On the Pathological Path-star Task for Language Models

Arvid Frydenlund
University of Toronto, Computer Science
Vector Institute
`arvie@cs.toronto.edu`

Abstract

The recently introduced path-star task is a minimal toy task designed to exemplify limitations to the abilities of language models (Bachmann and Nagarajan, 2024). It involves a *path-star* graph where multiple arms radiate from a single starting node and each node is unique. Then, given the start node and a specified target node which ends one of the arms, the task is to generate the arm containing that target node. This is straightforward for a human but surprisingly difficult for a language model, which they found failed to predict above chance. They hypothesized this is due to a deficiency in teacher-forcing and next-token prediction paradigm.

In this extended abstract, we demonstrate that the task is learnable using teacher-forcing in alternative settings and that the issue is (partially) due to representation. We analyze situations when the models fail to solve the task which leads us to introduce a regularization technique where we pack each training batch with multiple instances of the same graph but with differing target nodes to prevent overfitting. Initial results indicate this helps in solving the task.

1 Introduction

Recently, language models (LMs) have become increasingly capable of solving a variety of complex tasks (Brown et al., 2020; Zoph et al., 2022; Bubeck et al., 2023). This has led to increased interest in determining why this is and the limits to these abilities (Chen et al., 2024). Language models can do many spectacular things, which makes it all the more shocking when they fail on simple tasks. Recently, Bachmann and Nagarajan (2024) introduced one such seemingly simple task designed to showcase pathological behaviour of causal (decoder-only) autoregressive (AR) LMs trained via teacher-forcing. The task is simple by design and thus failure of AR models is both surprising and informative. We begin by describing the task in Sec. 1.1, before analyzing why it is hard for LMs in Sec. 2.

1.1 The Path-star Task

We need to describe the path-star graph, G, i.e. the data meant to be manipulated, the problem specification or question, Q, i.e. the prompt specifying the desired manipulation, and their tokenization.

Let N be the set of unique nodes forming G. A path-star graph contains one central starting node $s \in N$ and D radial arms each of length M (inclusive of s), s.t. $|N| = D(M-1)+1$. s has degree D, all final nodes which end an arm, $F \subset N$ s.t. $|F| = D$, have a single degree, and all others have a degree of 2. See Fig. 1 for an example.

Given G, and a task specification, Q, containing s and a target node $t \in F$, the task is to generate the unique arm, R_t, as a sequence of nodes starting from s until t. i.e. $R_t = \text{sort}(\{\, r \in N \,|\, \forall_{f \in F} \, \text{dist}(r,\, t) \leq \text{dist}(r,\, f) \})$.[1] Let L be the set of possible leading nodes which are adjacent to s i.e. $L = \{\, l \in N \,|\, \text{dist}(l,\, s) = 1\}$. The challenge of the task is predicting the correct leading node $l_t \in L \cap R_t$ from all other leading nodes. By design, there is a uniform $1/D$ chance of this given only G. Prediction over chance *should* be possible by inferring the correct target arm and thus l_t given t in Q. Note, as all nodes are unique, all non-leading nodes are deterministic given their preceding neighbour (closer to s).

When generating the dataset, the nodes in a single graph are uniformly sampled from a set of possible nodes, V, without replacement. G is tokenized as a series of $D(M-1)$ edges where each edge is internally ordered by distance to s and marked by special token '|' so that a given edge, $(u,\, v)$, is a three token sequence '$u\ v$ |'. Q is tokenized as a sequence of four tokens with special tokens marking the beginning and end of Q as '/ $s\ t$ ='. Special beginning- and end-of-sequence tokens are also used, making the final vocabulary size $|V| + 5$.

[1]'dist' is graph distance. We abuse notation by treating R_t as a set and G and Q as sequences after having been tokenized.

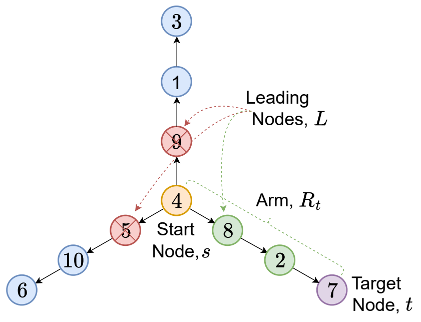

Figure 1: An example path-star graph. $D = 3$, $M = 4$, s is '4', t is '7' R_t is '4 8 2 7', and l_t is '8'. One possible tokenization of $[V,\, Q,\, R_t]$, where the arms **(and not the edges)** are permuted is: 'BOS 4 9 | 9 1 | 1 3 | 4 8 | 8 2 | 2 7 | 4 5 | 5 10 | 10 6 | / 4 7 = 4 8 2 7 EOS'.

1.2 Autoregressive models and training

A causal or decoder-only AR LM models the joint probability of a T-length sequence, y, as a factorized product of local probabilities, as in

$$p(y_1,\, y_2,\, \ldots,\, y_T \,|\, y_0,) = \prod p(y_j \,|\, y_{<j}). \quad (1)$$

Here, we model the path-star task as

$$p(r_1,\, \ldots,\, r_M \,|\, [G,\, Q]) = \prod_{j=1}^{M} p(r_j \,|\, [G,\, Q,\, r_{<j}]), \quad (2)$$

where $x = [G,\, Q,\, r_{<j}]$ is the concatenation of the tokenized graph and problem specification along with the partial ground-truth sequence, $r_{<j}$, forming the given conditioning input to the model. Such a model is trained by via maximum likelihood training, generally referred to as 'teacher-forcing' in the context of language models, as the partial ground-truth sequence is used to condition the model during training instead of the model's own predictions as done during inference (Williams and Zipser, 1989). We minimize $-\sum_{r \in R_t} \log p(\,.\,|\, x)$. Thus the loss is only over the target sequence R_t and not on tokens in the prefix $[G,\, Q]$. This is because the node ids forming G are random and Q necessarily must be provided, thus both are not predictable and can only be used to condition the model.

During inference, G and Q are provided. We consider a non-traditional 'teacher-forced' inference procedure where, instead of generating the arm autoregressively, it is conditioned on $r_{<j}$. Thus inference exactly matches the training procedure and prevents any potential training-inference bias.

We focus on transformer models (Vaswani et al., 2017), where the causal parameterization of AR models is enforced via an attention mask which prevents the token at any step j from depending on any token at step $> j$. This causal restriction applies across the entire input x. Positional embeddings make each token unique. To prevent learning a trivial answer based on position, as a data preprocessing step, **the edges in G are shuffled, which can be seen as a random permutation applied to the edge order of tokenization of G.**

1.3 Failure to learn: Clever Hans hypothesis

Bachmann and Nagarajan (2024) empirically demonstrated three different LMs – finetuned GPT2, a smaller GPT2 trained from scratch, and a state-space model, Mamaba – all fail to predict above $1/D$ chance, even in settings as small as $D = 2$ and $M = 5$ (Radford et al.; Gu and Dao, 2023). They hypothesized this was caused by teacher-forcing. The idea being that there are two possible modes of predictions which the model can learn. The first is the desired mode which learns to represent the entire path between s to t. This mode is necessary for predicting l_t. Whereas, the second mode makes trivial predictions about the next node in the arm given the previous node. This mode only needs to lean superficial information about edge structure but not graph structure and is, by design of the task, sufficient for predicting all non-leading nodes given the correct preceding node.

Bachmann and Nagarajan (2024) argued that teacher forcing will result in learning the second mode, referred to as the *Clever Hans* cheat (CHC). This is because teacher-forcing conditions on the correct ground-truth, which in this case is the correct preceding node in the arm. Also, when applied to AR models, it is restricted to making a single next-token prediction and hence precludes learning any long term planning. Then, once the CHC is learnt, it will discourage learning the desired mode necessary for predicting l_t. Their intuition, which admittedly is not proven, is that, sequence modelling relies on the intermediate training steps across the sequence to form a coherent representation of the overall sequence. In our case, that would be a representation of the entire arm structure, however, here those intermediate steps do not participate in learning such a structure but are rather absorbed into learning the trivial CHC, resulting in a loss of this intermediate training signal.

They presented empirical evidence for the CHC hypothesis by considering the overall sequence accuracy when provided with the correct preceding predictions (i.e. teacher-forced generation). Here, all non-leading tokens are learnt with 100% accuracy and the leading token is only predicted at $1/D$ chance, leading to an overall sequence accuracy of $1/D$ (See their Fig. 3 and our Fig. 3).

Interestingly, a trivial solution to the task exists if the model can look-ahead M tokens to the end of the arm as the model just needs to find and match the correct target token. Once done, it can apply the CHC in reverse order to determine the arm. This led them to provide two additional supporting empirical arguments as to why they believe the issue stems from teacher-forcing. First, they modified the task to require that the arm be generated in reverse order. This makes task trivial as the CHC can just be applied in reverse order via supervision.

Second, they introduced a 'teacher-less' model (Monea et al., 2023). This works by using M masked tokens, m, to make make all M predictions in independently of the ground-truths i.e. $x = [G, Q, m_1, \ldots, m_M]$. This completely removes teacher-forcing as it removes all dependencies between target-tokens during prediction. Of the 15 reported experiments, this method allows the model to solve the task in 5 instances: for the $D = 2$ experiment using the small GPT2, and for $D \in \{2, 3, 4\}$ (but not $D = 5$) when using large GPT2. Thus this method did not work consistently.

Importantly, they establish that, 1), the failure is not due to the amount of training data, 2), that the failure is in-distribution, and 3), that it is not due to any exposure bias or other differences between the training and inference procedures (Bengio et al., 2015; Ranzato et al., 2016; Arora et al., 2022). This allows them to disclude these alternative explanations and conclude that the CHC causes the learning problem which is itself a consequence of teacher-forcing and next-token prediction. **This leads to a discussion concerning possible fundamental limitations to the next-token prediction paradigm, with the path-star task being offered as a counter example to the paradigm being sufficient to learn any task**.

2 Methods and Results

We solely focus on the small LM setting under the belief that such models should be able to learn such a simple task and that the biases from the pretrained data and any emergent abilities of LLMs will just obfuscate the root problem. We implement our models using Fairseq (Ott et al., 2019). Our AR model have 6 layers and 200 dim. embeddings. Each layer has a feed-forward dim. of 800 and 8 heads. We use Adam (Kingma and Ba, 2014) with a learning rate of 0.0005, a dropout rate of 0.1 and a weight decay of 0.01. We train with 16-bit precision and a batch size of 1024. Each model is given a maximum of 100 epochs and stop training if the validation loss drops below 0.001 (See Fig 2). Note this is smaller than both GPT2 models used by Bachmann and Nagarajan (2024) which used 36 and 12 layers with larger embeddings.

Following Bachmann and Nagarajan (2024), $|V| = 100$ and $M = 5$. Each dataset for $D \in \{2, 3, 4, 5\}$ is made up of 2,000,000 training and 20,000 test samples of randomly generated G, Q pairs without any overlap. Unlike them, we randomly permute G at every epoch instead of just once in an attempt to prevent overfitting.

We present our work as an investigation over a series of hypotheses and corresponding experiments to get at the heart of the path-star mystery. As such, we report intermediate results and describe new methodology as it becomes motivated. We try to present results in order of our findings, however, we need to give some post-hoc explanations for our methodology in order for the reader to understand the contents of Tables (1, 2, 3, 4). First, in our initial experimentation (using $D = 2$), we found that the models would be able to solve the task seemingly at random (under modified task conditions). This motivated the use of running multiple trials for each experiment under different random seeds. For all listed experiments we consider the percent of trials that correctly succeed in learning the task across 11 trials. Second, we found it was necessary to set attention dropout to zero, which makes sense given the task requires routing node information across positions. Third, we also found that we required learned positional embeddings instead of sinusoidal embeddings. We suspect that the later results in too strong of a positional bias when randomly permuting the edges in G. As an aside, we also found that for the decoder-only model, not using any positional embeddings could also work. This is because positional information will arise out of the asymmetry induced by the causal masking.

2.1 A reproduction of empirical results

As the results of Bachmann and Nagarajan (2024) are surprising, we independently verify them as an initial step. Experiment (exp.) 1 of Table 1, confirms that the task is not learnable under the initial conditions. Exp. 2 confirms that reversing

ID	Perm.	Q	Tgt./Dir.	C.	$D = 2$		$D = 3$		$D = 4$		$D = 5$	
1	Edge	End	Fwd.	0	0%	0%	0%	0%	0%	0%	0%	0%
2	Edge	End	Rev.	0	100%		100%		100%		100%	
3	Edge	End	l_t-only	0	0%	0%	0%	0%	0%	0%	0%	0%
4	Arm	End	Fwd.	0	100%		36%	0%	9%	0%	9%	0%
5	Arm	Start	Fwd.	0	100%		100%		100%		100%	
6	Edge	Start	Fwd.	0	0%	0%	0%	0%	0%	0%	0%	0%
7	Arm	End	Fwd.	1	100%		91%	9%	91%	9%	36%	55%

Table 1: Percent of successful trials (n=11) using the AR (decoder-only) model. 'ID' is the experiment ID. 'Perm.' is the type of random permutation applied to G (Sec. 2.2.2). 'Q' is the relative position of Q to G when tokenizing (Sec. 2.2.3). 'Tgt./Dir.' is the type of target we are trying to generate (Sec. 2.2.1). And 'C.' is the number of contrastive samples used (Sec. 2.4). For each experiment in $D \in \{2, 3, 4, 5\}$, we report the percent of the 11 trials which succeeded in learning the task to at least a threshold of 95% sequence accuracy (which meant 100% for the AR models) in the first column. In the second column, we report the percent of unsuccessful trials where the valid and training loss has *not* diverged i.e. 0% means all trials have overfit

the arm results in a trivial 100% success rate.

2.2 Simplifying the task

Having confirmed the results in the original task setting, our method to investigate the issue will be to simplify the task until it becomes consistently solvable. We begin by considering the target-side.

2.2.1 Evidence against the CHC hypothesis

Under one interpretation of the CHC hypothesis, l_t is indecipherable due to the model being overwhelmed by the CHC. As such, we should be able to learn a simplified version of the task where we only predict l_t directly instead of the entire arm R_t. Exp. 3 of Table 1, shows this produces the same negative result as when predicting the entire arm.

The more charitable interpretation of the CHC hypothesis is that the core issue concerns the fact that the CHC removes necessary intermediate training signal for learning the task. Under this interpretation, we have not disproved the core hypothesis, but have shown that the entire CHC aspect is irrelevant to the underlying issue. That is, if lack of intermediate supervision is the core issue, we should just explicitly remove it from the task description and cut out the red-herring of the CHC.

However, we do not believe that lack of intermediate supervision is the real issue and take this result as a sign that something else is at play. To solve this task, all the model needs to do is 1) determine the final node, 2) trace back each arm from the final node to its leading node, and 3) predict that leading node. Importantly, this requires that we can correctly represent the arms in the graph. This motivates us to experiment with simplifying the source-side of the task instead of the target-side.

2.2.2 Alternative hypothesis: representation issues due to the permutation of G

Our first hypothesis as to what is preventing learning the solution is that it is a representation issue due to randomly permuting the edges of G. This will corrupt the arm structure with the model seemingly unable to recover the structure. In particular, when using a causal model, all information can only be routed forward in 'time' and this may induce difficulties when trying to recover and route information across the arm structure. Not only does permutation make routing information across the arms harder, or even impossible, but the difficulty in learning might also be due to the assumptions we make when we decompose the joint probability as in Eq. 1. Specifically, we are parameterizing the model to a specific decomposition (Yang et al., 2019; Liao et al., 2020). However, by permuting the arms, we are forcing the model to learn an exponential number of possible decompositions. This may be a challenge, even when using an overparameterized model like a transformer and may explain the difficulty of the task.

Thus we can simplify the task where we retain the arm structure by only permuting the order of the arms relative to each other (but not the internal order of the edges). Refer to this change as *Edge*- v.s. *Arm*-wise permutation. If this is solvable, then we know that the issue lies in the corruption of the arm information via permuting the edges. Exp. 4 of Table 1 shows this improves the results, with $D = 2$ being consistently solved, but with a diminishing success-rate as D increases. These partial improvements lead us to a related hypothesis.

2.2.3 Alternative hypothesis: representation issues due to the order of G and Q

If we can only route information into the future, maybe our representation issue stems from that fact that we have placed the problem specification after the graph during tokenanization. That is, we have placed the information needed to specify what to do with the data after the actual data. This means that the latent representation of G formed by the LM can not depend on Q. Thus instead we form our input as $x = [Q, G, r_{<j}]$. Refer to this as Q's position being either *Start* v.s *End*. Exp. 5 of Table 1 demonstrates that this consistently solves the task when combined with permuting the arms only, but goes back to being completely unsolved when combined with permuting the edges (Exp. 6). While this shows that the task is solvable, it is unsatisfying as we require stronger supervisory information in this setting. This also begs the question as to why placing Q after G is at times solvable, even if we understand why it makes the task harder.

As the causal constraint of decoder-only models potentially induces these issues, we are motivated to change the model specification to see if abandoning this constraint will solve the task.

2.3 Changing the model parameterization

2.3.1 Encoder-decoder model, or, alternative hypothesis: it's the causal constraint

Here we use encoder-decoder model with a 6-layer encoder with a 3-layer decoder with tied embeddings. Removing the causal constraint on the source-side encoding of $[G, Q]$ makes the relative position of Q to G irrelevant. If this model can consistently solve the task, it will demonstrate that the underlying issue is that the causal constraint prevents the decoder-only model from recovering the arm structure with edge-wise permutation.

Exp. 9 in Table 2 demonstrates that using a non-causal encoder representation does not solve the problem with edge-wise permutation. This motivates us to revisit the 'teacher-less' methodology as it has been shown to partially work and is an alternative non-causal methodology.

2.3.2 Non-autoregressive models

Bachmann and Nagarajan (2024) reported that 'teacher-less' models where unable to solve the task in the small LM setting. Here we attempt to improve their results. We begin by modify their their 'teacher-less' model as it was designed to modify an LM post-hoc, which is not applicable here (Monea et al., 2023). Instead, note this is actually just a kind of non-autoregressive model (NAR) (Gu et al., 2018; Wang et al., 2018; Gu and Kong, 2021).

NAR models treat all targets as independent in order to make multiple predictions in parallel instead of sequentially. This is achieved by removing the causal constraint i.e. attention mask. In the case of (fully) NAR models, full independence is assumed. However, this can lead a poor model as it limits the ability to learn from dependencies inherent in the sequence (Lee et al., 2018; Qian et al., 2021). This lead to the development of iterative autoregressive models (IAR)[2] which assume partial dependencies, both during training and inference – except in the first generation step (Lee et al., 2018; Ghazvininejad et al., 2019). Importantly, IAR models assume no order-of-generation. This allows for the model to potentially learn the reverse order solution without supervision.

To train an IAR model, an order-permutation of the sequence is sampled, along with a time-step, j such that model conditions on the permuted or 'unmasked' ground-truths prior to step j. This is equivalent to the MLM objective with a dynamically sampled masking rate, where the the uniform masking is acting as the permutation (Devlin et al., 2019; Lee et al., 2018; Ghazvininejad et al., 2019). Thus use teacher-forcing, but it is just applied to a permuted sequence order. We use CMLM (Ghazvininejad et al., 2019) for an encoder-decoder IAR model and an encoder-only model using the same hyper-parameters as the decoder-only model (i.e same model but without causal masking).

We evaluated these models both in the NAR and IAR generation setting using either 1 or M iterative steps (results not reported). Both settings produced the same results. That is, once, the model learnt the solution, it could generate the entire arm in one step just as well as over M steps. In principle, the IAR models should be able to first generate t in the last position, condition on it, and then just generate the arm in reverse order via CHC – which should be much easier than learning the true solution. This did not happen. **It should be disconcerting for practitioners of IAR models that the trivial generation order does not seem to be found.**

We demonstrate that small IAR models are capable of learning the task in Tables 3 and 4. Exp. 11 and 14 may indicate that IAR models are not as performative on the task, however, this is not true

[2]Often called iterative NAR models, which is a misnomer.

ID	Perm.	Dir.	Ctra.	$D = 2$	$D = 3$	$D = 4$	$D = 5$
8	Arm	Fwd.	0	100%	100%	100%	100%
9	Edge	Fwd.	0	0% 0%	0% 0%	0% 0%	0% 0%
10	Arm	Fwd.	1	100%	100%	100%	100%

Table 2: Results using the encoder-decoder AR model. Q pos. is 'End'.

ID	Perm.	Train	Dir.	Ctra.	$D = 2$	$D = 3$	$D = 4$	$D = 5$
11	Arm	IAR	Fwd.	0	100%	100%	82% 0%	82% 0%
12	Edge	IAR	Fwd.	0	0% 0%	0% 0%	0% 0%	0% 0%
13	Arm	IAR	Fwd.	1	100%	100%	100%	100%

Table 3: Results using the CMLM (encoder-encoder) IAR model with IAR training (teacher-forcing).

once contrastive samples are used (Sec 2.4).

2.4 Contrastive samples

Our observations of the failures of the above models lead us to conclude that, in the instances where the model failed to solve the task, the model would overfit. See the top graph of Fig 2 with $D = 2$. Here all trials end up successfully learning the task, which can be seen when the validation accuracy branches off from chance. However, the last example nearly overfits. In the third graph, we see the same experimental setting but with $D = 4$. Here only a single trial succeeded and the rest maintained a stagnant validation accuracy at chance while the training and validation losses diverge.

To prevent this, we experimented with standard methods to combat overfitting such as lowering learning rate, increasing batch size or L2 regularization, etc. without success. This lead to developing an alternative method where we supplement the training data with multiple instances of G but with different target nodes, and hence, different Q and target arms to be generated. This was achieved via expanding each batch with one or more of these *contrastive samples* per original G. These extra instances should act as interference on any spurious training signal. Note, apart from sampling, these are treated as independent and are not part of a contrastive loss. This can be viewed as extra supervisory information applied at the batch-level.

Results of exp. 7, 13, and 16 show that this prevents overfitting and leads to improved success rate across models. The arm-wise decoder-only exp. 7 shows improved rates at $D \in \{4, 5\}$. This can be seen in the first and third plots in Fig 2 in contrast to their corresponding second and forth plots where we see the validation loss tracks the training loss instead of diverging when provided with contrastive samples. This lead to learning the solutions in less epochs in the $D = 2$ case and lead to 10/11 instead of 1/11 of the trials succeeding in the $D = 4$ case.

3 Limitations and Future work

The major limitation to this extended abstract is that the experiments with contrastive samples are in progress. We included partial and preliminary results of this method in order to show the future direction of this work. Despite this, we have already demonstrated a number of interesting findings: 1) We independently reproduced the empirical results of Bachmann and Nagarajan (2024). 2) We show that the CHC is a red-herring, if the problem is due to loss of intermediate training signal. 3) We show that retaining the arm structure leads to improved results and that combining this with a better ordering of Q and G makes the task solvable. We take this as strong evidence that poor representation is a key factor in why the task is difficult. 4) We show that NAR/IAR models can successfully solve the task in limited settings, and describe a surprising failing to these models. Finally, 5), we show that overfitting is a key issue and provide initial positive results in tackling this.

We set out the hypothesis that trials where the training and validation losses do not diverge will succeed given sufficient training time. However, that has yet to be shown. It is an open question if contrastive samples will always lead to finding the solution given sufficient time or if this scales with D and M. More importantly, it is an open question as to why the models prefer to overfit instead of finding the correct solution. On lead we have is that the task hinges on a single token which determines the necessary latent representation of G, and this seems to have large consequences on the behaviour

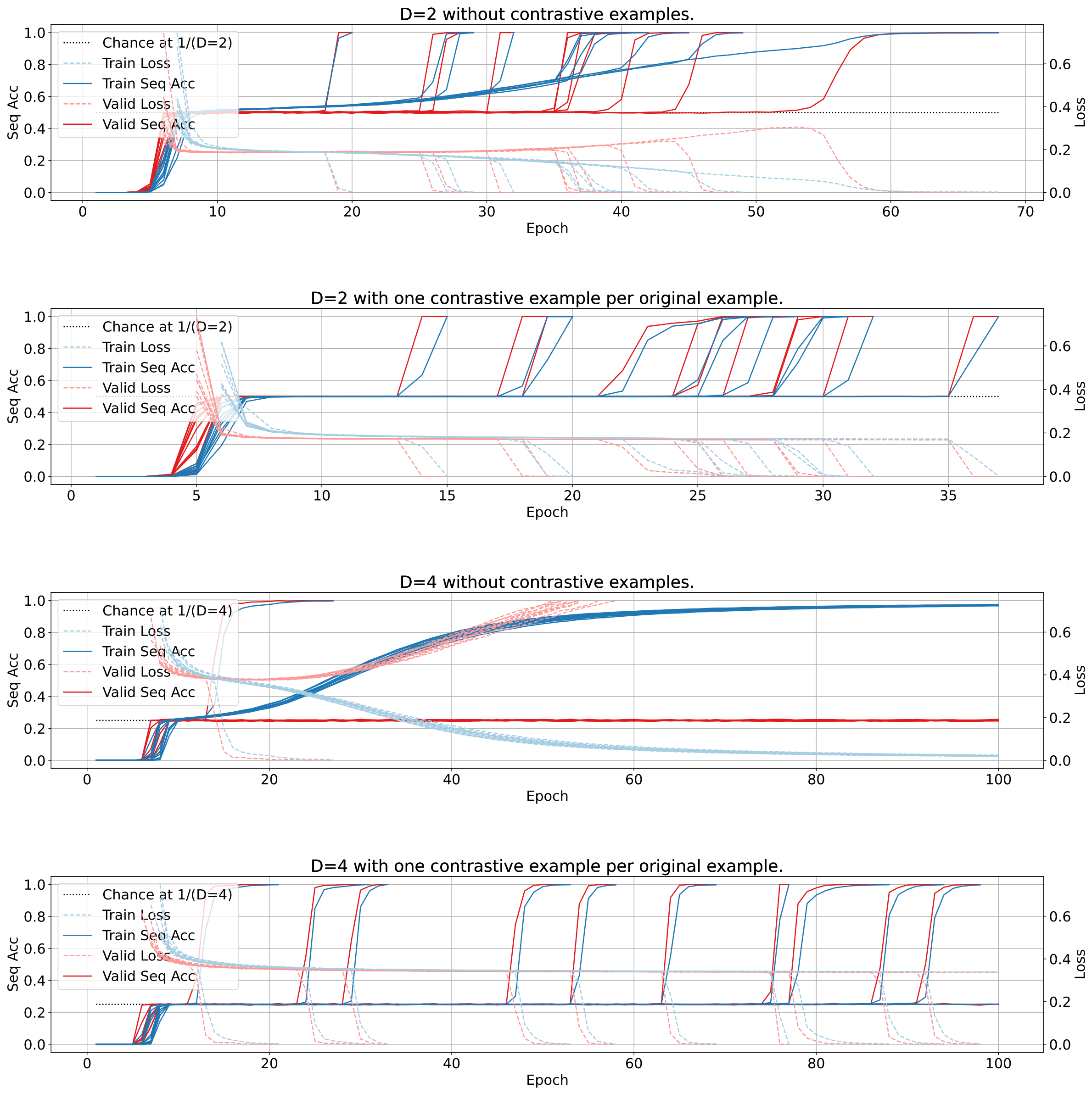

Figure 2: Plots 1 and 3 visualize the training of the experiments of row/exp. 4 in Table 1 where $D = 2$ and $D = 4$ respectively. Plots 2 and 4 visualize the corresponding experiments in row/exp. 7 when constrictive exampled are employed. Each plot shows the loss and sequence accuracy across all 11 trials of the given experiment for both the training and validation partitions. When a trial succeeds in finding the desired solution, the sequence accuracy spikes to 100% and the validation loss drops to near-zero. The loss is cutoff at 0.75 for visibility.
In plot 1, all trials succeed, however, when D is increased to 4, only 1/11 trials succeed as shown in plot 3. Here we see that the training and validation losses diverge shortly after epoch 20, resulting in overfitting. In plot 4, the use of contrastive samples prevents this divergence, leading to 10/11 trials succeeding, with the remaining trial not finding the solution within the 100 epoch limit.

ID	Perm.	Train	Dir.	Ctra.	$D = 2$		$D = 3$		$D = 4$		$D = 5$	
14	Arm	IAR	Fwd.	0	100%		82%	0%	36%	0%	9%	0%
15	Edge	IAR	Fwd.	0	36%	0%	0%	0%	0%	0%	0%	0%
16	Arm	IAR	Fwd.	1	100%		100%		100%		91%	9%

Table 4: Results using the encoder-only IAR model.

of the learning algorithm and the difficulty of learning a correct solution. This indicates that the issue may stem from the sensitivity of the task to the target token (Hahn et al., 2021; Chen et al., 2023; Chakraborty et al., 2023; Bhattamishra et al., 2023; Hahn and Rofin, 2024).

A large and important part of the scientific process is testing hypotheses and putting forth counter arguments. If, as we believe, the CHC hypothesis is incorrect, then the broader discussion of Bachmann and Nagarajan (2024) concerning limitations of next-token prediction is not be supported by their findings. This is the main contribution of this work, even if, we do not have a full solution or replacement hypothesis. Even if the CHC hypothesis is wrong, the path-star task is still a seemingly trivial but deceptively difficult problem, making it worthy of study. In addition to questioning the CHC hypothesis, we make headway into the mystery of the path-start task by demonstrating multiple simplifications or alternatives of the task which make it (more) solvable. Finally, we introduce a contrastive method which has early indications of helping to solve the task, however, open questions remain as to why this method works and why it is necessary in the first place.

3.1 Post-submission findings

Between submission and acceptance, new findings have come to light, which we summarize here.

1) Constrastive samples are necessary but not sufficient to solve the task consistently. We find that, as each example is randomly sampled, there will be $|V|^{D(M-1)+1} \times D$ possible graph-target pairs to sample from. Thus it is easy to see why the models would overfit as they can learn to make many possible spurious correlations between any unique node or combination of nodes in the source-side and the targets. Contrastive samples will alleviate these spurious correlations by indicating that the targets depend on a single token only. Above, we wondered if contrastive samples will always lead to finding the solution given sufficient time. We found this is not true. However, we also conduct an analysis of the task using RASP (described below), which lead to a counter-intuitive result that, even though overfitting is an issue, we need to increase the size of the models to solve the task.

2) The RASP programming language is a formal computation model used to verify if a transformer is capable of solving a given symbolic (non-numerical) task where the existence of a valid RASP program proves there exists at least one transformer which can (Weiss et al., 2021; Zhou et al., 2024). We conduct a RASP analysis which lead us to find to many new insights to the task. We can formally prove, via the existence of RASP programs, that the task is solvable via transformers both using non-causal and causal models. Due to the graph structure, we find that the simplest RASP programs which solve the task with edge-wise permutation require $\mathcal{O}(M)$ number of layers in order to route information about leading nodes to final nodes (or vise versa). We show a $\mathcal{O}(\log M)$ algorithm exists, but conjecture than no $\mathcal{O}(1)$ exists. This would mean that the task will not be generalizable to higher values of M. However, the problem is easily solvable with a $\mathcal{O}(1)$ algorithm when using arm-wise permutation, which explains why the task is solvable under the simpler conditions demonstrated above. We also find that using a causal decoder makes the task harder as it can only route information into the future, which requires different rules depending on if the connecting edge is before or after a current edge when routing.

3) Given the RASP analysis we increased the number of layers of each model. This lead to worse in overfitting in all models, except the encoder-only model, which then is able to consistently solve the task. Both this model and the CMLM (enocder-encoder) model employ the IAR training method. The fact that it is just the encoder-only model which works indicates that the reason why it works is not due to the training method. It is an open mystery why it is only this model which can consistently solve the task.

Please look for an updated pre-print (available soon) or contact arvie@cs.toronto.edu.

Acknowledgments

We thank our reviewers as well as Siavash Kazemian, Gagandeep Singh, and Shunan Zhao for their insightful feedback.

Resources used in preparing this research were provided, in part, by the Province of Ontario, the Government of Canada through CIFAR, and companies sponsoring the Vector Institute

References

Kushal Arora, Layla El Asri, Hareesh Bahuleyan, and Jackie Cheung. 2022. Why exposure bias matters: An imitation learning perspective of error accumulation in language generation. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 700–710, Dublin, Ireland. Association for Computational Linguistics.

Gregor Bachmann and Vaishnavh Nagarajan. 2024. The pitfalls of next-token prediction. In *ICLR 2024 Workshop: How Far Are We From AGI*.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. *Advances in neural information processing systems*, 28.

Satwik Bhattamishra, Arkil Patel, Varun Kanade, and Phil Blunsom. 2023. Simplicity bias in transformers and their ability to learn sparse Boolean functions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5767–5791, Toronto, Canada. Association for Computational Linguistics.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, et al. 2023. Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv preprint arXiv:2303.12712*.

Mohna Chakraborty, Adithya Kulkarni, and Qi Li. 2023. Zero-shot approach to overcome perturbation sensitivity of prompts. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5698–5711, Toronto, Canada. Association for Computational Linguistics.

Xinyun Chen, Ryan A Chi, Xuezhi Wang, and Denny Zhou. 2024. Premise order matters in reasoning with large language models. *arXiv preprint arXiv:2402.08939*.

Yanda Chen, Chen Zhao, Zhou Yu, Kathleen McKeown, and He He. 2023. On the relation between sensitivity and accuracy in in-context learning. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 155–167, Singapore. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Marjan Ghazvininejad, Omer Levy, Yinhan Liu, and Luke Zettlemoyer. 2019. Mask-predict: Parallel decoding of conditional masked language models. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6112–6121, Hong Kong, China. Association for Computational Linguistics.

Albert Gu and Tri Dao. 2023. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*.

Jiatao Gu, James Bradbury, Caiming Xiong, Victor OK Li, and Richard Socher. 2018. Non-autoregressive neural machine translation. In *International Conference on Learning Representations*.

Jiatao Gu and Xiang Kong. 2021. Fully non-autoregressive neural machine translation: Tricks of the trade. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 120–133, Online. Association for Computational Linguistics.

Michael Hahn, Dan Jurafsky, and Richard Futrell. 2021. Sensitivity as a complexity measure for sequence classification tasks. *Transactions of the Association for Computational Linguistics*, 9:891–908.

Michael Hahn and Mark Rofin. 2024. Why are sensitive functions hard for transformers? *ACL*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Jason Lee, Elman Mansimov, and Kyunghyun Cho. 2018. Deterministic non-autoregressive neural sequence modeling by iterative refinement. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1173–1182, Brussels, Belgium. Association for Computational Linguistics.

Yi Liao, Xin Jiang, and Qun Liu. 2020. Probabilistically masked language model capable of autoregressive generation in arbitrary word order. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 263–274, Online. Association for Computational Linguistics.

Giovanni Monea, Armand Joulin, and Edouard Grave. 2023. Pass: Parallel speculative sampling. *arXiv preprint arXiv:2311.13581*.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. *arXiv preprint arXiv:1904.01038*.

Lihua Qian, Hao Zhou, Yu Bao, Mingxuan Wang, Lin Qiu, Weinan Zhang, Yong Yu, and Lei Li. 2021. Glancing transformer for non-autoregressive neural machine translation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1993–2003, Online. Association for Computational Linguistics.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2016. Sequence level training with recurrent neural networks. In *4th International Conference on Learning Representations, ICLR 2016*.

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

Chunqi Wang, Ji Zhang, and Haiqing Chen. 2018. Semi-autoregressive neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 479–488, Brussels, Belgium. Association for Computational Linguistics.

Gail Weiss, Yoav Goldberg, and Eran Yahav. 2021. Thinking like transformers. In *International Conference on Machine Learning*, pages 11080–11090. PMLR.

Ronald J Williams and David Zipser. 1989. A learning algorithm for continually running fully recurrent neural networks. *Neural computation*, 1(2):270–280.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*, 32.

Hattie Zhou, Arwen Bradley, Etai Littwin, Noam Razin, Omid Saremi, Joshua M. Susskind, Samy Bengio, and Preetum Nakkiran. 2024. What algorithms can transformers learn? a study in length generalization. In *The Twelfth International Conference on Learning Representations*.

Barret Zoph, Colin Raffel, Dale Schuurmans, Dani Yogatama, Denny Zhou, Don Metzler, Ed H. Chi, Jason Wei, Jeff Dean, Liam B. Fedus, Maarten Paul Bosma, Oriol Vinyals, Percy Liang, Sebastian Borgeaud, Tatsunori B. Hashimoto, and Yi Tay. 2022. Emergent abilities of large language models. *TMLR*.

A The Clever Hans Phenomenon

In Fig 3 we show how the CHC appears during training. Here we see that the first token to fit to 100% accuracy is the given start node, s. The next token is the given target node, t. While this might seem strange as it is generated at the end of the sequence, this token is actually easily predicable since the model can infer that the target token should always be placed in the M^{th} position. This is because there is no requirement that predictions be generalizable to different arm lengths and hence the target token is always in the M^{th} position. This is explicitly done to maintain that the test data is in-domain with the training data. Next we see that all other non-leading nodes fit via the CHC. As no trials succeeded in this experiment, the validation accuracy of the leading token becomes stagnant at chance, while the training accuracy improves.

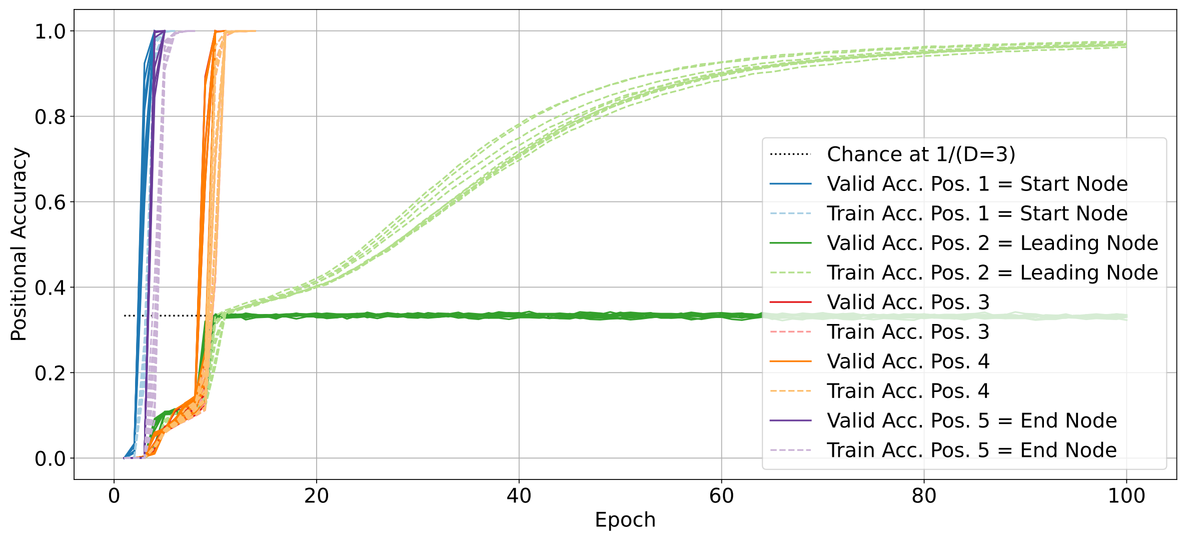

Figure 3: The appearance of the Clever Hans cheat over training. Data corresponds to row/exp. 1 of Table 1, where D=3, $M = 5$.