Transformers meet Neural Algorithmic Reasoners

Wilfried Bounsi
Google DeepMind
wilcoln@google.com

Borja Ibarz
Google DeepMind
bibarz@google.com

Andrew Dudzik
Google DeepMind
adudzik@google.com

Jessica B. Hamrick
Google DeepMind
jhamrick@google.com

Larisa Markeeva
Google DeepMind
lmarkeeva@google.com

Alex Vitvitskyi
Google DeepMind
avlife@google.com

Razvan Pascanu
Google DeepMind
razp@google.com

Petar Veličković
Google DeepMind
petarv@google.com

Abstract

Transformers have revolutionized machine learning with their simple yet effective architecture. Pre-training Transformers on massive text datasets from the Internet has led to unmatched generalization for natural language understanding (NLU) tasks. However, such language models remain fragile when tasked with algorithmic forms of reasoning, where computations must be precise and robust. To address this limitation, we propose a novel approach that combines the Transformer's language understanding with the robustness of graph neural network (GNN)-based neural algorithmic reasoners (NARs). Such NARs proved effective as generic solvers for algorithmic tasks, when specified in graph form. To make their embeddings accessible to a Transformer, we propose a hybrid architecture with a two-phase training procedure, allowing the tokens in the language model to cross-attend to the node embeddings from the NAR. We evaluate our resulting TransNAR model on the text-based version of the CLRS-30 benchmark, demonstrating significant gains over Transformer-only models for algorithmic reasoning, both in and out of distribution.

1. Introduction

Recent work motivated [8] and showcased [6, 14] the effectiveness of graph neural networks [30, GNNs] at robustly solving algorithmic tasks of various input sizes, both in and out of distribution—such systems are often referred to as *neural algorithmic reasoners* [33, NARs]. Provided appropriate inductive biases are used, NARs are capable of holding perfect generalisation even on 6× larger inputs than ones seen in the training set, for highly complex algorithmic tasks with long rollouts [16]. NARs are, however, still relatively *narrow* forms of AI, as they require rigidly structured formatting of inputs, and they hence cannot be directly applied to problems posed in more noisy forms—such as

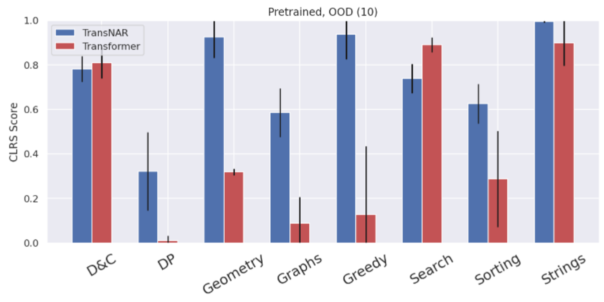

Figure 1. Our TransNAR architecture, with its direct synergy of Transformers and Neural Algorithmic Reasoners, yields clear improvements in out-of-distribution reasoning across wide categories of algorithmic tasks in a textual version of the CLRS-30 benchmark [34]. Here, the x-axis indicates one of the eight algorithmic families of CLRS-30, and the y-axis spans the average execution accuracy across a dataset of out-of-distribution examples. TransNAR enables *emerging capabilities* in the particular out-of-distribution regime depicted here, with over 20% absolute improvement in several of the algorithm classes.

in *natural language*—even when the underlying problem is still algorithmic in nature.

Conversely, the current undisputed state-of-the-art approach for modelling noisy text data are Transformer-based [29] language models [2, 5]. In spite of their unrivalled natural language understanding properties, they are also notoriously brittle when faced with even the simplest algorithmic tasks [9]—especially if out-of-distribution generalisation is required [4].

It appears that ***uniting*** *Transformers with NARs* can lead to fruitful returns on both sides. In this paper, we explore this interface for the first time, building the **TransNAR** model.

Contributions Our exploration proved fruitful. The key takeaways we present in this work are as follows:

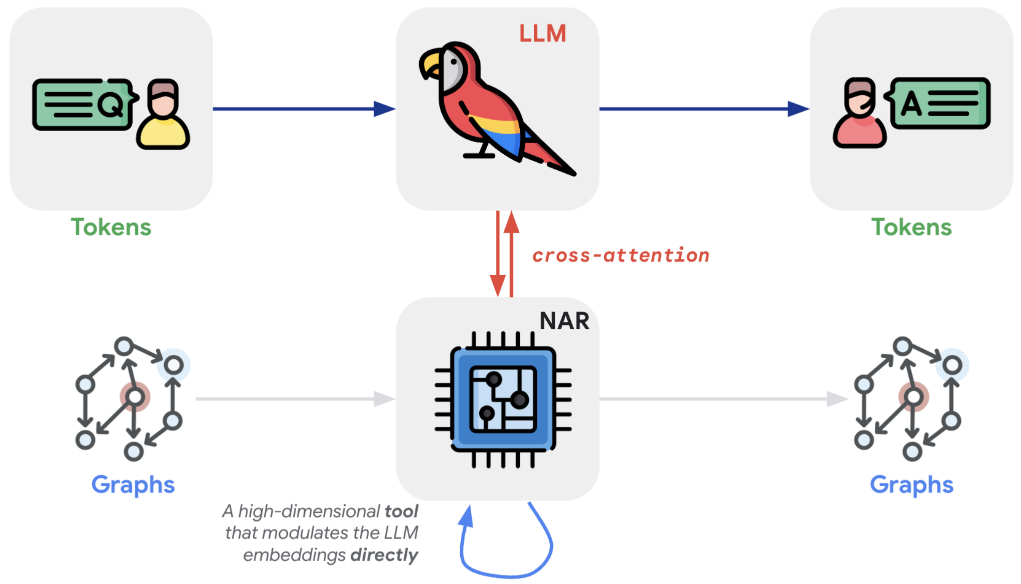
LLM
Tokens
Tokens
cross-attention
NAR
Graphs
Graphs
A high-dimensional tool that modulates the LLM embeddings directly

Figure 2. **Augmenting LLMs with algorithmic reasoning: a bird's eye view of TransNAR.** A large language model (LLM) consumes input tokens and produces output tokens, as common for a unimodal Transformer. The neural algorithmic reasoner (NAR) module is a graph neural network (GNN) pre-trained to execute various algorithmic computation on a collection of graph-based inputs [14]—the pre-training pipeline is denoted by faded arrows. Throughout its forward pass, the Transformer may access the embeddings computed by the NAR, by leveraging cross-attention (trained by learnable "glue" weights).

1. We propose a hybrid architecture combining the language understanding of a Transformer with the robustness of reasoning of a pre-trained GNN-based NAR. The Transformer uses the NAR as a *high-dimensional tool* that will modulate its tokens embeddings.
2. We show, through an evaluation on a textual version of the CLRS-30 benchmark [34], that such a NAR-augmented large language model (LLM) exhibits improved and more robust reasoning capabilities out-of-distribution (Figure 1).

Our work presents one of the most comprehensive size generalisation challenges given to Transformers to date, and the introduction of NARs moves the needle significantly.

2. Related work

Our work sits at the intersection of several areas: neural algorithmic reasoning, length generalisation in language models, tool use, and multimodality. Here, we briefly survey various relevant works in each area. Due to the diversity of perspectives, to preserve brevity, we do not offer a comprehensive review of related work, but rather aim to provide an indication of specific works that inspired ours the most.

Neural algorithmic reasoning NAR is, in general terms, the art of building neural networks that are capable of capturing algorithmic computation. Such capabilities can be amplified by careful choices in algorithmic alignment [37], step-wise training [31] or contrastive objectives [6].

Recently, it was demonstrated that: (1) it is possible to learn an NAR capable of executing *multiple* algorithms simultaneously in its latent space [36]—with the Triplet-GMPNN [14] skillfully doing so for a collection of thirty algorithms across the CLRS benchmark [34]; (2) Once trained, such NARs can be usefully deployed in various downstream tasks: reinforcement learning [7, 12], self-supervised learning [32], combinatorial optimisation [10, 22], computational biology [11] and neuroscience [20].

Our work's use of NAR is mostly motivated by two of the works listed before: we use a relatively small, pre-trained, multi-task NAR [14], and deploy it in a far more scaled environment: as shown by Numeroso et al. [20], NAR should in principle be scalable to systems that are orders-of-magnitude greater than the NAR's training distribution ($180,000\times$ in that particular case).

Length generalisation in LLMs While NARs can often strongly generalise to far greater test inputs [16], LLMs have seen significantly less success in such scenarios. We attribute this to their autoregressive, causally-masked objective, which may not always correspond to the most logical order in which outputs of algorithms should be predicted. Just as a simple example, performance of various LLMs on

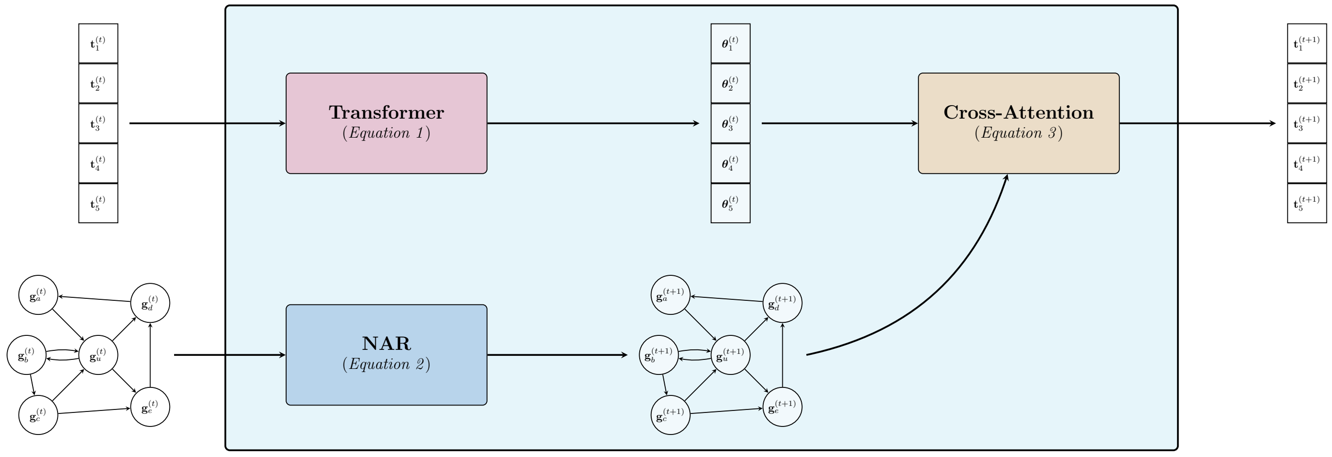

Figure 3. **TransNAR hybrid architecture.** Similar to Alayrac et al. [3], we interleave existing Transformer layers with gated cross-attention layers which enable information to flow from the NAR to the Transformer. We generate queries from tokens while we obtain keys and values from nodes and edges of the graph. The node and edge embeddings are obtained by running the NAR on the graph version of the reasoning task to be solved. When experimenting with pre-trained Transformers, we initially close the cross-attention gate, in order to fully preserve the language model's internal knowledge at the beginning of training.

multiplication can be significantly improved by predicting the result in reverse order [19]. Of course, on more complicated algorithms, it may be much harder to determine the best way to permute the input, and it may not be the most human-readable.

Knowledge of the above issues has led to a significant amount of effort being invested in building Transformers that can generalise in length. While length generalisation is not the only kind of distribution shift of interest to OOD reasoning, it is among the most easy such shifts to simulate. Accordingly, various works have attempted to induce length generalisation in LLMs, through the use of careful prompting [27, 38], randomised positional encoding [25], curricula [1] or scratchpads [4]. We firmly believe that an important trait of reasoning is robustness with respect to prompt quality—so long as the prompt unambiguously specifies the problem—and hence deliberately do not explore prompt modification approaches here; only randomised positions [25] are leveraged out of the works above in our model.

Tool use and multimodality Another way to obtain robust generalisation performance is to leverage a hard-coded algorithm (also known as a *tool*) by teaching an LLM to invoke its API [26]. Arguably, most of the major successes of reasoning with LLMs [18, 24, 28] can primarily be attributed to an LLM's clever usage of a tool rather than the LLM itself, as a tool will by definition not have issues in generalising to diverse inputs.

Since our aim is to directly evaluate reasoning capabilities of LLMs, we explicitly do not permit tool use in our baselines. That being said, we envision the pre-trained NAR as a *modulator* for the Transformer's embeddings which is more robust to OOD noise. Hence, we may observe the NAR as an *"internal tool"*: rather than using raw tokens, the Transformer and NAR can communicate using their embeddings, breaking the associated algorithmic bottlenecks [7, 21].

How to actually realise this communication and embedding exchange? For this, we turn to *multimodal LLMs* [15] for inspiration, since we need to integrate signals coming from two different representations of algorithmic problems (text and graph). Specifically, our exchange operator is directly inspired by vision language models (VLMs) and the cross-attention operator used in Flamingo [3], which offered a principled way of fusing information from text and image modalities.

3. TransNAR: Augmenting Transformers with a pre-trained GNN-based NAR

This section describes our hybrid TransNAR architecture (refer to Figure 3). TransNAR accepts a dual input consisting of a textual algorithmic problem specification (of T tokens) and its corresponding CLRS-30-specific graph representation (of N nodes) and outputs a textual response to the problem. We can assume that, once encoded, the textual input is stored in $\mathbf{T} \in \mathbb{R}^{T\times k}$, and the graph input is stored in $\mathbf{G} \in \mathbb{R}^{N\times l}$. Note that, for simplifying the equations to follow, we make an assumption that all of the information relevant to the graph version of the problem is stored in the nodes—which is often not true in CLRS-30 (there may be edge- and graph-level inputs as well) but it doesn't change the underlying dataflow presented below.

The forward pass of TransNAR unfolds as follows. First, we properly initialise the inputs by setting $\mathbf{T}^{(0)} = \mathbf{T}$ and $\mathbf{G}^{(0)} = \mathbf{G}$. Next, to compute the representation of a step $(t+1)$, the text (token) representations are fed to the current layer of the Transformer [29]:

$$\mathbf{\Theta}^{(t+1)} = \mathrm{FFN}\left(\mathrm{softmax}\left(\frac{(\mathbf{T}^{(t)}\mathbf{Q}_t)^\top \mathbf{T}^{(t)}\mathbf{K}_t}{\sqrt{d_k}}\right)\mathbf{T}^{(t)}\mathbf{V}_t\right) \tag{1}$$

where $\mathbf{Q}_t, \mathbf{K}_t \in \mathbb{R}^{k\times d_k}, \mathbf{V}_t \in \mathbb{R}^{k\times k}$ are the key, query and value transformations, respectively, and FFN is a feed-forward network. In a similar manner, the graph representations are fed to the NAR layer, implementing e.g. a standard max-MPNN [31]:

$$\mathbf{g}_u^{(t+1)} = \phi\left(\mathbf{g}_u^{(t)}, \max_{1\leq v\leq N} \psi\left(\mathbf{g}_u^{(t)}, \mathbf{g}_v^{(t)}\right)\right) \tag{2}$$

where $\psi, \phi : \mathbb{R}^k \times \mathbb{R}^k \rightarrow \mathbb{R}^k$ are learnable *message* and *update* functions, respectively, and $\max$ is the elementwise-max aggregation. Note that Equation 2 only provides pairwise interactions between nodes for brevity—in reality, our NAR is a Triplet-GMPNN [14], which also contains triplet interactions and a gating mechanism. Further, note that there is no timestep index on the learnable parts of the NAR—at each step, a *shared* function is applied. This aligns well with the iterative, repeated nature of algorithmic computation on graphs.

Once both streams have prepared their representations, $\mathbf{\Theta}^{(t+1)}$ and $\mathbf{G}^{(t+1)}$, the node embeddings in the graph condition the Transformer's token embeddings to produce the final outcome of the TransNAR block in the Transformer stream, inspired by Flamingo [3]:

$$\mathbf{T}^{(t+1)} = \mathrm{FFN}\left(\mathrm{softmax}\left(\frac{(\mathbf{\Theta}^{(t)}\mathbf{Q}_t^\times)^\top \mathbf{G}^{(t)}\mathbf{K}_t^\times}{\sqrt{d_k}}\right)\mathbf{G}^{(t)}\mathbf{V}_t^\times\right) \tag{3}$$

where $\mathbf{Q}_t^\times, \mathbf{K}_t^\times \in \mathbb{R}^{k\times d_k}, \mathbf{V}_t^\times \in \mathbb{R}^{k\times k}$ are the key, query and value transformations of the cross-attention, respectively. No additional transformations are performed on $\mathbf{G}^{(t+1)}$ before concluding this layer.

This process repeats until the final, N_l-th layer, when the final text output is read out from $\mathbf{T}^{(N_l)}$. The final output is converted into token logits by a prediction head produced by the final layer, which we supervise by means of a standard next-token prediction objective.

Prior to the start of TransNAR fine-tuning, we pre-train the NAR to robustly execute the thirty algorithms spanned by CLRS-30 [34], in a manner similar to Ibarz et al. [14]. Such procedures are known to yield out-of-distribution generalisation at up-to-$4\times$ larger inputs in graph space. The parameters of the NAR are generally kept *frozen* during fine-tuning, as additional gradients would eliminate the model's original robustness properties. This is also, similarly, the reason why no cross-attention is performed by the graph embeddings. The LLM itself may be pre-trained over large-scale datasets [13], to establish its general language priors, though we recover the same experimental findings even if the LM is randomly initialised at the onset.

4. Experiments

In our experimentation, we will demonstrate that the recipe offered by TransNAR admits significant benefits to out-of-distribution reasoning in language model architectures. In this section we provide details of our experimental setup.

Transformer architecture and initialisation. We use a decoder-only, 6 layers, transformer model from the Chinchilla family [13] pretrained on MassiveText [23]. In particular we use a model of 70 million parameters with a context size $2,048$. To showcase the suitability of our approach regardless of the starting point of training, we run two ablative variants. In the first, the Transformer weights are initialised with the outcome of the pre-training—emulating a *fine-tuning* scenario—and in the second, we use a fully random initialisation. In our figures and tables of results that follow, we will refer to these two setups as *"Pretrained"* and *"Untrained"*.

Randomized positional encoding. Previous work has emphasised the significant relevance of *randomised* positional embeddings in Transformers, especially for enabling more robust reasoning [25]. Corresponding to previous studies on the generalization capabilities of language models, randomised positional embeddings have indeed led to significant gains on both our baselines and TransNAR, allowing more interesting reasoning behaviour to emerge in both. As such, all our experiments in this paper will use randomised positional embeddings. We provide more details in Appendix 6.

Pre-training the NAR. Following Ibarz et al. [14], we pre-train a multi-task MPNN-based NAR on input problem sizes of up to 16, from the CLRS-30 benchmark [34]. Owing to its graph structure formulation, such NARs are capable of significant OOD generalisation—sometimes staying competitive on graphs that are $4\times$ the size. We will attempt to utilise such models through TransNAR, to convey this rich representational knowledge into text.

Combining cross-attention contributions from nodes and edges. The NAR pre-trained by the method presented in Ibarz et al. [14] produces both node and edge latent representations, and we cross-attend to both of them, as they may contain complementary useful information. To cross-attend over the edge features, $\mathbf{E}^{(t)} \in \mathbb{R}^{N\times N\times k}$, we apply Equation 3 one more time (with $\mathbf{\Theta}^{(t)}$ cross-attending over $\mathbf{E}^{(t)}$), with the caveat that we need to flatten the first and second axis of $\mathbf{E}$ into one, to make sure the dimensionalities match. We combine the cross-attention contribution from the node and edge embeddings provided by the pre-trained NAR by con-

Algorithm	Input & Target	#tokens

Articulation points	`articulation_points:`
	`A: [[0 0 0 0], [0 1 0 0], [0 0 0 0], [0 0 0 1]]`
	`is_cut: [0 0 0 0]`
		63

Binary search	`binary_search:`
	`key: [0.011 0.029 0.635 0.719], target: 0.122`
	`return: 2`
		49

Insertion sort	`insertion_sort:`
	`key: [0.561 0.081 0.892 0.565]`
	`pred: [0.081 0.561 0.565 0.892]`
		65

Jarvis' march	`jarvis_march:`
	`x: [-1.22 -1.05 0.331 -1.55], y: [-1.48 1.39 0.899 0.1]`
	`in_hull: [1 1 1 1]`
		75

KMP Matcher	`kmp_matcher:`
	`string: [0 0 0 1], key: [3 3 2 3]`
	`match: 0`
		37

Matrix Chain Order	`matrix_chain_order:`
	`p: [0.461 0.957 0.462 0.42]`
	`s: [[0 0 0 0], [0 0 1 2], [0 0 0 2], [0 0 0 0]]`
		77

Task Scheduling	`task_scheduling:`
	`d: [2 3 3 4], w: [0.042 0.875 0.954 0.761]`
	`selected: [0 1 1 1]`
		61

Table 1. **Samples from CLRS-text for various algorithmic tasks of problem size 4.** Input and target parts of the examples are clearly specified. Note that variability is limited at size 4, meaning that some algorithms may have trivial answers for the given inputs. Such effects tend to quickly disappear with scaling the problem size.

catenation, followed by the application of a linear layer. We have attempted to use other reduction schemes such as summing the vectors, or applying a 2-layer MLP. We have also attempted different preprocessing schemes such as orthogonalising the contributions using the Gram-Schmidt process to ensure their algebraic complementarity before combining them. However, none of these variations have brought improvements over our original approach.

Datasets. We build a text version of the CLRS-30 benchmark [34], which we call *CLRS-text*, that is suitable for training and evaluating language models. Table 1 showcases several samples from this dataset, along with their input size and number of tokens. Note that the textual representation is directly derived from the graph-based CLRS-30 in a deterministic manner, so the two datasets convey exactly the same information. However, due to the tokenised representation, there are stringent limitations on how large of a problem size we can evaluate on without running out of context length for the Chinchilla models.

Accordingly, we train our algorithms on smaller problem sizes—$[4, 8]$ and 12, and evaluate on problem sizes 10 (*out-of-distribution—interpolation*), 12 (*in-distribution*), 14 (*out-of-distribution—extrapolation*).

It is worth noting that CLRS-text is among the most challenging long-range reasoning tasks for language models, compared to the present evaluation landscape—a clear step-up in complexity from grade school math, mainly because it allows for explicitly controlling for out-of-distribution generalisation. Yet, there exists a clear polynomial-time-algorithmic description for each of them, meaning that they can be explained in relatively little parameters—certainly way less than a typical large language model of today!

The dataset comprises $10,000$ samples per algorithm per input size, making up a total of $2,400,000$ data points, split as per above into 70% for training and 30% for validation.

Training details. We train all models over seven epochs of the training data with a batch size of 256 and employ an Adam optimizer [17] with a learning rate of 10^{-4}. We apply randomized positional encoding with a maximal length of $8,192$ on top of Rotary Positional Encoding (RoPE) used in the base Chinchilla transformer [13]. As previously mentioned, for all TransNAR models, we keep the NAR frozen during training.

Evaluation metrics. We refrain from computing the accuracy of each model using exact string matching, on the grounds that this metric does not provide insights as to the causes of failure on a particular datapoint, and more crit-

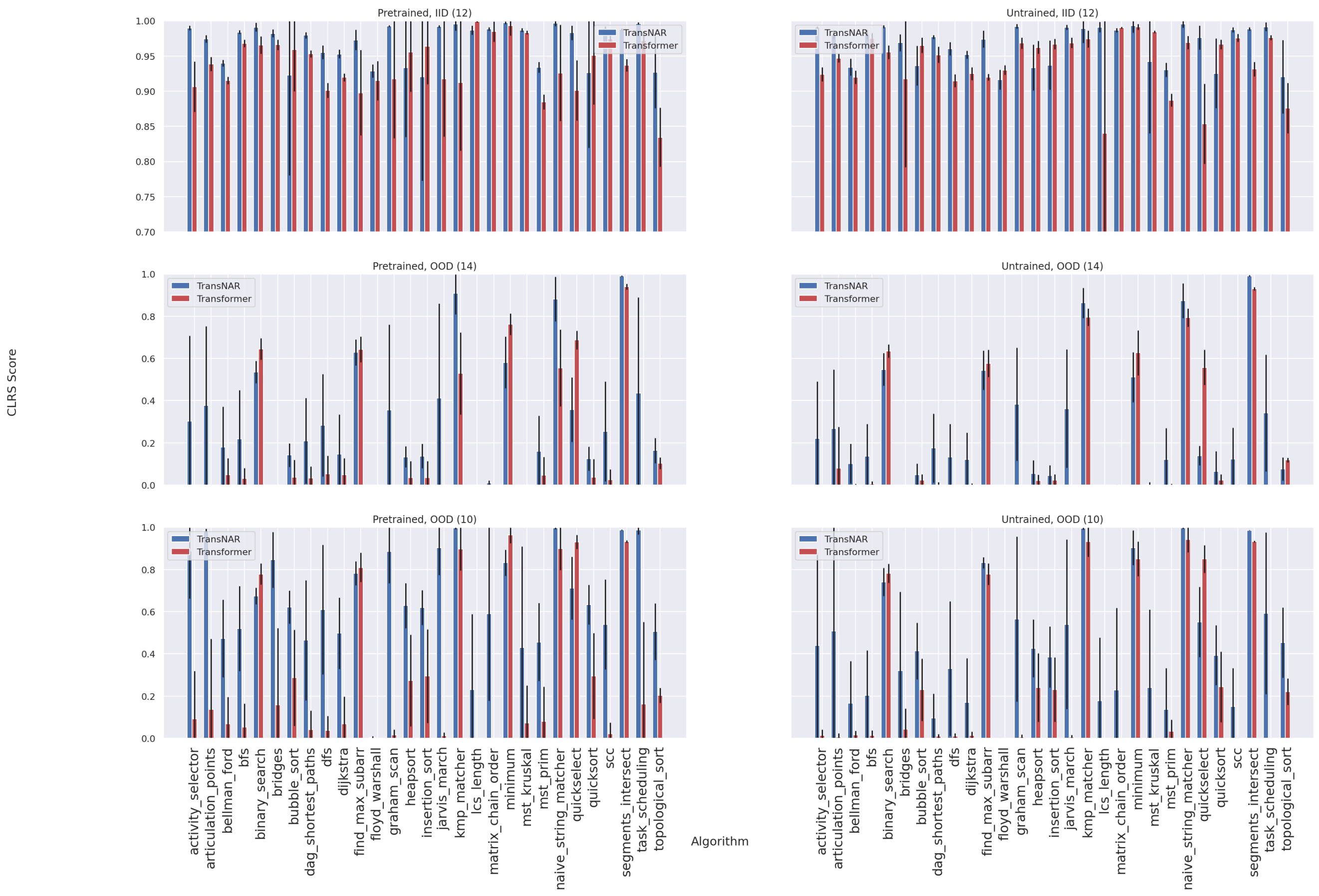

Figure 4. **TransNAR significantly outperforms the baseline Transformer.** We compare TransNAR to its corresponding Transformer baseline on various algorithms and for various input sizes: 12 is the largest size in-distribution. The other two sizes tested—10 and 14—are out-of-distribution, with the former testing interpolation and the latter extrapolation. Note that in-distribution generalisation is much easier for Transformers, and as such, we have modified the y-axis for this setting only to the $[0.7, 1.0]$ range. It is evident that, on most algorithmic tasks of interest, the TransNAR is capable of outperforming its baseline Transformer. Additionally, we see that this advantage is consistent across both training regimes: initial training and finetuning. The metric used is the CLRS score. Each model was trained with 4 random seeds. Error bars indicate ± 1 standard deviation.

ically, it fails to capture how close to correctness a given model output is (as observed by Veličković et al. [34]). Instead, we evaluate the performance of each model according to three metrics measuring capabilities of increasing complexity over the generated text:

1. The *shape score*: a binary-valued metric capturing whether the output has the right shape. For example, if we consider a sorting task, the output should have exactly the same number of elements as the input. Similarly, if the output is a matrix, we ensure its shape is consistent with both the input and the task.
2. The *parse score*: a binary-valued metric capturing whether the output is free from any illicit characters, for example, considering again a sorting task on a list of numbers, the output shouldn't contain any letters of the alphabet.
3. The *CLRS score*: The percentage of elements in the output that match the ground truth answer. This score is the one traditionally used in CLRS-30 [14, 34], hence its name. Note that we automatically assign a CLRS score of 0 if the shape score is 0, as there is no clear correspondence between output indices.

These multi-faceted scores are explicitly designed to capture the various failure modes of LLMs when learning to reason over text inputs: they may overly specialise to the training problem sizes (leading to incorrect shapes at test time), fail to cope with unseen number combinations (leading to incorrect parsing), and of course, produce incorrect or inconsistent outputs, captured by the CLRS score.

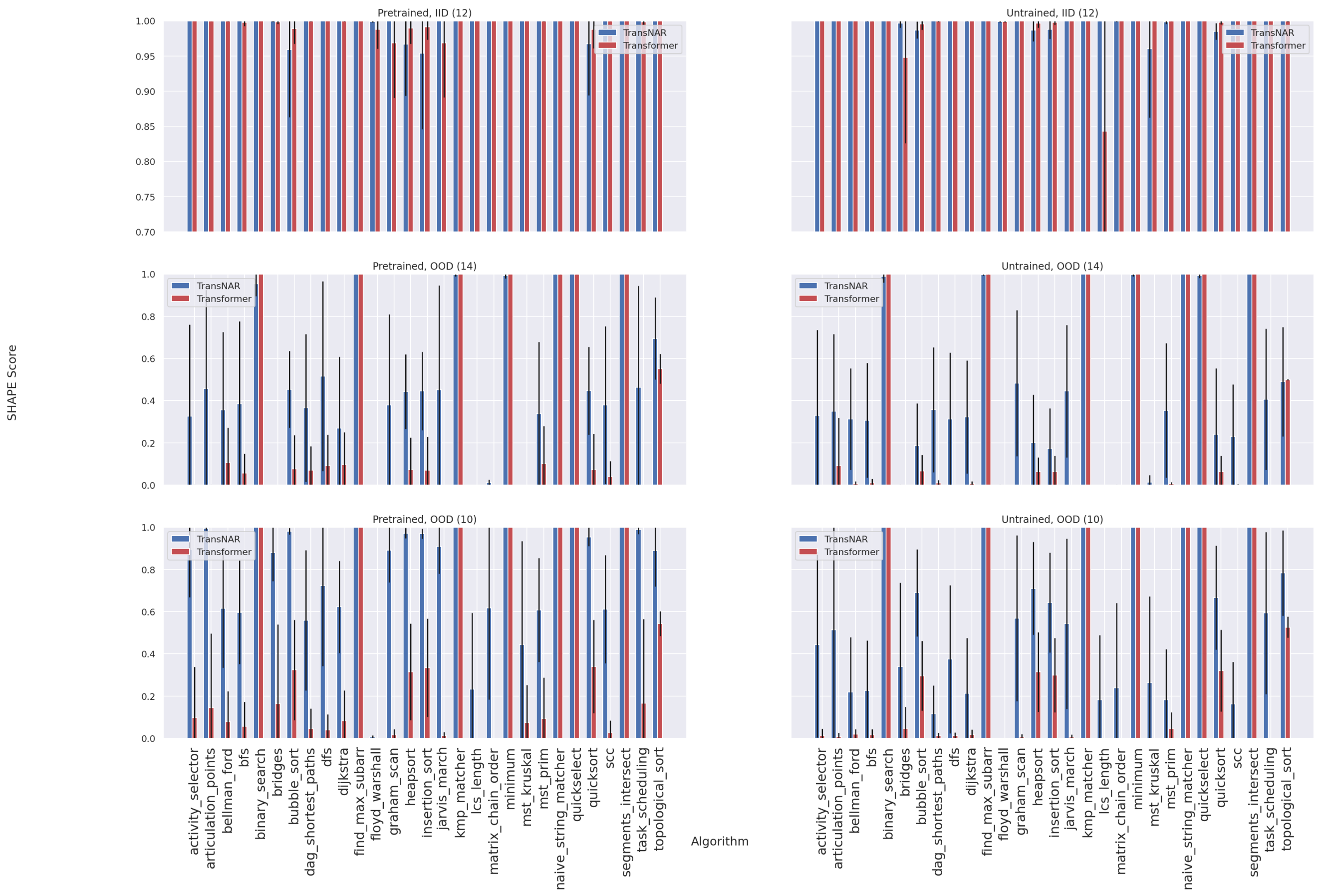

Figure 5. **Shape Score:** The TransNAR significantly outperforms its baseline in terms of producing correct shapes. This score sheds light on an obvious failure model of regular Transformers out-of-distribution: they fail to capture the seemingly trivial dependency between input size and output size, and so irrespective of the complexity of the algorithm itself. The TransNAR model manages to considerably alleviate this problem (with many emerging gains), albeit, these gains do not always lead to perfect scores, implying a fruitful direciton for future research.

4.1. Results

We summarize our findings in Figure 4 (for CLRS score). Our results show that our TransNAR significantly outperforms the baseline Transformer overall, and on most individual algorithms, both in- and out-of-distribution. In particular, we see that our approach not only enhances existing out-of-distribution generalisation capabilities, but also causes the emergence of these capabilities when there was a complete lack thereof—reflected in the figure by zero or near-zero performance of the baseline [35].

The analysis of shape score (Figure 5) provides an additional way to shed light on why TransNAR performed as well as it did. Recall, first, that CLRS score is necessarily zero if shapes do not match. Observing the shape scores achieved, it appears that grounding Transformer outputs in NAR embeddings significantly increases the proportion of inputs for which a Transformer will produce an output of the correct shape—indicating that this is one very specific failure mode that TransNAR helps alleviate.

We note, however, that there remain a few algorithms for which TransNAR is not able to outperform the baseline. A closer look at the results indicates that such tasks (Binary Search, Find Maximum Subarray, Minimum, and Quickselect) all involve an element of *searching* for a particular index in an input list. This hints at a unified failure mode: as these failures persist both when interpolating and extrapolating, the model as implemented is not able to generalise to novel *index boundaries* unseen in the training data. We therefore suspect that the use of *index hints*—as already demonstrated by Zhou et al. [39]—is a promising avenue for ameliorating this behaviour. Alternatively, it might be the case that the final NAR-computed hidden states are harder to decode by the cross-attention layers in a generalisable way, and therefore might require either giving an additional capacity to the cross-attention and/or performing a more *progressive* decoding in that: instead of having all cross-attention layers decoding from the final NAR-computed hidden states, s, we could have early cross-attention layers decode from hidden states coming from earlier message passing steps, and later cross-attention layers decode from the later message passing steps.

Lastly, we provide parse scores in Appendix 7—omitting them from the main text because, in most cases, parsing can be done at full accuracy.

4.2. Limitations

While our approach demonstrates favourable average performance under all out-of-distribution regimes we have evaluated, we highlight that TransNAR requires access to both textual and graph-representation inputs to be efficiently trainable and usable. While this limits TransNAR to cases where a particular ground-truth executor or simulator (or prior belief about one) is available, now that we know that TransNAR-like ideas are beneficial, future research can enable the deployment of such ideas into purely unimodal Transformers. For example, lifting the need for a second data stream can be done by *distilling* the knowledge acquired by the trained TransNAR model into a vanilla Transformer model.

5. Conclusions

We presented a Transformer-NAR hybrid architecture: a language model that combines the language understanding skills of a Transformer with the robust algorithmic reasoning capabilities of a pre-trained graph neural network-based neural algorithmic reasoner, to solve algorithmic tasks specified in natural language. We have demonstrated the superiority of our model over its Transformer-only counterpart on the CLRS-text benchmark, in the in-distribution, and more importantly, in two out-of-distribution regimes, with respect to the input problem size. We hope that future work will draw on our results and insights shared here, and further investigate expansions of interest, notably, datasets with more ambiguous problem specifications (as often encountered in the real world), and for which their corresponding equivalent solver-ready symbolic inputs are not given in advance.

References

[1] Emmanuel Abbe, Samy Bengio, Aryo Lotfi, and Kevin Rizk. Generalization on the unseen, logic reasoning and degree curriculum. *arXiv preprint arXiv:2301.13105*, 2023. 3

[2] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023. 1

[3] Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katie Millican, Malcolm Reynolds, Roman Ring, Eliza Rutherford, Serkan Cabi, Tengda Han, Zhitao Gong, Sina Samangooei, Marianne Monteiro, Jacob Menick, Sebastian Borgeaud, Andrew Brock, Aida Nematzadeh, Sahand Sharifzadeh, Mikolaj Binkowski, Ricardo Barreira, Oriol Vinyals, Andrew Zisserman, and Karen Simonyan. Flamingo: a visual language model for few-shot learning, 2022. 3, 4

[4] Cem Anil, Yuhuai Wu, Anders Andreassen, Aitor Lewkowycz, Vedant Misra, Vinay Ramasesh, Ambrose Slone, Guy Gur-Ari, Ethan Dyer, and Behnam Neyshabur. Exploring length generalization in large language models. *Advances in Neural Information Processing Systems*, 35: 38546–38556, 2022. 1, 3

[5] Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023. 1

[6] Beatrice Bevilacqua, Kyriacos Nikiforou, Borja Ibarz, Ioana Bica, Michela Paganini, Charles Blundell, Jovana Mitrovic, and Petar Veličković. Neural algorithmic reasoning with causal regularisation. In *International Conference on Machine Learning*, 2023. 1, 2

[7] Andreea-Ioana Deac, Petar Veličković, Ognjen Milinkovic, Pierre-Luc Bacon, Jian Tang, and Mladen Nikolic. Neural algorithmic reasoners are implicit planners. *Advances in Neural Information Processing Systems*, 34:15529–15542, 2021. 2, 3

[8] Andrew Dudzik and Petar Veličković. Graph neural networks are dynamic programmers. *ArXiv*, abs/2203.15544, 2022. 1

[9] Nouha Dziri, Ximing Lu, Melanie Sclar, Xiang Lorraine Li, Liwei Jian, Bill Yuchen Lin, Peter West, Chandra Bhagavatula, Ronan Le Bras, Jena D Hwang, et al. Faith and fate: Limits of transformers on compositionality. *arXiv preprint arXiv:2305.18654*, 2023. 1

[10] Dobrik Georgiev, Danilo Numeroso, Davide Bacciu, and Pietro Liò. Neural algorithmic reasoning for combinatorial optimisation. *arXiv preprint arXiv:2306.06064*, 2023. 2

[11] Dobrik Georgiev, Ramon Vinas, Sam Considine, Bianca Dumitrascu, and Pietro Lio. Narti: Neural algorithmic reasoning for trajectory inference. 2023. 2

[12] Yu He, Petar Veličković, Pietro Liò, and Andreea Deac. Continuous neural algorithmic planners. In *Learning on Graphs Conference*, pages 54–1. PMLR, 2022. 2

[13] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals, and L. Sifre. Training compute-optimal large language models. *ArXiv*, abs/2203.15556, 2022. 4, 5

[14] Borja Ibarz, Vitaly Kurin, George Papamakarios, Kyriacos Nikiforou, Mehdi Abbana Bennani, R. Csordás, An-

drew Dudzik, Matko Bovsnjak, Alex Vitvitskyi, Yulia Rubanova, Andreea Deac, Beatrice Bevilacqua, Yaroslav Ganin, Charles Blundell, and Petar Veličković. A generalist neural algorithmic learner. In *LOG IN*, 2022. 1, 2, 4, 6

[15] Andrew Jaegle, Sebastian Borgeaud, Jean-Baptiste Alayrac, Carl Doersch, Catalin Ionescu, David Ding, Skanda Koppula, Daniel Zoran, Andrew Brock, Evan Shelhamer, et al. Perceiver io: A general architecture for structured inputs & outputs. *arXiv preprint arXiv:2107.14795*, 2021. 3

[16] Jonas Jürß, Dulhan Hansaja Jayalath, and Petar Veličković. Recursive algorithmic reasoning. In *The Second Learning on Graphs Conference*, 2023. 1, 2

[17] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017. 5

[18] Rémi Leblond et al. AlphaCode 2 Technical Report. 2023. 3

[19] Nayoung Lee, Kartik Sreenivasan, Jason D Lee, Kangwook Lee, and Dimitris Papailiopoulos. Teaching arithmetic to small transformers. *arXiv preprint arXiv:2307.03381*, 2023. 3

[20] Danilo Numeroso, Davide Bacciu, and Petar Veličković. Dual algorithmic reasoning. *arXiv preprint arXiv:2302.04496*, 2023. 2

[21] Euan Ong. Probing the foundations of neural algorithmic reasoning. Technical report, University of Cambridge, Computer Laboratory, 2023. 3

[22] Chendi Qian, Didier Chételat, and Christopher Morris. Exploring the power of graph neural networks in solving linear optimization problems. *arXiv preprint arXiv:2310.10603*, 2023. 2

[23] Jack W. Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, Jordan Hoffmann, Francis Song, John Aslanides, Sarah Henderson, Roman Ring, Susannah Young, Eliza Rutherford, Tom Hennigan, Jacob Menick, Albin Cassirer, Richard Powell, George van den Driessche, Lisa Anne Hendricks, Maribeth Rauh, Po-Sen Huang, Amelia Glaese, Johannes Welbl, Sumanth Dathathri, Saffron Huang, Jonathan Uesato, John Mellor, Irina Higgins, Antonia Creswell, Nat McAleese, Amy Wu, Erich Elsen, Siddhant Jayakumar, Elena Buchatskaya, David Budden, Esme Sutherland, Karen Simonyan, Michela Paganini, Laurent Sifre, Lena Martens, Xiang Lorraine Li, Adhiguna Kuncoro, Aida Nematzadeh, Elena Gribovskaya, Domenic Donato, Angeliki Lazaridou, Arthur Mensch, Jean-Baptiste Lespiau, Maria Tsimpoukelli, Nikolai Grigorev, Doug Fritz, Thibault Sottiaux, Mantas Pajarskas, Toby Pohlen, Zhitao Gong, Daniel Toyama, Cyprien de Masson d'Autume, Yujia Li, Tayfun Terzi, Vladimir Mikulik, Igor Babuschkin, Aidan Clark, Diego de Las Casas, Aurelia Guy, Chris Jones, James Bradbury, Matthew Johnson, Blake Hechtman, Laura Weidinger, Iason Gabriel, William Isaac, Ed Lockhart, Simon Osindero, Laura Rimell, Chris Dyer, Oriol Vinyals, Kareem Ayoub, Jeff Stanway, Lorrayne Bennett, Demis Hassabis, Koray Kavukcuoglu, and Geoffrey Irving. Scaling language models: Methods, analysis & insights from training gopher, 2022. 4

[24] Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Matej Balog, M Pawan Kumar, Emilien Dupont, Francisco JR Ruiz, Jordan S Ellenberg, Pengming Wang, Omar Fawzi, et al. Mathematical discoveries from program search with large language models. *Nature*, pages 1–3, 2023. 3

[25] Anian Ruoss, Grégoire Delétang, Tim Genewein, Jordi Grau-Moya, R. Csordás, Mehdi Abbana Bennani, Shane Legg, and Joel Veness. Randomized positional encodings boost length generalization of transformers. *ArXiv*, abs/2305.16843, 2023. 3, 4

[26] Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. *arXiv preprint arXiv:2302.04761*, 2023. 3

[27] Ruoqi Shen, Sébastien Bubeck, Ronen Eldan, Yin Tat Lee, Yuanzhi Li, and Yi Zhang. Positional description matters for transformers arithmetic. *arXiv preprint arXiv:2311.14737*, 2023. 3

[28] Trieu H Trinh, Yuhuai Wu, Quoc V Le, He He, and Thang Luong. Solving olympiad geometry without human demonstrations. *Nature*, 625(7995):476–482, 2024. 3

[29] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. 1, 4

[30] Petar Veličković. Everything is connected: Graph neural networks. *Current Opinion in Structural Biology*, 79:102538, 2023. 1

[31] Petar Veličković, Rex Ying, Matilde Padovano, Raia Hadsell, and Charles Blundell. Neural execution of graph algorithms. *arXiv preprint arXiv:1910.10593*, 2019. 2, 4

[32] Petar Veličković, Matko Bošnjak, Thomas Kipf, Alexander Lerchner, Raia Hadsell, Razvan Pascanu, and Charles Blundell. Reasoning-modulated representations. In *Learning on Graphs Conference*, pages 50–1. PMLR, 2022. 2

[33] Petar Veličković and Charles Blundell. Neural algorithmic reasoning. *Patterns*, 2, 2021. 1

[34] Petar Veličković, Adrià Puigdomènech Badia, David Budden, Razvan Pascanu, Andrea Banino, Mikhail Dashevskiy, Raia Hadsell, and Charles Blundell. The clrs algorithmic reasoning benchmark. In *International Conference on Machine Learning*, 2022. 1, 2, 4, 5, 6

[35] Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*, 2022. 7

[36] Louis-Pascal Xhonneux, Andreea-Ioana Deac, Petar Veličković, and Jian Tang. How to transfer algorithmic reasoning knowledge to learn new algorithms? *Advances in Neural Information Processing Systems*, 34:19500–19512, 2021. 2

[37] Keyulu Xu, Mozhi Zhang, Jingling Li, Simon S Du, Kenichi Kawarabayashi, and Stefanie Jegelka. How neural networks extrapolate: From feedforward to graph neural networks. *arXiv preprint arXiv:2009.11848*, 2020. 2

[38] Hattie Zhou, Azade Nova, Hugo Larochelle, Aaron Courville, Behnam Neyshabur, and Hanie Sedghi. Teaching

algorithmic reasoning via in-context learning. *arXiv preprint arXiv:2211.09066*, 2022. 3

[39] Hattie Zhou, Arwen Bradley, Etai Littwin, Noam Razin, Omid Saremi, Josh Susskind, Samy Bengio, and Preetum Nakkiran. What algorithms can transformers learn? a study in length generalization. *arXiv preprint arXiv:2310.16028*, 2023. 7

6. Effect of Randomized Positional Encoding

Using randomized positional encoding has benefitted both our model and the baseline. In particular, combining them with NAR hiddens led to improvements OOD, most prevalently in the interpoloation regime (at length 10), but also, to some extent, in the extrapoloation regime (at length 14). One result we found interesting, was that before instating randomized positional encoding, the OOD performance of our hybrid models was limited (in fact thresholded) by the performance of the base LLM. Concretely, if the base LLM achieved near-zero performance, the hybrid architecture would fatally share the same fate. We can see that this is no longer the case: if the base LLM uses randomized positional encoding, even if its performance is near-zero, that of the hybrid architecture can still be reasonably good. This is illustrated in the second column of the figure 4, for example on the Graham Scan, Jarvis March, MST Prim algorithms.

7. Parse Scores

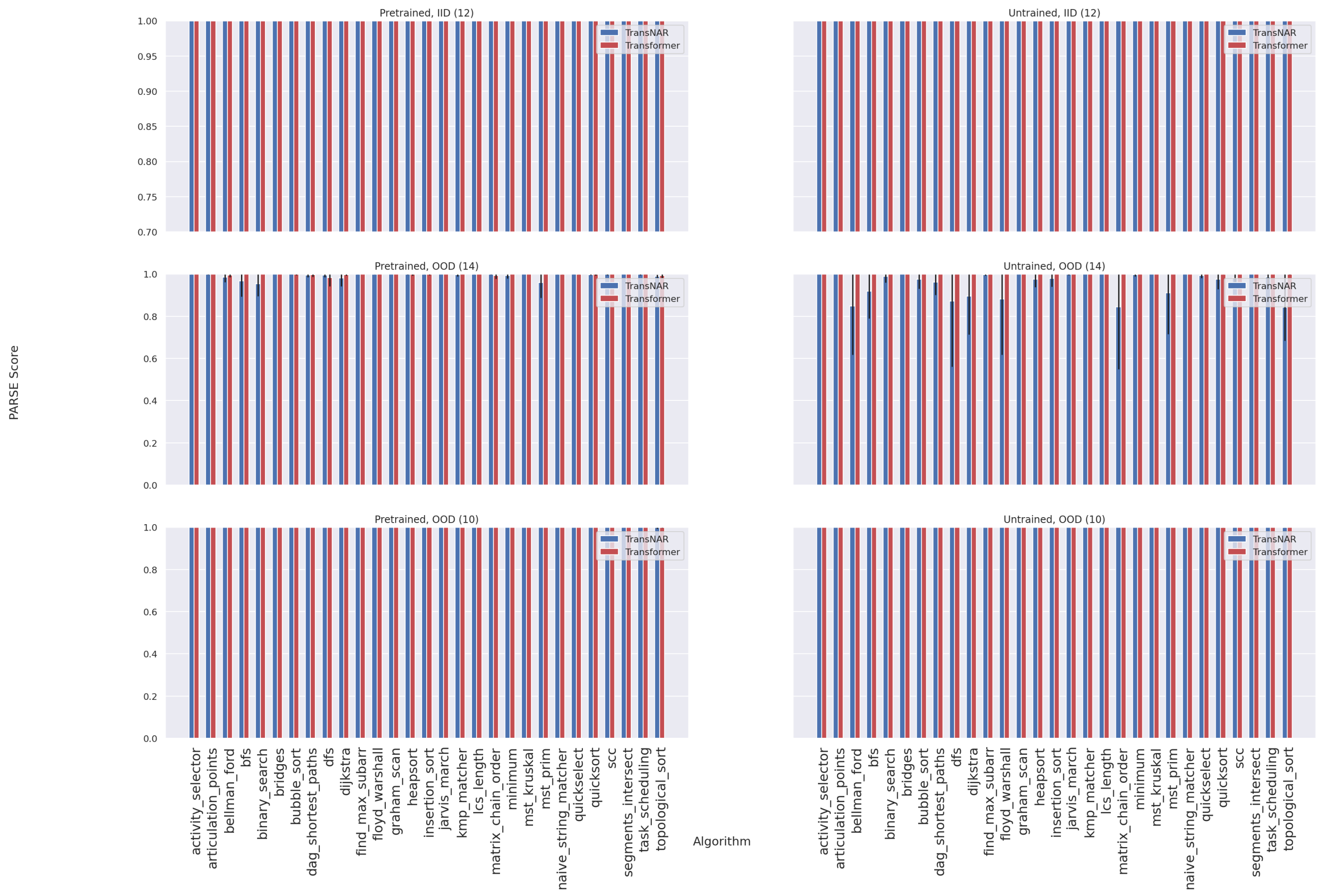

Figure 6. **Parse Score:** We can see that for a few algorithms, the TransNAR architecture falls behind the baseline in the extrapolation regime likely due to an unsufficient capacity of the cross-attention in charge of decoding from the NAR's outputs.