Attend or Perish: Benchmarking Attention in Algorithmic Reasoning

Anonymous ACL submission

Abstract

Can transformers learn to perform algorithmic tasks reliably across previously unseen input/output domains? While pre-trained language models show solid accuracy on benchmarks incorporating algorithmic reasoning, assessing the reliability of these results necessitates an ability to distinguish genuine algorithmic understanding from rote memorization. In this paper, we propose AttentionSpan, an algorithmic benchmark comprising five tasks of infinite input domains where we can also disentangle and *trace* the correct, robust algorithm necessary for the task. This allows us to assess (i) models' ability to *extrapolate* to unseen *types* of inputs, including new lengths, value ranges or input domains, but also (ii) to *assess* the robustness of their learned mechanisms. By analyzing attention maps and performing targeted interventions, we causally demonstrate that the attention mechanism is a key bottleneck, directly contributing to failures in extrapolation. We make the implementation of all our tasks and interpretability methods publicly available.[1]

1 Introduction

The neural architecture of Transformer (Vaswani et al., 2017) presents a backbone for a vast majority of modern language processing applications. A growing body of these applications, including code generation, conversational assistants, or data processing automatization, require Transformers to exhibit robust *reasoning*, i.e., an ability to identify and combine relevant pieces of information to infer *new* information (Yu et al., 2024).

The Attention mechanism (Bahdanau et al., 2014) is a critical component for Transformer reasoning, uniquely enabling information mixing across token streams, which is an important process for long-form reasoning. However, its effectiveness often degrades, especially with increasing sequence lengths (Veličković et al., 2025). Despite the theoretical expressivity of Transformers in modelling even complex reasoning tasks (Lin et al., 2021; Merrill and Sabharwal, 2024), Transformers often depend on oversimplified, non-robust, or spurious features of data (Mikula et al., 2024) causing even high-end models to fail in unexpected scenarios. This unreliability currently presents a critical bottleneck across a variety of applications. However, bridging this gap requires fundamental improvements not only in *architecture* (Ye et al., 2025; Veličković et al., 2025) but also *evaluation* to rigorously assess how *robust* is the reasoning process of our models.

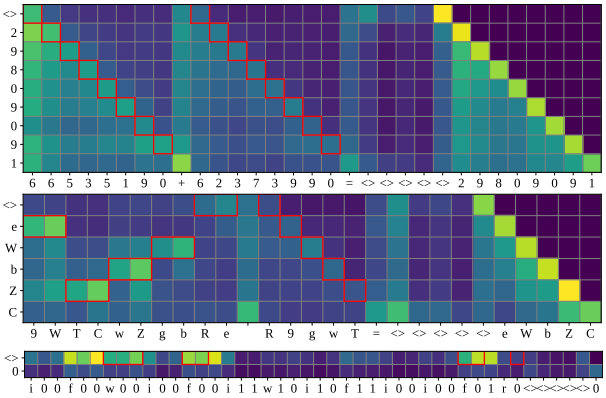

Figure 1: Examples of attention visualizations used to evaluate model reasoning patterns. Each panel displays normalized attention scores (post-softmax), aggregated across heads and layers via attention rollout, overlaid with a reference mask highlighting essential tokens in red. Visualizations are shown for the addition, value assignment, and FFML tasks (top to bottom).

In this work, we contribute to bridging this gap by creating AttentionSpan, a new evaluation suite focused on assessing fundamental reasoning capabilities of language models in out-of-distribution scenarios.

Each task in our benchmark has a solver algorithm, which generates a step-by-step solution and traces *which* past tokens are necessary for correctly generating the next one. This allows us to con-

[1] See the supplementary material

struct a benchmark enabling deeper analyses of the model's reasoning behavior that were not possible with existing resources:

- Provision of **reference attention masks** representing the ground truth reasoning patterns that a successful model has to follow in order to achieve a correct prediction.
- Parametrization of **distribution shifts**, i.e. systematic changes in the constructed dataset that allow for a reliable assessment of the robustness of models' reasoning.

First, we apply our benchmark to evaluate two different facets of generalization of existing language models: (i) an ability to learn to accurately *combine* the information necessary for the task in-distribution, uncovering the models ability to fit an algorithm explaining perturbations of the input samples, and (ii) an ability to *generalize* to out-of-distribution data, exploiting the inherent limitations of existing architectures. We find that Transformer models can learn to robustly execute algorithms on arbitrary in-distribution inputs. Based on our evaluation, models explain ID data by a seemingly correct algorithm, but despite that fail to generalize on new, out-of-distribution (OOD) inputs.

Based on this finding and previous work, we hypothesize that out-of-distribution (OOD) errors stem from attention misalignments. Intervening to align attention with reference masks during OOD inference causally verified this: OOD accuracy increased by up to 90 percentage points (absolute) and, crucially, remained stable across increasing sequence lengths. This establishes a direct causal link between attention failures and poor generalization, pinpointing attention as a key bottleneck for length extrapolation.

The presented benchmark will empower future work by a toolset for deeper analyses of models' internal functioning, cleansed from other covariates such as memorization. Our findings also motivate future work in architectural refinements, particularly those addressing limitations of the current Attention mechanism.

2 Related Work

Closest to our work, CRLS-Text (Markeeva et al., 2024) is a benchmark specialising in algorithmic reasoning implementing many traditional algorithms and trains and evaluates recent state-of-the-art LLMs. We build upon the methodology of CRLS-Text and extend it to allow for, not only accessing the performance, but also to provide means for interpretation and investigation of the results by means of the reference attention masks.

BIG-Bench (Srivastava et al., 2023) is a massive benchmark comprised of more than 200 tasks, many of which specialize in evaluating algorithmic reasoning, e.g. addition or dyck languages. However, as a fixed test set, it is hard to use it to robustly evaluate models on extrapolation, while the recent work finds that BIG-Bench was indeed leaked into the training data of recent models (Fajcik et al., 2024), including Qwen. We extend the tasks from BIG-Bench into configurable generators capable of generating infinite data, allowing training and evaluation while avoiding data contamination.

Flip-Flop Language Modeling is a synthetic task introduced by Liu et al. (2023). Authors introduce this simple algorithmic task to analyze hallucinations caused by attention glitches. We extend this idea and implement novel analysis of attention on a number of diverse algorithmic tasks.

3 AttentionSpan: Dataset and Evaluation Suite

Model	Task	ID Acc.	OOD Acc.	OOD Partial Acc.
Llama-3.2	String Reversal	95.83	53.83	96.18
	Long Addition	96.87	1.61	64.76
	Long Multiplication	86.00	0.00	73.87
	FFML	100.00	99.20	99.79
	Value Assignment	93.95	0.52	65.26
Qwen2.5	String Reversal	98.95	21.77	76.26
	Long Addition	100.00	44.75	92.77
	Long Multiplication	56.25	0.00	80.14
	FFML	100.00	88.50	96.71
	Value Assignment	76.04	1.04	81.06
gemma-3	String Reversal	96.87	6.04	37.64
	Long Addition	100.00	2.62	67.85
	Long Multiplication	89.58	0.00	77.16
	FFML	100.00	90.12	96.71
	Value Assignment	98.95	0.00	19.35

Table 1: Table shows the accuracy of finetuned *Llama-3.2-1B-Instruct*, *Qwen2.5-1.5B-Instruct* and *gemma-3-1b-it* on various tasks with uniformly generated in-distribution and out-of-distribution datasets. We find that while models are able to generalize well even to unseen in-distribution data. Despite a sharp decline in overall OOD accuracy in almost all cases, the OOD partial accuracy (average percentage of correctly predicted target tokens for each data sample) reveals that models correctly predict a large proportion of target tokens, indicating some extrapolation abilities are present.

To evaluate the reasoning robustness of Transformers, we introduce AttentionSpan, a framework for analyzing models' attention patterns in step-by-step reasoning tasks.

AttentionSpan is composed of synthetic tasks with a highly controlled setting such as string reversal, addition or multiplication. See Appendix A for detailed description of each task. Examples of inputs and outputs can be found in Table 3. Task instances (problems) can be randomly generated in arbitrary quantity and with configurable difficulty. The configuration also allows for systematic ID/IID splits that we also apply in our evaluations, including input lengths, ranges or domain. We detail provided configurations of AttentionSpan's tasks in Appendix F.

3.1 Reference Attention Masks

A key contribution of this work is the inclusion of a *reference attention mask* with every generated data sample. This mask precisely identifies the past tokens essential for correct next-token inference. Notably, these reference masks are designed to be independent of any specific model's algorithmic implementation. We utilize these masks as an expected attention pattern for an ideal model and subsequently measure the alignment of a model's learned attention with this reference. Our experiments demonstrate that reference attention masks are a powerful tool for diagnosing reasoning errors in transformers. We propose that they could facilitate future research aimed at enhancing model reliability through architectural modifications.

The reference attention mask is a discrete boolean matrix. For each target token to be predicted, it identifies significant (reference) past tokens that the model should attend to for robust problem-solving. An element in the matrix is set to 1 if the corresponding past token carries information relevant to predicting the target token; otherwise, it is set to 0, indicating the token is irrelevant and should not be attended. We empirically validate our reference masks by showing learned attention reflects these patterns and that interventionally reinforcing these patterns boosts accuracy.

4 Experiments and Evaluation

Using the newly constructed benchmark, we aim to understand to what extent recent language models are capable of robustly representing and executing the underlying algorithms of our tasks. Towards this goal, we fine-tune and evaluate popular *LLama-3.2-1B-Instruct*, *Qwen2.5-1.5B-Instruct* and *gemma-3-1b-it* on all tasks with instruction prompt in a few-shot setting. While models trained from random initialization struggled with convergence and OOD generalization even with extensive data, initializing from pre-trained weights significantly improved convergence efficiency, yielding non-trivial performance with minimal samples. As detailed in Table 1, these fine-tuned models exhibit difficulty generalizing to out-of-distribution (OOD) data (See Appendix F for details on exact ID/OOD parameters). However, their ability to correctly predict a majority of target tokens in each sample (See OOD Partial Acc. in Table 1) suggests underlying length extrapolation capabilities and potential for future improvement. Training setup and hyperparameter search are described in Appendix E.

To inspect which tokens the model considers in each reasoning step, we employ *attention rollout* (Abnar and Zuidema, 2020), a standard method for aggregating attention across heads and layers. This allows us to visualize the attention patterns (e.g., Figure 1) and assess whether the model learned the expected attention patterns crucial for generalization. Quantitatively, we leverage our dataset's reference attention masks to calculate the proportion of attention scores assigned to tokens identified as essential for correct prediction (see Appendix C for details). We particularly analyze these metrics by comparing model performance on correct and incorrect test (token-level) predictions to identify systematic attention patterns associated with errors.

5 Results and Discussion: Attention is the Bottleneck

Several tasks display errors in OOD evaluations that are often associated with a marked reduction in attention on reference tokens (see Table 2 and Appendix B). This pattern suggests a class of errors, where insufficient attention to reference tokens directly contributes to faulty predictions. We find that this phenomena is consistent across different pre-trained models and architectures, implying that it is a general, naturally emergent problem.

5.1 Attention Reinforcement

To causally validate that attention deficits drive out-of-distribution (OOD) failure, we first identified key attention heads (typically <10% of total,

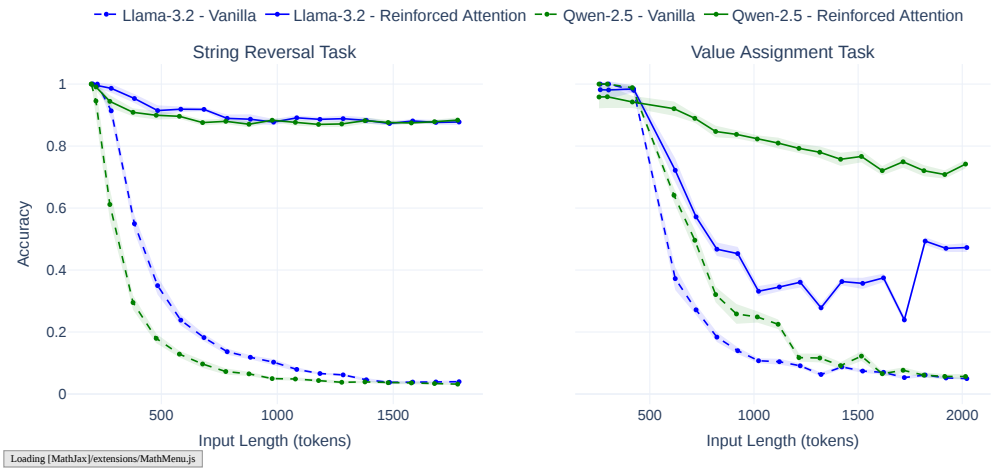

Figure 2: We demonstrate that by intervening on models' (*Llama-3.2-1B-Instruct*, *Qwen2.5-1.5B-Instruct*) activations and directly adjusting their attention scores to reinforce our reference attention pattern, we are able to drastically improve the length extrapolation performance over the vanilla models without invervention. This intervention clearly attributes the failure to extrapolate to the attention mechanism.

Model	Task	OOD Acc.	Attn Score (Correct)	Attn Score (Error)
Llama-3.2	String Reversal	53.83	0.0455	0.0236
	Value Assignment	0.52	0.0333	0.0126
Qwen2.5	String Reversal	21.77	0.0307	0.0217
	Value Assignment	1.04	0.0127	0.0120
gemma-3	String Reversal	6.04	0.0397	0.0230
	Value Assignment	0.00	0.0504	0.0491

Table 2: **Error in prediction largely correlates with lower attention to reference tokens.** Attn Score represents the mean (across samples and all target tokens) proportion of attention scores attributed to the reference tokens (see Appendix C for details). We measure this score separately for cases when the model makes an correct/incorrect (token-level) prediction and find a statistically significant shift, attributing the prediction failures to low attention on reference tokens.

e.g., 3-40 out of 512) by summing their scores on reference tokens during in-distribution inference and selecting those that attended the most to these reference tokens. We then selectively reinforced this reference pattern during OOD inference by directly increasing these heads' post-softmax attention on reference tokens. This intervention (Figure 2) yielded up to a 90 percentage point (absolute) increase in OOD accuracy, providing direct **causal evidence that insufficient attention to reference tokens contributes to extrapolation failures**. For details on the implementation of the intervention, see Appendix G.

While our intervention targets attention outputs, the root cause of out-of-distribution (OOD) attention deficits may stem from deeper computational issues, such as faltering key-query interactions under distributional shift. Notably, we observed misallocated attention shifting to distant, rather than neighboring, tokens in OOD scenarios. This suggests a potential breakdown in the generalization of positional embeddings like RoPE. Future work should investigate RoPE's behavior under such shifts and explore integrating these insights into model training for more robust attention and positional encoding.

6 Conclusion

We introduced a novel algorithmic benchmark with reference attention masks to robustly assess Transformer extrapolation and reasoning, minimizing memorization effects. Our evaluations reveal that while pre-training helps, models struggle with out-of-distribution (OOD) generalization on these challenging tasks. Crucially, through attention analysis and targeted interventions, we causally attribute these OOD failures primarily to the attention mechanism's inability to maintain focus on essential tokens. Reinforcing correct attention patterns significantly improved OOD accuracy and length extrapolation, pinpointing attention as a critical bottleneck. This work underscores the difficulty these tasks pose for current architectures and highlights the need for more robust attention mechanisms. By releasing our benchmark and findings, we aim to spur research towards more robust and reliable AI systems.

Limitations

We identify several limitations of our work and mention what we believe are the main ones below. First, our interpretability of models' internal functioning builds upon the assumption that models robustly executing the correct algorithm should fully attend only to tokens that are relevant to the algorithm. Nevertheless, we note that even a model with a systematic dispersion of attention across irrelevant tokens might still be able to robustly execute algorithm, as long as the irrelevant attended tokens do not significantly alter the attention's output representations. Therefore, there is not a necessary equivalence between the model's robustness and accuracy of attention with respect to our references. However, in the situation where the model does not attend the relevant tokens at all, we can still claim that the model does not represent the task's correct/robust algorithm.

Finally, we note the limitation in using a single interpretability method in our analyses in Section 4 (Attention rollout). While we argue that this method best represents the computation flow within the transformer across tokens, it still does not take into account some computation parts of the model, such as the impact of feed-forward layers which might, theoretically, exclude the impact of even some attended tokens.

References

Samira Abnar and Willem Zuidema. 2020. Quantifying attention flow in transformers.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural Machine Translation by Jointly Learning to Align and Translate. ArXiv:1409.0473v1.

Martin Fajcik, Martin Docekal, Jan Dolezal, Karel Ondrej, Karel Beneš, Jan Kapsa, Pavel Smrz, Alexander Polok, Michal Hradis, Zuzana Neverilova, et al. 2024. Benczechmark: A czech-centric multitask and multimetric benchmark for large language models with duel scoring mechanism. *arXiv preprint arXiv:2412.17933*.

Chu-Cheng Lin, Aaron Jaech, Xin Li, Matthew R. Gormley, and Jason Eisner. 2021. Limitations of autoregressive models and their alternatives. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5147–5173, Online. Association for Computational Linguistics.

Bingbin Liu, Jordan T. Ash, Surbhi Goel, Akshay Krishnamurthy, and Cyril Zhang. 2023. Exposing attention glitches with flip-flop language modeling.

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.

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

Lukáš Mikula, Michal Štefánik, Marek Petrovič, and Petr Sojka. 2024. Think Twice: Measuring the Efficiency of Eliminating Prediction Shortcuts of Question Answering Models. In *Proceedings of the 18th Conference of the European Chapter of the ACL (Volume 1: Long Papers)*, pages 2179–2193, St. Julian's, Malta. ACL.

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.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

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, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-Art Natural Language Processing. In *Proc. of the 2020 Conf. EMNLP: System Demonstrations*, pages 38–45. ACL.

Tianzhu Ye, Li Dong, Yuqing Xia, Yutao Sun, Yi Zhu, Gao Huang, and Furu Wei. 2025. Differential transformer. In *The Thirteenth International Conference on Learning Representations*.

Fei Yu, Hongbo Zhang, Prayag Tiwari, and Benyou Wang. 2024. Natural language reasoning, a survey. *ACM Comput. Surv.*

A Task Descriptions

A.1 String Reversal

This task requires the model to generate the input sequence in the reverse order. The task generator

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 3: Example instances of our tasks. The spacing is adjusted for clarity and does not denote a separator of tokens. How the tasks handle tokenization is described in greater detail in Appendix D

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.

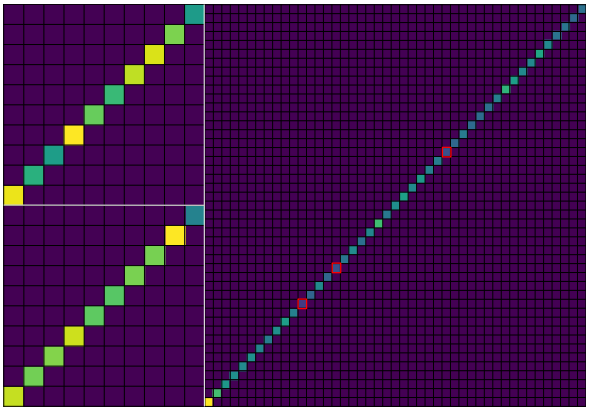

Figure 3: In String Reversal, the model must learn a diagonal attention pattern. In ID evaluation (left), the model attributes high scores to all reference tokens. In OOD (right), it fails to do so for some tokens (highlighted in red), leading to prediction errors.

B OOD Evaluation of String Reversal

Figure 3 illustrates the correlation between low attention on reference tokens and prediction errors.

C 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 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.

D 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.

E 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.

F OOD Evaluation

F.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

F.2 FFML Task Evaluation Parameters

The following configuration details the evaluation setup for the FFML 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

F.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

F.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

F.5 Successor Task Evaluation Parameters

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

In-distribution:

- The starting number is between 1 and 90
- The length of the series is 2-4 numbers

Out-of-distribution:

- The starting number is between 100 and 900
- The length of the series is 5-6 numbers

F.6 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

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

G Attention Intervention Details

Our intervention method aims to causally link attention deficits to out-of-distribution (OOD) performance degradation by selectively reinforcing attention to reference tokens. The process involves two main stages: identifying key attention heads and applying the intervention.

1. Identifying Key Attention Heads: To pinpoint the attention heads most responsible for implementing the desired reference attention pattern, we perform the following steps:
   - We run inference on multiple in-distribution (ID) data samples (typically 30 in our experiments).
   - For each attention head, we calculate the sum of its post-softmax attention scores on the pre-defined reference tokens. This sum is accumulated across all ID samples.
   - This cumulative scoring helps identify heads that consistently attend to reference tokens, as well as those that might activate for specific patterns present only in a subset of samples (e.g., particular carry operations in addition tasks).
   - Heads are then ranked in descending order based on this cumulativ e score.
   - We select the top N heads for intervention. N is a hyperparameter optimized to achieve significant performance improvement on the end-to-end task (e.g., string reversal) post-intervention.
2. Applying the Intervention during OOD Inference: The intervention is applied exclusively to the N selected heads during OOD inference.
   - Standard Intervention (e.g., for String Reversal): For each selected head, we directly modify its post-softmax attention scores. A constant value C (a hyperparameter, typically ranging from 0.3 to 2.0) is added to the attention score of every token position corresponding to a reference token. These modified attention scores are then propagated through the network. This approach proved effective for tasks like string reversal.
   - Conditional Intervention (e.g., for Value Assignment): For more complex tasks

like value assignment, we observed that the reference attention pattern was often distributed across multiple heads, and a simple global reinforcement was ineffective. Instead, we adopted a conditional reinforcement strategy:

- For each selected head, we add the constant C to the post-softmax attention score at a reference token position only if the original attention score at that specific position already exceeds a certain threshold (another optimizable hyperparameter).
- This approach reinforces existing, albeit potentially weak, attention subpatterns within a head, rather than imposing the entire reference pattern uniformly.
- The conditional intervention for value assignment, while improving performance, sometimes results in a slightly lower accuracy boost compared to the standard intervention on simpler tasks. This is because if the initial activation for a crucial reference token falls below the threshold, our intervention, by design, will not reinforce it, even if doing so would be beneficial.