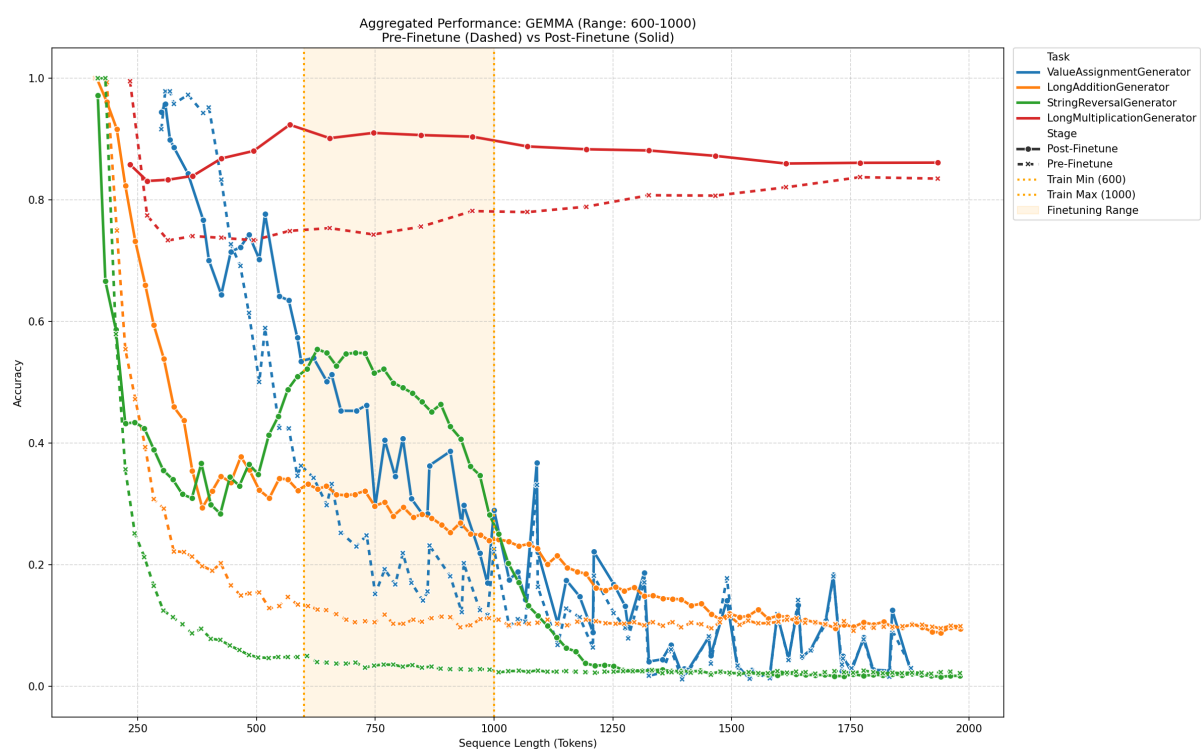


Figure 11: *Gemma-3-1b-it*, adaptation range 600–1,000 tokens. The adaptation prevents collapse but does not reach parity; see Section 6.3.

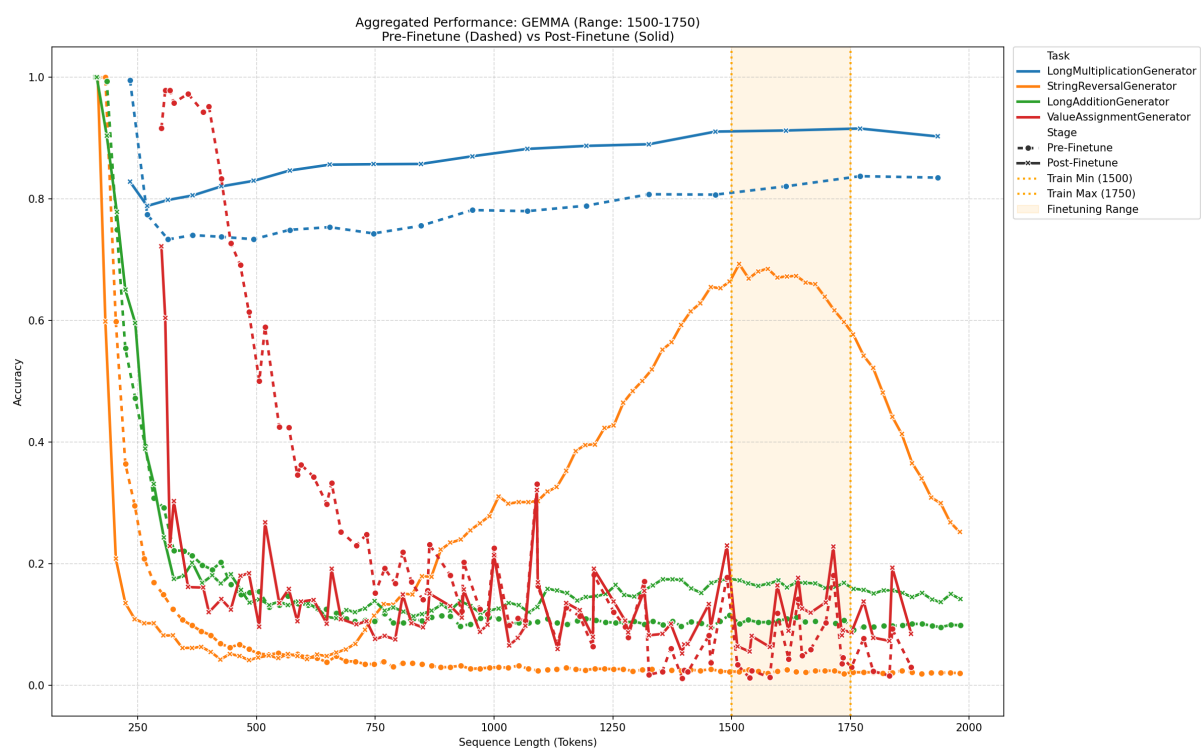


Figure 12: *Gemma-3-1b-it*, adaptation range 1,500–1,750 tokens.

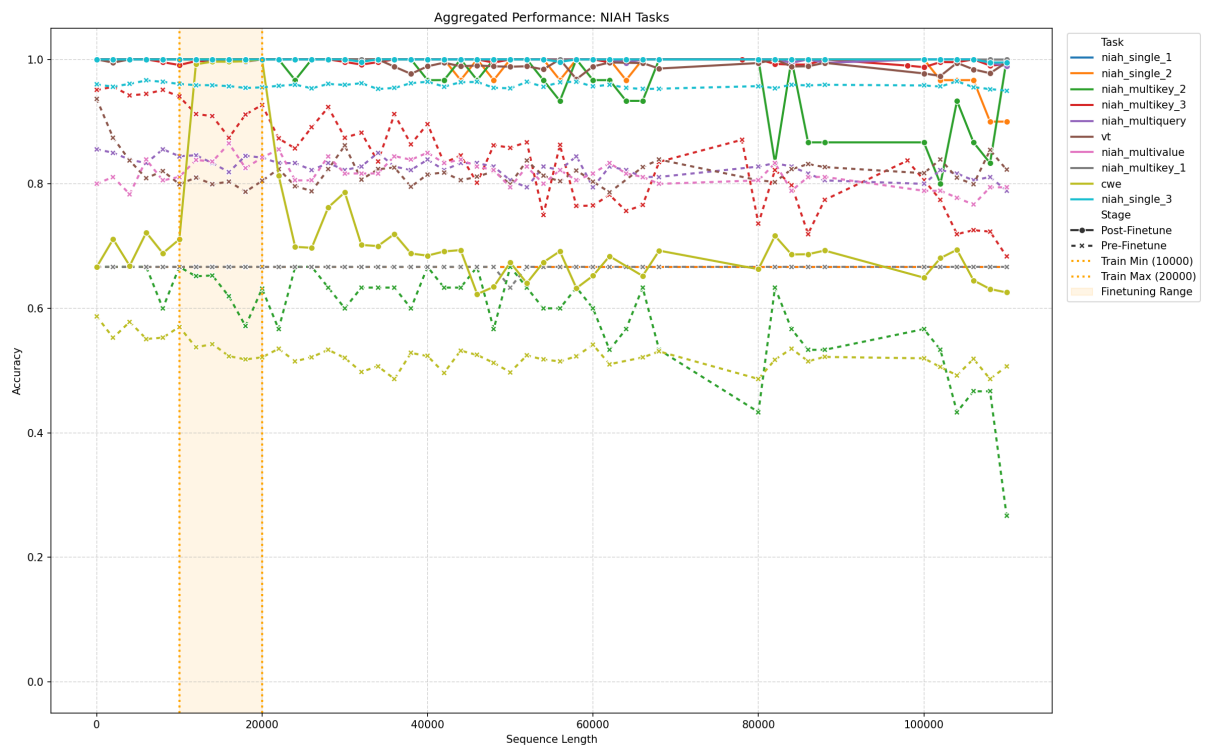


Figure 13: Full RULER evaluation across all 13 tasks for *Llama-3.2-1B-Instruct*, baseline versus minimal W_K/W_Q column adjustment.