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 cleanse models' functional capabilities from memorization. In this paper, we propose an algorithmic benchmark comprising six 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 the functional mechanism in recent models through the lens of their attention maps. We make the implementation of all our tasks and interoperability methods publicly available.[1]

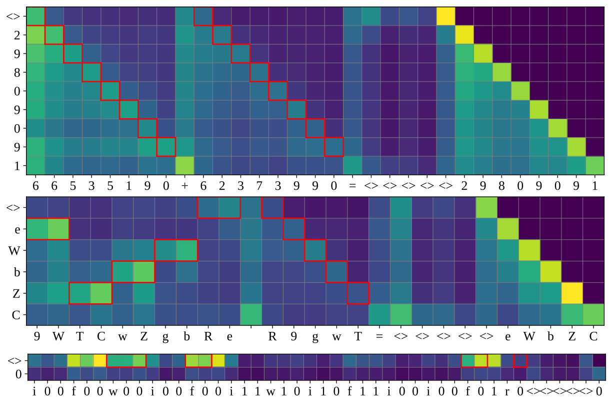

Figure 1: Examples of reference attention maps we use to evaluate the models' attention reasoning patterns. From top to bottom, you can see the attention scores with reference tokens highlighted in red, for addition, value assignment and FFLM task.

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).

A critical component of the Transformer's reasoning process is the Attention mechanism (Bahdanau et al., 2014), which derives the representation of each newly generated token by *weighing* the representations of previous tokens.

Despite the theoretical expressivity of Transformers in modelling even complex reasoning tasks (Lin et al., 2021; Merrill and Sabharwal, 2024), Transformers often depend on over-simplified, 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. 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.

In this work, we contribute to bridging this gap by creating a new evaluation suite that presents a collection of diverse reasoning tasks.

Each task 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 construct *reference attention maps* representing the ground truth reasoning patterns a successful model has to exhibit and compare it to the models' actual attention map. We find that this comparison exposes some aspects of the model's internal reasoning process, is highly predictive of the models' reliability, and can identify sources of errors in the models' reasoning process.

We apply our benchmark to assess two differ-

[1] See the supplementary materials.

ent facets of generalization of existing language models: (i) existing models' ability to robustly *execute* the task given a clear instruction of the task, uncovering the robustness of models' instruction-following capabilities, and (ii) models' ability to *learn to generalize* our with *unlimited* data, underlying the inherent limitations of existing architectures.

We find that current state-of-the-art models are able to learn to robustly execute algorithms on arbitrary in-distribution inputs. We also find that the errors that models exhibit in-distribution (ID) are not connected with not attending significant tokens.

However, when evaluating models on out-of-distribution data (OOD), especially longer inputs, models struggle to apply the reasoning attention pattern they learned in ID and make prediction errors rooted in not attending the correct tokens.

Our benchmark will empower future work in improving language models to not only assess the empirical improvements on our benchmarks but also to understand the implications of different architectural refinements on the robustness of models' internal functioning cleansed from other covariates such as memorization.

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 maps.

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

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. 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 E.

Every problem has a single unambiguous solution, consisting of a deterministic sequence of steps that can be verified algorithmically.

A key contribution of our work is that every solution includes a *reference attention mask* that exactly specifies which past tokens are needed for correctly inferring the next one. It is important to mention that the reference maps are constructed in such way that they are independent of how the model implements the given algorithm. The indicated reference token are always crucial to completing the task. As we demonstrate in our experiments, reference attention masks are a powerful tool for inspecting the errors of transformers' reasoning. We argue that they might facilitate future work in improving model reliability via architectural adjustments.

In the remainder of the Section, we describe the tasks in our suite. Examples of inputs and outputs can be found in Table 1.

3.1 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.

3.2 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.

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
Successor	234	235 236 237 238 239 240

Table 1: 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 C

3.3 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.

3.4 Successor

The Successor task requires a model to generate a sequence of natural numbers starting from a given initial value. It can be parametrized by the length of the series and the allowed range of the starting value. This is a straightforward task requiring precise representations of how digits form natural numbers.

3.5 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.

3.6 Flip Flop Language Modeling

Flip Flop Language Modeling, as introduced by Liu et al. 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.

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 train and evaluate a popular *LLama-3.2-1B-Instruct* model on all tasks in two settings. First, we fine-tune the trained model with instruction in a few-shot setting. Secondly, we train the same architecture from scratch without instructions or few-shot examples. Differences in evaluations between the two variants can be attributed to the pre-training and instruction fine-tuning. Training setup and hyperparameter search are described in Appendix D.

We evaluate the models' accuracy separately on ID and OOD data to measure their robustness to changes in problem length (See Appendix E). Next, we relate these results to the model's ability to *focus on relevant tokens*, as provided by our dataset. To inspect which tokens the model considers in each reasoning step, we employ *attention rollout* (Abnar and Zuidema, 2020), a method for aggregating all models intermediate attention maps.

Using this aggregated attention map, we compute the proportion of attention scores allocated to tokens that our reference attention map identifies as necessary for correct prediction (See details in Appendix B). In particular, we measure and compare these metrics on the test instances where the model makes correct and erroneous predictions to uncover a possible pattern across error cases in the attention scores.

Model Type	Task	ID		OOD	
		Acc.	Attn Score	Acc.	Attn Score
From Scratch	String Reversal	5.21	0.0578	0	0.0248
	Long Addition	9.37	0.1713	0	0.0891
	Long Multiplication	18	0.1302	0	0.0827
	FFML	68.75	0.0129	50.2	0:0047
	Value Assignment	4.17	0.3060	0	0.0683
	Successor	100	0.4069	0	0.1450
Finetuned	String Reversal	95.83	0.0836	53.83	0.0448
	Long Addition	96.87	0.1380	1.61	0.0779
	Long Multiplication	86	0.0432	0	0.0257
	FFML	100	0.1854	99.2	0.1461
	Value Assignment	100	0.1668	0	0.0378
	Successor	100	0.4425	65.73	0.3770

Table 2: Performance of models trained from scratch and finetuned on various tasks. The changes in the mean Attn Score between ID and OOD are statistically significant in all cases.

5 Results and Discusssion

Task	OOD Acc.	Attn Score (Correct)	Attn Score (Error)
String Reversal	53.83	0.0455	0.0236
Successor	65.73	0.4141	0.3685
FFML	99.2	0.1948	0.3829

Table 3: Error in prediction significantly correlates with low attention score on reference tokens. The change in mean Attn Score is statistically significant in all cases.

Table 2 shows that models trained from scratch struggle with convergence and rarely generalize to OOD data, even with large sample sizes.

In contrast, an initialization from a pre-trained model dramatically improves the efficiency of convergence, achieving non-trivial performance already after using just tens or hundreds of samples. To a limited extent, resulting models *can* generalize to OOD data, even though the resulting accuracy consistently falls behind the ID performance.

We further analyze the distribution of attention scores on reference tokens. Welch's t-test confirms a significant difference between ID and OOD data, with average attention scores dropping on OOD inputs. This may be due to longer sequences dispersing attention (Veličković et al., 2025) or an inability to reliably identify key tokens.

Finally, in tasks where models perform well on in-distribution data, errors in OOD evaluations are often associated with a marked reduction in attention on reference tokens (see Table 3 and Appendix A). This pattern suggests one class of error, where insufficient attention directly contributes to faulty predictions. By contrast, tasks such as FFML show stable or even increased attention scores during errors, implying that different error mechanisms are at work. Importantly, because the OOD evaluation is not compromised by sequence length effects, the significant changes in attention scores for some tasks clearly reflect distinct classes of inference problems.

6 Conclusion

In this work, we introduced a novel algorithmic benchmark designed to assess both the extrapolation and reasoning capabilities of Transformer-based models robustly to memorization. Our evaluation framework, which leverages configurable tasks and reference attention maps, provides a transparent and fine-grained analysis of models' internal reasoning processes beyond traditional accuracy metrics. This allows us to show that models trained from scratch struggle with generalization while pre-training projects into significant and consistent improvements on out-of-distribution inputs.

Importantly, our attention-based evaluation revealed that errors in reasoning are often associated with a dispersion of attention to relevant tokens. This finding not only validates our approach for diagnosing internal model behaviors but also lays the foundation for future architectural refinements aimed at enhancing robustness. By releasing our benchmark and methodologies, we hope to foster further research into the reliability and interpretability of Transformer models, ultimately contributing to the development of much-needed, robust AI systems for dynamic, real-world applications.

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.

Further, we acknowledge our focus only on a single model architecture as a limitation of our analyses. While at the time of writing, Llama model family represents a state-of-the-art among open-source models, we note that some trends, e.g. *which tasks can/can not be learned* can still be model-specific. Nevertheless, we focus our contribution instead to broadening a set of tasks and assuring a reliability of attention labels, leaving the investigation of further models to future work.

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.

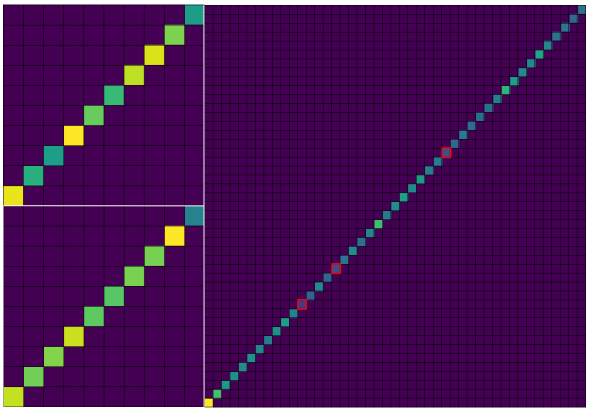

Figure 2: In String Reversal, the model must learn a diagonal attention pattern. In ID evaluation (left), the model attributes high scores to all reference tokens. In OOD (right), it fails to do so for some tokens (highlighted in red), leading to prediction errors.

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 OOD Evaluation of String Reversal

B 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.

C 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 maps. 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.

D 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.

E OOD Evaluation

E.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

E.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

E.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

E.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

E.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

E.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