The Pitfalls of Next-Token Prediction

Gregor Bachmann [*1] **Vaishnavh Nagarajan** [*2]

Abstract

Can a mere next-token predictor faithfully model human intelligence? We crystallize this emerging concern and correct popular misconceptions surrounding it, and advocate a simple multi-token objective. As a starting point, we argue that the two often-conflated phases of next-token prediction — autoregressive inference and teacher-forced training — must be treated distinctly. The popular criticism that errors can compound during autoregressive inference, crucially assumes that teacher-forcing has learned an accurate next-token predictor. This assumption sidesteps a more deep-rooted problem we expose: in certain classes of tasks, teacher-forcing can simply fail to learn an accurate next-token predictor in the first place. We describe a general mechanism of how teacher-forcing can fail, and design a minimal planning task where both the Transformer and the Mamba architecture empirically fail in that manner — remarkably, despite the task being straightforward to learn. Finally, we provide preliminary evidence that this failure can be resolved using *teacherless* training, a simple modification using dummy tokens that predicts multiple tokens in advance. We hope this finding can ground future debates and inspire explorations beyond the next-token prediction paradigm. We make our code available under `https://github.com/gregorbachmann/Next-Token-Failures`

1. Introduction

Long after its inception in the seminal work of Shannon (1948; 1951), next-token prediction has made its way into the core of the modern language model. But despite its long list of achievements, there is a small but growing belief that a next-token predicting model is merely an impressive *improv* artist that cannot truly model human thought. Humans, when navigating the world, meticulously imagine, curate and backtrack plans in their heads before executing them. Such strategies are unfortunately not explicitly built into the backbone of the present-day language model. This criticism has been floating around as an informal viewpoint (LeCun, 2024; Bubeck et al., 2023). Our paper is aimed at crystallizing this intuitive criticism, clarifying popular misconceptions, and developing new core arguments for the next-token prediction debate.

Let us start by making more precise, what it means to say that human-generated language, or problem-solving, does not follow next-token prediction. When formalizing this, we hit an immediate roadblock: isn't every sequence generation task possible autoregressively? Put differently, an optimist would say, every distribution over a sequence of tokens can be captured by an appropriately sophisticated next-token predictor simulating the chain rule of probability i.e., $\mathbb{P}(r_1, r_2, \ldots) = \prod_i \mathbb{P}(r_i | r_1 \ldots r_{i-1})$. Thus, the autoregressivity in our systems is not antithetical to learning human language, after all.

Although this argument is compelling, a pessimist would worry, realistically, even with minor imperfections in the next-token predictor, the accuracy may fall spectacularly for long sequences (Kääriäinen, 2006; Ross & Bagnell, 2010; LeCun, 2024; Dziri et al., 2024). Say, even if every next-token error is as little as 0.01, the probability of encountering an erroneous token exponentially compounds along the way, and by the end of 200 tokens, blows up to 0.86.

This is a simple and powerful observation. Yet, we explain why this does not completely capture the intuition that next-token predictors may be poor planners. Crucially, this argument does not carefully distinguish between the two types of next-token prediction: inference-time autoregression (where the model consumes its own previous outputs as inputs), and training-time *teacher-forcing* (Williams & Zipser, 1989) (where the model is taught to predict token-by-token consuming all previous ground truth tokens as inputs). Framed this way, the compounding of errors only pinpoints a superficial failure to execute a plan during *inference*. It leaves open the possibility that we may have still learned a near-perfect next-token predictor; perhaps, with an appropriate post-hoc wrapper that verifies and backtracks, we can elicit the right plan without compounding errors.

[*]Equal contribution [1]ETH Zürich, Switzerland [2]Google Research, US. Correspondence to: Gregor Bachmann <gregorb@ethz.ch>, Vaishnavh Nagarajan <vaishnavh@google.com>.

Proceedings of the 41^{st} International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

Drawing this distinction allows us to articulate a much more concerning possibility: is it safe to assume that next-token based learning (teacher-forcing) always learns an accurate next-token predictor? We identify this is not always the case. Consider a task where we expect the model to witness a problem statement $\boldsymbol{p} = (p_1, p_2 \ldots,)$ and produce the ground truth response tokens $(r_1, r_2, \ldots)$. Teacher-forcing trains the model to produce each token r_i by not only providing the problem statement $\boldsymbol{p}$ but also by revealing part of the ground truth $r_1, \ldots r_{i-1}$. Depending on the task, we first argue that this can induce shortcuts that use the revealed prefix of the ground truth answer to spuriously fit future answer tokens. We call this the *Clever Hans cheat.* [1] Next, while the later tokens (r_i for large i) become easy to fit by the Clever Hans cheat, in contrast, the earlier answer tokens (say, r_0, r_1 etc.,) become harder to learn. This is because they no longer come with any supervision about the full answer — part of the supervision is lost to the Clever Hans cheat. We argue that these two flaws would together arise in "lookahead tasks": tasks that require implicitly planning a later token in advance of an earlier token. In such tasks, teacher-forcing would result in a highly inaccurate next-token predictor that would struggle to generalize to unseen problems $\boldsymbol{p}$, even those sampled in-distribution.

Empirically, we demonstrate that the above mechanism leads to complete in-distribution failure in a path-finding setup on a graph, that we propose as a minimal lookahead task. We design our setup in a way that it is demonstrably straightforward to solve, implying that the failure of any model is remarkable. Yet, we observe failure for both the Transformer (Vaswani et al., 2017) and the Mamba architecture, a structured state space model (Gu & Dao, 2023). We then point towards a very simple multi-token modification called *teacherless* training — an idea that has appeared in other contexts (Tschannen et al., 2023; Monea et al., 2023; Zhao et al., 2023) — which predicts multiple future tokens and is able to circumvent this failure in some settings. Thus, we pinpoint a precise and easy-to-learn scenario where, rather than properties that are criticized in existing literature — like convolution or recurrence or autoregressive inference (see §6), — it is next-token prediction during training that is at fault.

We hope that these findings inspire and set future debates around next-token prediction on solid ground. In particular, we believe that the failure of the next-token prediction objective on our straightforward task casts a shadow over its promise on more complex tasks (such as say, learning to write stories). We also hope that this minimal example of failure and the positive results on teacherless training can motivate alternative paradigms of training.

We summarize our contributions below.

1. We consolidate existing critiques against next-token prediction and crystallize new core points of contention (§6 and §3, §4).
2. We identify that the next-token prediction debate must not conflate autoregressive inference with teacher-forcing. Both lead to vastly different failures (§3,§B).
3. We conceptually argue that in lookahead tasks, next-token prediction during training (i.e., teacher-forcing) can give rise to problematic learning mechanisms that are detrimental to even in-distribution performance (§4).
4. We design a minimal lookahead task (§4.1). We empirically demonstrate the failure of teacher-forcing for the Transformer and Mamba architectures, despite the task being easy to learn (§5).
5. We identify that a teacherless form of training that predicts multiple future tokens at once — proposed in Monea et al. (2023) for orthogonal inference-time efficiency goals — shows promise in circumventing these training-time failures in some settings (§5, Eq 4). This further demonstrates the limits of next-token prediction.

2. The Two Modes of Next-Token Prediction

Consider a set of tokens $\mathcal{V}$. Let $\mathcal{D}$ be a ground truth distribution over sequences that consist of a prefix $\boldsymbol{p}$ and a response $\boldsymbol{r}$, denoted as $\boldsymbol{s} = \boldsymbol{p}, \boldsymbol{r}$ where $\boldsymbol{p} = (p_1, p_2, \ldots,) \in \mathcal{V}^{L_{\text{pref}}}$ and $\boldsymbol{r} = (r_1, r_2, \ldots) \in \mathcal{V}^{L_{\text{resp}}}$. We assume sequences of fixed length merely for simplicity.

For any sequence $\boldsymbol{s}$, let $\boldsymbol{s}_{<i}$ denote the first $i-1$ tokens of $\boldsymbol{s}$, and $\boldsymbol{s}_{i<}$ the tokens following the ith token. Note that $\boldsymbol{s}_{<1}$ is the empty prefix. With an abuse of notation, let $\mathbb{P}_{\mathcal{D}}(s_i|\boldsymbol{s}_{<i})$ denote the ground truth probability mass on s_i being the ith token given the prefix $\boldsymbol{s}_{<i}$. Consider a next-token-predicting language model $\texttt{LM}_\theta$ (with parameters θ) such that $\texttt{LM}_\theta(\hat{s}_i = s_i; \boldsymbol{s}_{<i})$ is the probability that the model assigns to the ith output $\hat{s}_i$ taking the value s_i, given as input the sequence $\boldsymbol{s}_{<i}$. Note that the next-token predictor only defines the probability for a single future token given an input, but not the joint probability of multiple future tokens. This joint probability is axiomatically defined analagous to the chain rule of probability:

$$\texttt{LM}_\theta(\hat{\boldsymbol{r}} = \boldsymbol{r}\ ; \boldsymbol{p}) \coloneqq \prod_{i=1}^{L_{\text{resp}}} \texttt{LM}_\theta\left(\hat{r}_i = r_i; \boldsymbol{p}, \boldsymbol{r}_{<i}\right) \qquad (1)$$

where $\hat{\boldsymbol{r}} = \boldsymbol{r}$ denotes an exact token-by-token match.

[1] *Clever Hans* (Pfungst & Rahn, 1911) was a famous show horse that could solve simple arithmetic tasks by repeatedly tapping with his hoof until he reached the correct count. It turns out however, Clever Hans did not really solve the problem, but merely stopped tapping upon detecting certain (involuntary) cues from his coach. Clever Hans' answers were wrong when the coach was absent.

To train the above model, two distinct types of next-token prediction are used. First, during inference, for a given prefix, we autoregressively sample from the model token-by-token, providing as input the prefix and all previously-generated tokens. Formally,

Definition 1. ***(Inference-time next-token prediction via autoregression)*** *Autoregressive inference is a form of inference-time next-token prediction in that to generate a response $\hat{\boldsymbol{r}}$, we iterate over $i = 1, \ldots, L_{resp}$, to sample the next token $\hat{r}_i$ with the distribution given by $\texttt{LM}_\theta(\hat{r}_i \,; \boldsymbol{p}, \hat{\boldsymbol{r}}_{<i})$. We denote this as $\hat{\boldsymbol{r}} \overset{\text{ag}}{\sim} \texttt{LM}_\theta(\cdot \,; \boldsymbol{p})$.*

There is also a second phase of next-token prediction, one that is applied during the training process, called *teacher-forcing*. Here, instead of feeding the model its own output back as input, the model is fed with prefixes of the *ground truth* response $\boldsymbol{r}_{<i}$. Meanwhile, the model is assigned as supervisory target, r_i, the next ground truth token. Then, the model maximizes a sum of next-token log-probabilities:

Definition 2. ***(Training-time next-token prediction via teacher-forcing)*** *Teacher-forced training is a form of training-time next-token prediction in that we find parameters θ that maximize the next-token log-probability sum:*

$$\begin{aligned} \mathcal{J}_{next\text{-}token}(\theta) &= \mathbb{E}_{(\boldsymbol{p},\boldsymbol{r})\sim\mathcal{D}}\left[\log \texttt{LM}_\theta\left(\hat{\boldsymbol{r}} = \boldsymbol{r} \,; \boldsymbol{p}\right)\right] \\ &= \mathbb{E}_{\mathcal{D}}\Big[\sum_{i=1}^{L_{resp}} \log \texttt{LM}_\theta\left(\hat{r}_i = r_i; \boldsymbol{p}, \boldsymbol{r}_{<i}\right)\Big] \quad (2) \end{aligned}$$

The key property of the objective is that we extract the model's output, *allowing the model access to the ground truth response preceding the current token.* This property will be crucial to the failure we describe in §4.

3. Failure due to Auto-Regressive Inference

A broad criticism against next-token predictors is that intuitively these models are not explicitly designed to plan ahead, and during inference, they do not know how to recover from their own errors. This discourse has been fragmented in literature. Furthermore, the umbrella term "next-token prediction" is used interchangeably with "autoregressive architecture". Our goal is to analyze these intuitions more systematically, and be careful about distinguishing between the two phases of next-token prediction: teacher-forcing and autoregression. A key insight we will arrive at is that existing arguments capture only a part of the intuitive concern that next-token predictors may not be able to plan.

The chain-rule-of-probability defense: We first outline what is arguably the most tempting defense for next-token prediction: the chain rule of probability always promises us a next-token predictor that can fit our distribution.

Fact 1. ***(Every sequence distribution can be represented by a next-token predictor)*** *By the chain rule of probability we have $\mathbb{P}_{\mathcal{D}}(\boldsymbol{r} \mid \boldsymbol{p}) = \prod_{i=1}^{L_{resp}} \mathbb{P}_{\mathcal{D}}(r_i \mid \boldsymbol{p}, \boldsymbol{r}_{<i})$. Therefore, define a next-token predictor $\texttt{LM}$ such that for every valid value of $i, \boldsymbol{p}$, and $\boldsymbol{r}$, we have $\texttt{LM}(\hat{r}_i = r_i \,; \boldsymbol{p}, \boldsymbol{r}_{<i}) := \mathbb{P}_{\mathcal{D}}(r_i | \boldsymbol{p}, \boldsymbol{r} < i)$. Then, sampling $\boldsymbol{r} \sim \mathcal{D}|\boldsymbol{p}$, is equivalent to autoregressively sampling $\boldsymbol{r} \overset{\text{ag}}{\sim} \texttt{LM}(\cdot \,; \boldsymbol{p})$.*

The cleverness of this argument lies in the fact that it can apply to *any* imaginable distribution. Thus, as long as the next-token predictor is sufficiently expressive (by scaling up the context, memory and compute), it can model both natural language and problem-solving. Thus, it may seem that next-token predictors are not antithetical to planning-based tasks, after all.

The snowballing errors criticism: A skeptic would however raise the following concern. Regardless of the abundance of computational resources, realistic models may still predict the next token with a slight probability of error in each step; these error probabilities may then exponentially accumulate over time. This has been formalized in various contexts, from that of autoregressive models LeCun (2024), to that of the limits of Transformers in compositional tasks Dziri et al. (2024), and in a different form, in much earlier work in imitation learning and structured prediction Kääriäinen (2006); Ross & Bagnell (2010) (see §6). We present a minimal formalization of this below:

Failure 1. ***(Snowballing error due to autoregressive inference)*** *Consider a model $\texttt{LM}_\theta$, prefix $\mathbf{p}$ and a unique ground truth response $\mathbf{r}$ such that the next-token error obeys*

$$\forall i \leq L_{resp},\ \texttt{LM}_\theta\left(\hat{r}_i \neq r_i; \boldsymbol{p}, \boldsymbol{r}_{<i}\right) \approx \epsilon. \quad (3)$$

Then, for $\hat{\boldsymbol{r}} \overset{\text{ag}}{\sim} \texttt{LM}_\theta(\cdot \,; \boldsymbol{p})$ the probability that the generated response exactly matches the ground truth $\boldsymbol{r}$ obeys

$$\mathbb{P}(\hat{\boldsymbol{r}} = \boldsymbol{r}) \approx (1 - \epsilon)^{L_{resp}}.$$

We argue that the snowball failure mode only indicates how an autoregressive model can fail to *execute* a plan during inference-time. It does not preclude the possibility that the model may have *learned* a good plan that it simply fails to execute during inference. Concretely, it may still be possible that, at each step, the model has high accuracy of predicting a next token that is consistent with a good plan (as assumed in Eq 3). Depending on the setting, one can potentially exploit this accuracy to elicit a good plan during inference. For instance, one may be able to use a post-hoc wrapper that verifies whether an error has taken place, then backtracks and executes a different action. One may even simulate backtracking using more elaborate techniques such as tree-of-thought (Wei et al., 2022; Yao et al., 2023a; Besta et al., 2024; Yao et al., 2023b), or using the model to give itself feedback (Madaan et al., 2024; Huang et al., 2022; Shinn et al., 2023) to elicit the plan that the model has learned.

Thus, the snowball failure mode captures what is primarily a shortcoming of an autoregressive architecture. Likewise,

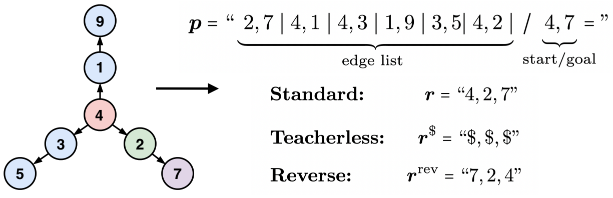
9
1
4
3
2
5
7
p = " 2,7 | 4,1 | 4,3 | 1,9 | 3,5| 4,2 | / 4,7 ="
edge list
start/goal
Standard: r = "4, 2, 7"
Teacherless: $r^\$$ = "\$, \$, \$"
Reverse: r^{rev} = "7, 2, 4"

Figure 1. Illustration of a path-star graph. The prefix $\boldsymbol{p}$ represents the adjacency list and the (central) start and goal node. The target is represented by $\boldsymbol{r}$. Under "standard" teacher-forcing, we condition the model on prefixes of $\boldsymbol{r}$ to predict $\boldsymbol{r}$. But in §5 we explore alternatives where we train without a teacher (condition on $\boldsymbol{r}^\$$ and predict $\boldsymbol{r}$) or train with a reversal (condition on and predict $\boldsymbol{r}^{\text{rev}}$).

the chain-rule-of-probability defense captures only the expressive power of an autoregressive architecture. Neither of these arguments address the possibility that learning with next-token prediction may itself have shortcomings in learning how to plan. In this sense, we argue that existing arguments capture only a part of the intuitive concern that next-token predictors fare poorly at planning.

4. Failure due to Teacher-Forcing

Can a model trained to predict the next token, fail to predict the next token with high accuracy during test-time? Mathematically, this would mean showing that a model trained with the teacher-forcing objective of Eq 2 has high next-token prediction error on the very distribution it was trained on (thus breaking the assumption in Eq 3 of the snowballing failure mode). Consequently, no post-hoc wrapper can salvage a plan out of the model. The goal of this section is to conceptually argue that this failure can happen for lookahead tasks: tasks that implicitly require computing a future token in advance before an earlier token.

As a running example for our argument, we design a path-finding problem on a simple class of graphs. We view this example as a minimal setting that captures the core essence of what it means to solve problems with lookahead, without irrelevant confounding factors. This task is also demonstrably straightforward to solve, as we will see, thus making any observed failures remarkable. Thus we view this running example as a template for an intuitive argument that can be made about teacher-forced models on more general and harder problems that require lookahead.

4.1. Path-Finding on Path-Star Graphs: A Minimal and Easy Lookahead Task

Consider a path-finding problem on a directed graph G with a set of nodes $\{v_{\texttt{start}}, v_{\texttt{goal}}, v_1, v_2, \ldots\}$. The graph is a "path-star" graph with $v_{\texttt{start}}$ as the central node, with at least 2 paths (each of length $l \geq 2$ edges) emanating from it, with a unique path ending in $v_{\texttt{goal}}$. The task is to find a path from $v_{\texttt{start}}$ to $v_{\texttt{goal}}$. Correspondingly, we assume that the distribution $\mathcal{D}$ is over sequences where the prefix $\boldsymbol{p}$ represents a (randomly generated) graph, and the response represents the path from the start to the goal. In particular, we sample a graph G which is represented as an adjacency list as $\texttt{adj}(G) = e_1, e_2, \ldots$ where each edge $e = (v, v')$ is represented such that v' farther away from $v_{\texttt{start}}$ than v. We then set the prefix as $\boldsymbol{p} = (\texttt{adj}(G), v_{\texttt{start}}, v_{\texttt{goal}})$ so the model knows what the graph, and the desired start and goal states are. The ground truth response $\boldsymbol{r}$ corresponds to the sequence of vertices $\boldsymbol{r} = v_{\texttt{start}}, \ldots v_{\texttt{goal}}$ on the start-to-goal path. We visualize this construction in Fig. 1.

The straightforward lookahead solution. Ideally, we want the model to learn a mapping from the input $\boldsymbol{p}$ consisting *only* of $(\texttt{adj}(G), v_{\texttt{start}}, v_{\texttt{goal}})$ to an output that is the full path $\boldsymbol{r}$. Two such solutions are possible. One idea is to plan by examining all the paths emanating from $v_{\texttt{start}}$ and choosing the one that ends at $v_{\texttt{goal}}$. But a second, straightforward solution exists: the model simply needs to look ahead at the sequence "right-to-left" and observe that it corresponds to the one unique path starting from $v_{\texttt{goal}}$ and ending at $v_{\texttt{start}}$. After internally computing the path from $v_{\texttt{goal}}$ and reversing it, the model can emit its response.

4.2. Outline of Failure Mechanism

While we will use the path-star example as a running example, we make our claim more generally for problems that require lookahead (such as story-writing, as we will discuss later). In such problems, we claim that teacher-forcing prevents learning the true mechanisms, causing failure. Intuitively, in teacher-forcing, we decompose the learning of $\boldsymbol{p} \rightarrow \boldsymbol{r}$ into multiple problems, one for each token r_i. Specifically, we make the model learn a mapping from the input $(\boldsymbol{p}, \boldsymbol{r}_{<i})$ — not just $\boldsymbol{p}$ — to the output r_i. The additional information $\boldsymbol{r}_{<i}$ in the input, we argue, is problematic and destroys the core challenge in what the model has to learn. Specifically, our argument puts forth two debilitating mechanisms that would together emerge under teacher-forcing (explained over the next two subsections). While, we will empirically verify these mechanisms for path-star graphs in §5, we also provide a discussion of how our ideas apply to a text-based scenario at the end of this section.

4.3. The Clever Hans Cheat

First, and most importantly, by revealing parts of the answer to the model as input, we allow the model to fit the data by *cheating* i.e., by using trivial mechanisms that use the extra information in $\boldsymbol{r}_{<i}$ to produce r_i. Such cheats must especially be abundant for the later tokens (large i) for which a larger prefix is revealed.

To illustrate this in our path-star example, without loss of

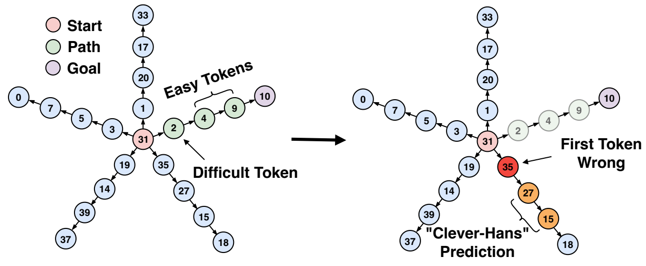

Figure 2. Illustration of the failure of teacher-forcing on a path-star graph. The left image marks the "easy tokens" which can be fit by the Clever Hans cheat (Failure 2a), while the "difficult token" cannot be learned (Failure 2b) due to lost supervision. The right image shows how the model would behave during autoregressive inference, under the absence of the "teacher".

generality, consider a ground truth path that is of the form $\boldsymbol{r} = v_{\texttt{start}}, v_1, v_2, \ldots, v_{\texttt{goal}}$. With a slight abuse of the indexing notation, let $\boldsymbol{r}_{<i} = v_{\texttt{start}}, v_1, \ldots, v_{i-1}$ be the prefix of length i (so we index from 0 instead of 1). Observe that nodes from v_2 onwards, until before $v_{\texttt{goal}}$, have precisely one edge going "away" from $v_{\texttt{start}}$. Thus, consider when the model is given as input, $(\boldsymbol{p}, \boldsymbol{r}_{<i})$ where $\boldsymbol{p} = (\texttt{adj}(G), v_{\texttt{start}}, v_{\texttt{goal}})$, to fit the target v_i. The model first merely needs to scan the adjacency list $\texttt{adj}(G)$ within $\boldsymbol{p}$ for the one edge containing v_{i-1} in the first position. Then, the model only has to predict the other node on that edge as v_i. Note though, this cheat cannot work on fitting the target v_1 given the input $\boldsymbol{r}_{<1} = v_{\texttt{start}}$ since $v_{\texttt{start}}$ has many outward edges — we will scrutinize this node in the next section. We illustrate this difference between v_1 and the remaining tokens as the "easy" vs. "difficult" tokens in Fig. 2.

Crucially, the above cheating mechanism for fitting the easy tokens does not require any lookahead. It is simple, and implementable by an induction head-like module (Olsson et al., 2022). Then, given the well-known simplicity bias of neural networks (Shah et al., 2020), we hypothesize that the later tokens will be quickly fit and ignored during training. This destroys any gradient signal (a.k.a gradient starvation (Pezeshki et al., 2021)) to efficiently learn the "right-to-left" solution — the solution that requires looking at all the tokens in $\boldsymbol{r}$, and learning that they are simply the unique path from $v_{\texttt{goal}}$ spelled in reverse.

We emphasize two key aspects of this cheating behavior. First, these shortcuts are unlike well-known shortcuts (see Remark 5) that map from the original input prefixes $\boldsymbol{p}$ to the ground truth $\boldsymbol{r}$. What we identify is unique to the mapping from the teacher-forced prefix $(\boldsymbol{p}, \boldsymbol{r}_{<i})$ to r_i. Hence, we name this *Clever Hans cheating*. Another notable point is that this does not come from a dearth of samples: even if we had infinite training data at our disposal, the model can still fit the easy tokens of all that data by Clever Hans cheating.

4.4. The Indecipherable Token

Perhaps, not all is lost. While the later tokens may be fit using the Clever Hans cheat, we may still have some of the earlier tokens (for small i), for which such cheats may be unavailable. The supervision from these tokens may eventually coerce the model into learning the true solution. For example, in the path-star task, the model still needs to learn to predict the first node v_1, where it is not possible to fit the training data by the Clever Hans cheat. If not memorize this token on the data, the most general way to fit this token is by actually solving the underlying task.

However, we argue that it is significantly harder for the model to learn the correct solution now. Consider the moment in training when the Clever Hans cheat is perfected. At this point, the model is deprived of information about much of the full solution which was once present as supervisory targets. The model is simply left with the task of mapping the input $\boldsymbol{p}$ to an *incomplete* solution (e.g., the first vertex v_1 in the path-star graph). Recovering the plan in this scenario must first of all be relatively harder from a statistical point of view due to the incomplete supervision. But more importantly, learning this task may become computationally harder, or even intractable under certain assumptions. We provide an informal intuition using the path-star problem below, but this intuition should extend to more general problems as well — indeed, Wies et al. (2023) and other literature on chain-of-thought echo similar negative results about learning from limited supervision (see §H).

Intuitively, our learner has to find an end-to-end algorithm that composes multiple subroutines. For instance, the straightfoward solution consists of l steps: start from the current vertex as $v_{\texttt{goal}}$, and find the preceding vertex in the graph in each subsequent step. Each vertex in this path can be thought of as "intermediate supervision" to learn a corresponding "find-the-adjacent-vertex" subroutine from a space of candidate subroutines.[2] Even if we conservatively assume that there is only a constant-sized space of candidate subroutines $\mathcal{C}$, the end-to-end search space is an exponential space of $|\mathcal{C}|^l$ algorithms composing l subroutines.

Now, after the Clever Hans cheat is in effect, the only supervision for this search is the single-token loss, $-\log \texttt{LM}_\theta\,(\hat{r}_1 = r_1; \boldsymbol{p})$. However, this loss is an "all-or-nothing" loss. Crucially, by the *discrete* nature of the task, even if one subroutine is incorrect, the final answer $\hat{r}_1$ would likely be incorrect on all inputs. For instance, imagine that

[2] As an illustration of what these candidate subroutines could be, imagine that the model can implement an induction head (Olsson et al., 2022) $\texttt{Ind}_k(\boldsymbol{p}, v)$ that finds v in the adjacency list of $\boldsymbol{p}$, and outputs the token that precedes it by k positions. Then the candidate space could be parameterized by k as $\{\texttt{Ind}_k(\boldsymbol{p}, v) | k = 1, 2, \ldots, \}$. For our specific tokenization, the correct subroutine at each of the l steps is the induction head for which $k = 2$.

the first subroutine is incorrect and its output takes us to an arbitrary location on the graph. Then, even if all subsequent subroutines were correct (i.e., they are "find-the-adjacent-vertex" subroutines), the final output would be arbitrary. Thus, we have $\hat{r}_1 = r_1$ precisely for the algorithm where all l subroutines are correct, and $\hat{r}_1 \neq r_1$ for any other choice of the algorithm. For such an all-or-nothing loss surface, the end-to-end learner must necessarily brute-force search the exponential space of algorithms. We encapsulate the overall claim more generally below:

Proposition 3. *Let $\mathcal{C}$ be a set of discrete-output candidate subroutines. Consider learning a task such that (i) it requires composing some l subroutines from $\mathcal{C}$, (ii) the k leading response tokens are sensitive in that even if one subroutine is altered, the first k tokens are each completely altered. Then learning the task with only supervision from the first k ground truth tokens requires exponential time of $\Omega(|\mathcal{C}|^l)$, given the assumptions listed in Remark 1.*

In §5, we will verify experiments to demonstrate that our models indeed fail to learn the Indecipherable Token as a result of Clever Hans cheating, and that conversely, they succeed whenever the Clever Hans cheat is prevented.

4.5. Beyond the Path-Star Setting

Framing our argument more generally, and informally, we argue that teacher-forcing can suffer the following failures in order, especially in tasks that require advance lookahead.

Failure 2a. ***(Clever Hans cheating due to teacher-forcing)*** *Although there is a true mechanism that recovers each r_i from the original prefix $\boldsymbol{p}$, there may be multiple other mechanisms that can recover each token r_i from the teacher-forced prefix $(\boldsymbol{p}, \boldsymbol{r}_{<i})$. These mechanisms may be simpler thus disincentivizing the model from learning the true mechanism.*

Failure 2b. ***(Indecipherable token due to lost supervision)*** *After the Clever Hans cheat is perfected during training, the model is deprived of a part of the supervision (especially, r_i for larger i). This makes it harder and potentially even intractable to learn the true mechanism from the remaining tokens alone.*

As we demonstrate in the next section, the above failures can cause the model to fail on the very distribution it was trained on. This breakdown of planning abilities emerges right from training, and is orthogonal to the Snowballing Failure that is primarily an inference-time issue (See §B).

While the path-star problem provides a concrete, verifiable setting of this failure, it can also help us speculate how such failures could occur in more complex and nebulous tasks. More generally, we expect this failure to occur when there are right-to-left dependencies i.e., a later-appearing token must be planned before an earlier-appearing token. We provide an example below.

Story-writing. Imagine training on novels that take the form of a conflict, followed by a backstory, followed by a resolution of the conflict, utilizing the backstory. Although the story explicitly reads as $v_{\texttt{conflict}}, v_{\texttt{backstory}}, v_{\texttt{resolution}}$, implicitly, one must learn to decide on $v_{\texttt{backstory}}$ before all else. We however conjecture that the teacher-forced model would suffer the Clever Hans cheat wherein it would first learn to fit $v_{\texttt{resolution}}$ using simple deductive skills. With this crucial part of the story lost as supervision, the model can no longer decipher how the remaining pieces relate i.e., how $v_{\texttt{backstory}}$ must be planned in advance of $v_{\texttt{conflict}}$. We conjecture that the resulting model would learn to generate uninteresting stories, interjecting arbitrary conflicts and backstories on a whim, subsequently forcing contrived resolutions upon them. While this hypothesis is not straightforward to empirically test for, we provide a more detailed conceptual illustration in §C.

5. Experimental Verification

In this section, we demonstrate our hypothesized failure modes on the graph path-finding task. We show this in both Transformers and Mamba to demonstrate that these failures are general to teacher-forced models. First, we establish that our teacher-forced models fit the training data but fail in-distribution. Next, we design metrics to quantify the extent to which the two hypothesized mechanisms (Failures 2a, 2b) occur. Finally, we design alternative objectives to intervene and remove each of the two failure modes, to test whether the performance improves. We report additional experiments in §F.4 for an arithmetic task, and in §F.1 quantifying the Snowballing Failure 1. We describe our experimental setting more precisely below.

Dataset. We denote by $G_{d,l}(N)$ for $d, l, N \in \mathbb{N}$, a path-star graph consisting of a center node $v_{\texttt{start}}$ with degree $d \in \mathbb{N}$, meaning there are d different paths emerging from the center node, each consisting of $l - 1$ nodes (excluding the start node). Node values are uniformly sampled from $(\{0, \dots, N-1\})$ where N can be larger than the actual number of nodes in the path-star graph. In every graph, we use the center node as the starting node $v_{\texttt{start}}$ and then pick as $v_{\texttt{goal}}$, the last node of one of the paths chosen uniformly at random. The order of the edges in the adjacency list is randomized. We describe the tokenization in §G.1.

For each experiment, we generate the training and test graphs from the same distribution $\mathcal{D}$, all with the *same* topology of $G_{d,l}(N)$ with fixed d, l and N. Thus, any failure we demonstrate is an *in-distribution* failure, and does not arise from the inability to generalize to different problem lengths (Anil et al., 2022). While the graphs have identical topology, this is not a trivial memorization problem for the model, since the graphs are labeled differently, and the adjacency list randomized — the model *has* to learn a general algorithm. Throughout the experiments, we fix the number of samples to $200k$ and fix the number of node values to $N = 100$ across topologies to enable diverse instantiations of the topology for training and testing.

Models. We evaluate models from two architectural families to highlight that the failures are not tied to a particular architecture but stem from the next-token prediction objective. For Transformers, we use from-scratch GPT-Mini, and pretrained GPT-2 large (Radford et al., 2019). For recurrent models, we use from-scratch Mamba (Gu & Dao, 2023). We optimize using *AdamW* (Loshchilov & Hutter, 2019) until perfect training accuracy. To rule out grokking behaviour (Power et al., 2022), we trained the cheaper models for as long as 500 epochs. More details are in §G.2.

5.1. Observations.

Verifying in-distribution failure. For a given distribution, we evaluate all our teacher-forced models by autoregressively generating solutions, and comparing that solution with the true one for an exact-match. We denote this accuracy as $\texttt{Acc}_{\texttt{ag}}(\texttt{LM}_\theta)$ (see Eq 5) and report it for path-star graphs of varying topologies in Fig. 3 and Table 2. As observed, all models (even when pre-trained) struggle to learn the task accurately. The accuracies are precisely limited to the value when uniformly guessing a path from $v_{\texttt{start}}$ i.e., $\approx \frac{1}{d}$, thus establishing complete in-distribution failure. This is so even when trained to fit sample sizes up to $200k$ to 100% accuracy, and despite the fact that the training and test graphs have identical topology. Next, we quantitatively demonstrate how this stark failure arises from our two hypothesized mechanisms (Failure 2a, 2b).

Verifying Failure 2a (The Clever Hans cheat) We had hypothesized that the teacher-forced model would cheat to fit the training tokens (the ones that follow r_1 in each instance). Specifically, to predict node v_i in the true path, the model can exploit the ground truth node v_{i-1} that is revealed as input. Rather than learning to plan, the model would simply predict the node that is outwardly adjacent to v_{i-1}. To quantify whether this behavior emerges, we "teacher-force" the model with a uniform random neigbhor v_1' of $v_{\texttt{start}}$. We then test whether the model indiscriminately applies the learned Clever Hans cheat here: does the model religiously follow the path that emanates from the neigbhor v_1', not necessarily ending in $v_{\texttt{goal}}$? We measure the exact match of this path on a held-out set as $\texttt{Acc}_{\texttt{cheat}}(\texttt{LM}_\theta)$ (Eq 6).

Empirically, in §F.1, Table 1, we find $\texttt{Acc}_{\texttt{cheat}}(\texttt{LM}_\theta) \approx 100\%$ almost across the board (except for high-degree graphs where training is challenging). This establishes that to fit the training data, the teacher-forced model has exploited the Clever Hans cheat.

Verifying Failure 2b (The Indecipherable Token) Recall that the Clever Hans cheat only applies to all but the first node v_1 after $v_{\texttt{start}}$ lying on the path. After the Clever Hans cheat fits the rest of the path during training, we hypothesized that node v_1 may become impossible to learn since the model is deprived of all information about the subsequent targets. To quantify this behavior, we evaluate how well the model is able to predict the difficult first node, v_1. We measure this on the held-out set and denote this as $\texttt{Acc}_{\texttt{1st}}(\texttt{LM}_\theta)$ (see Eq 7). As shown in Fig. 4 the model achieves a low $\texttt{Acc}_{\texttt{1st}}(\texttt{LM}_\theta)$, approximately $1/d$. Thus, the model indeed fails to identify that v_1 is the one on the path to $v_{\texttt{goal}}$ and instead randomly emitting one of the d neighbors of $v_{\texttt{start}}$.

Removing the Clever Hans cheat via teacherless training (Tschannen et al., 2023; Monea et al., 2023) We now consider a training setup where we prevent Clever Hans cheating (Failure 2a) and examine how learning differs. Concretely, consider modifying teacher-forcing by replacing the *input* $\boldsymbol{r}$ (which reveals the ground truth) with an uninformative input $\boldsymbol{r}^{\$}$, consisting of the same special ("lookahead") token \$ repeated l times. For supervision in the loss, we still use the original target $\boldsymbol{r}$. Thus, the model cannot fit the targets by looking at the prefixes $\boldsymbol{r}_{<i}$ and by predicting the next token v_i via cheating. Instead, the model only has access to the graph description in $\boldsymbol{p}$ to lookahead and fit all the targets v_i for $i = 1, \ldots, l$. Formally, we maximize:

$$\mathcal{J}_{\texttt{t-less}}(\theta) = \mathbb{E}_{\mathcal{D}}\Big[\sum_{i=1}^{L_{\text{resp}}} \log \texttt{LM}_\theta\Big(\hat{r}_i = r_i; \boldsymbol{p}, \boldsymbol{r}^{\$}_{<i}\Big)\Big]. \quad (4)$$

We denote a model trained this way by $\texttt{LM}^{\$}_\theta$ and perform inference simply by conditioning on \$ tokens i.e., to extract $\hat{r}_i$, we feed the uninformative prefix $\boldsymbol{r}^{\$}_{<i}$ as input, rather than autoregressively feeding the output $\hat{\boldsymbol{r}}_{<i}$ as input. We denote the resulting accuracy by $\texttt{Acc}_{\texttt{\$}}(\texttt{LM}^{\$}_\theta)$ (see Eq 9). Our goal however is to evaluate whether forcing the model to lookahead can dodge the Clever Hans cheat, thereby allowing the correct mechanism to be learned.

We report the accuracy of these teacherless models in Fig. 3 and Table 3. Unfortunately, in most cases, the teacherless objective is too hard for the models to even fit the training data, likely because there is no simple cheat to employ here. However, surprisingly, on some of the easier graphs, the models not only fit the training data, but generalize well to test data. This positive result (even if in limited settings) verifies two hypotheses. First, the Clever Hans cheat is indeed caused failure in the original teacher-forced model. Secondly, and remarkably, with the cheat gone, these models are able to fit the first node which had once

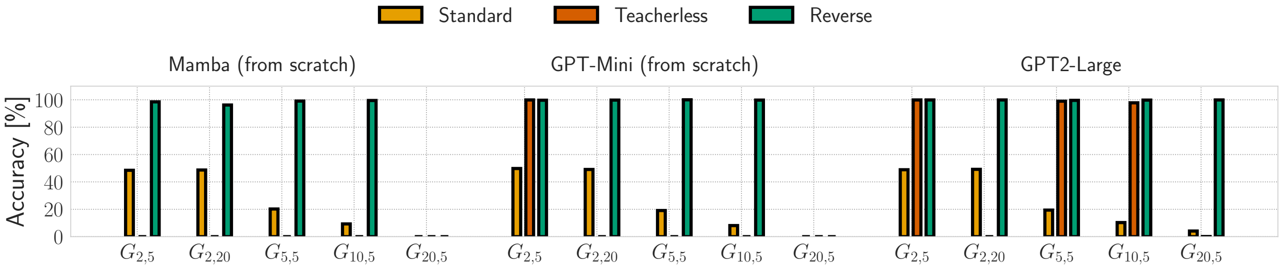

Figure 3. For different architectures, we report the accuracy of the standard teacher-forced model ($\texttt{Acc}_{\texttt{ag}}$, Eq 5), teacherless-trained model's accuracy ($\texttt{Acc}_{\texttt{\$}}$, Eq 9) and accuracy of the model trained with reversed targets ($\texttt{Acc}_{\texttt{rev}}$, Eq 10) evaluated on path-finding a range of graphs (with degree in the first subscript, and path length in the second).

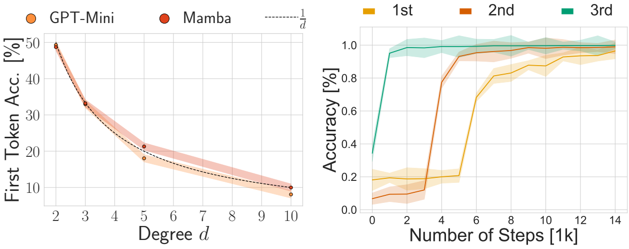

Figure 4. $\texttt{Acc}_{\texttt{1st}}(\texttt{LM}_\theta)$ (in percent $\%$, Eq 7) for path-star graphs of various degrees $d \in \{2, 3, 5, 10\}$ for fixed path length $l = 5$ (left). Individual token accuracies (for v_1, v_2, v_3) for the graph $G_{5,5}$ under teacherless training (Eq 4) with GPT2-large (right).

been indecipherable under teacher-forcing. This verifies our hypothesis that the Clever Hans cheat absorbs away supervision that is critical to learn the first token. At the end of this section, we provide more intuition for how the absence of Clever Hans cheat, allows the teacherless models to solve this task.

Removing the Indecipherable Token failure via path reversal. Back in the teacher-forcing setup, we make a slight change: we train the model to predict the reversal of the true path $\boldsymbol{r}$. Indeed, prior works (Lee et al., 2023; Shen et al., 2023) have proposed reversal in the context of addition tasks as a way of explicitly guiding the next-token predictor to learn a simpler algorithm. Likewise, in our "reversed" path-finding task, the model now needs to predict $v_{\texttt{goal}}$ first and make its way to $v_{\texttt{start}}$; the hope is that since there is only one unique path emanating from $v_{\texttt{goal}}$, there is no planning required. Thus we should never run into an Indecipherable Token. Every next node can be learned as the node that is inwardly adjacent to the previous node.

We display the results in Fig. 3 and Table 4. As expected, we observe that reversing significantly boosts learning, allowing even models trained from scratch to solve the task. This verifies that for the standard model, indecipherability of the first token was indeed a roadblock to successful learning.

5.2. Why the Failure of Teacher-Forcing is Remarkable.

The success of the reversed training (and of teacherless training) make the in-distribution failure of teacher-forcing particularly surprising. When viewed left-to-right, our problem requires complex planning — evaluating multiple paths and selecting the right one — but when viewed right-to-left, the problem is straightforward; the experiments on the reversed formulation confirm that the right-to-left solution is not only expressible by our architectures, but also learnable via gradient descent. Evidently, the left-to-right teacher-forced model is unable to view the problem any differently and falls into the traps outlined in §4.

Intuition for teacherless training. We hypothesize that even teacherless training allows the model to implicitly learn the right-to-left view. Concretely, the teacherless model cannot use the trivial Clever Hans cheat to fit the data, since the ground truth prefixes are not available during training. Nor is it explicitly prescribed to fit the target right-to-left. Instead the model is tasked with using only the graph description in $\boldsymbol{p}$ to fit all the target nodes $\boldsymbol{r}$ (implicitly requiring a lookahead beyond just the next token). Our key intuition is that, in this paradigm, the model would first fit the target token that is simplest to deduce using only information available in the prefix $\boldsymbol{p}$: this is the penultimate vertex r_{l-1} which is the unique token that precedes the goal (and can be discovered using a simple scan of the prefix). Once the model figures this out, the model can similarly work backwards to fit each node r_{i-1} using the previously-fit r_i. (Also see Remark 2)

Our hypothesis is borne out in Fig. 4 where we see that the later tokens achieve higher accuracy earlier, implying that the teacherless model voluntarily learns right-to-left. Thus, the teacherless objective provides an alternative training paradigm that forces models to look ahead, without falling into the various short-sighted pitfalls of next-token prediction, discussed in §4.

6. Related Work

We consolidate the arguments surrounding next-token prediction that has been fragmented over various lines of works. Part of our elaborate survey is deferred to §H.

Arguments in support of next-token prediction. Shannon (1948; 1951); Alabdulmohsin et al. (2024) demonstrate that language has enough redundancy to be conducive for next-token prediction. Empirically, Shlegeris et al. (2022) find that modern language models are surprisingly better than humans at next-token prediction on the text dataset, OpenWebText (Gokaslan & Cohen, 2019). But this does not preclude the possibility that next-token predictors may still be poor at planning. Furthermore, the above result may be confounded by the ability of language models to store more general knowledge than humans.

On the theoretical side, Merrill & Sabharwal (2024); Feng et al. (2023) show that autoregressive Transformers that generate chains of thought have a larger *expressive* power. Most relevant to us is the positive *learnability* results of Malach (2023); Wies et al. (2023) which argue that complex multi-hop tasks that are otherwise unlearnable, become learnable via next-token prediction when there is a preceding chain-of-thought supervision for each hop. Our negative result does not contradict this. In our problem, learning the first token requires an implicit chain of thought (the reversed path) that we do not provide *before* the first token.

Arguments against next-token prediction. The most well-formulated criticism is the snowballing failure mode, which appears scattered in various forms in literature Dziri et al. (2024); LeCun (2024); Kääriäinen (2006); Ross & Bagnell (2010). As explained earlier, this is orthogonal to our failure; indeed, in our setting the error happens "instantaneously" at the beginning, rather than snowball over time.

Our main counterexample can be seen as formalizing an emerging, informal intuition, often worded as "autoregressive next-token predictors are ill-suited for planning tasks". Indeed, Momennejad et al. (2023); Valmeekam et al. (2023a;b;c) report failures on several planning tasks framed as word problems (including path-finding in Momennejad et al. (2023)) and Bubeck et al. (2023) on various arithmetic, summarization and poem/story generation tasks. McCoy et al. (2023) argue that, for such tasks, the performance of the model must greatly depend on its frequency during pretraining. However, we show that even when trained on many samples from a distribution, the next-token predictor can fail on the very distribution.

Our work extends and clarifies this discourse by introducing the Clever Hans cheat and the Indecipherable Token failure. Next, we empirically report our failure modes in both the Transformer (Vaswani et al., 2017) and the Mamba structured state space model (Gu & Dao, 2023). Thus, what we witness is indeed a failure of next-token prediction (and not of the Transformer architecture as some existing criticisms are framed). Importantly, existing literature pins these failures broadly on the next-token prediction paradigm and interchangeably, on the inability of the autoregressive architecture to backtrack. We emphasize the need to differentiate between the two types of next-token prediction (teacher-forcing and autoregressive inference) as they lead to distinct planning-related failures and require distinct solutions.

Going beyond next-token prediction. Various works have explored architectures and objectives that train the backbone to go beyond next-token prediction. This includes non-autoregressive models (Gu et al., 2018), energy-based models (Dawid & LeCun, 2023), diffusion models (Gong et al., 2023), and variants of Transformers learning to either predict future tokens in a different ordering (Li et al., 2021) or all at one go (Qi et al., 2020; Monea et al., 2023), or injecting "lookahead" data (Du et al., 2023a).

Teacherless training was proposed for image-to-text captioning models in Tschannen et al. (2023) under the more generic name of parallel prediction with the similar motivation that the tokens in a caption must purely rely on the image rather than parts of the caption itself. Monea et al. (2023) proposed the same idea as "parallel speculative sampling" for the orthogonal goal improving the inference-time compute. On that note, we clarify that research in parallel decoding too is concerned with predicting multiple future tokens (Stern et al., 2018), the goal is purely inference-time efficiency. Finally, it is worth noting that action chunking (Zhao et al., 2023) in imitation learning is a close counterpart of teacherless training although motivated through different arguments.

7. Conclusion

Next-token prediction lies at the heart of modern language models which have empirically demonstrated tremendous success in wide-ranging tasks. Theoretically too, we know by the chain rule of probability that, next-token predictors can express any distribution over tokens. It is tempting then to view next-token prediction as a formidable approach to modeling intelligence. Our work crystallizes the core arguments around why this optimism may be misplaced.

We emphasize not to conflate the two modes of next-token prediction: autoregressive inference and teacher-forced training. While existing criticisms primarily challenge autoregressive inference, they assume that teacher-forcing learns a good next-token predictor. We challenge this very assumption, finding that even in a straightforward task, there is failure due to teacher-forcing — not due to autoregressive inference or the architecture. This casts a shadow over more complex tasks. For instance, as we speculate in §C, can a model trained to predict the next token of thousands of fiction novels, learn to generate plot twists?

An immediate way to circumvent this, as our reversal experiments suggest, is to train with chain-of-thought supervision, echoing Malach (2023); Wies et al. (2023). However, it is unclear how that is possible in more unstructured tasks like story-writing. To that end, our minimal counterexample and the idea of teacherless training (Monea et al., 2023) may inspire alternative paradigms to next-token prediction in practice. Overall, we hope our analyses provide a solid ground to pursue future debates on next-token prediction.

We point the reader to §A for a discussion of the limitations of our study.

Acknowledgments

We would like to thank Colin Raffel, Tiago Pimentel, and Surbhi Goel for their extensive feedback on a draft of this preprint, especially for pointers to some key references. We thank Garret Tanzer for pointing us to Tschannen et al. (2023) for a prior instance of the idea of teacherless training.

Impact Statement

Our results outline the limits of a foundational technique that lies at the heart of modern AI systems. Naturally, there are many potential downstream societal consequences that would apply at large to such foundational work, none we feel must be specifically highlighted here.

References

Alabdulmohsin, I., Tran, V. Q., and Dehghani, M. Fractal patterns may unravel the intelligence in next-token prediction, 2024.

Allen-Zhu, Z. and Li, Y. Physics of language models: Part 3.2, knowledge manipulation. *arXiv preprint arXiv:2309.14402*, 2023.

Anil, C., Wu, Y., Andreassen, A., Lewkowycz, A., Misra, V., Ramasesh, V. V., Slone, A., Gur-Ari, G., Dyer, E., and Neyshabur, B. Exploring length generalization in large language models. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022*, 2022.

Arkoudas, K. Chatgpt is no stochastic parrot. but it also claims that 1 is greater than 1. *Philosophy & Technology*, 36(3):54, 2023.

Artetxe, M., Du, J., Goyal, N., Zettlemoyer, L., and Stoyanov, V. On the role of bidirectionality in language model pre-training. In *Findings of the Association for Computational Linguistics: EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pp. 3973–3985. Association for Computational Linguistics, 2022.

Bahdanau, D., Brakel, P., Xu, K., Goyal, A., Lowe, R., Pineau, J., Courville, A. C., and Bengio, Y. An actor-critic algorithm for sequence prediction. In *5th International Conference on Learning Representations, ICLR 2017, Conference Track Proceedings*, 2017.

Bender, E. M., Gebru, T., McMillan-Major, A., and Shmitchell, S. On the dangers of stochastic parrots: Can language models be too big? In *FAccT '21: 2021 ACM Conference on Fairness, Accountability, and Transparency, Virtual Event / Toronto, Canada, March 3-10, 2021*, pp. 610–623. ACM, 2021.

Bengio, S., Vinyals, O., Jaitly, N., and Shazeer, N. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015*, pp. 1171–1179, 2015.

Besta, M., Blach, N., Kubicek, A., Gerstenberger, R., Podstawski, M., Gianinazzi, L., Gajda, J., Lehmann, T., Niewiadomski, H., Nyczyk, P., et al. Graph of thoughts: Solving elaborate problems with large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, 2024.

Bubeck, S., Chandrasekaran, V., Eldan, R., Gehrke, J., Horvitz, E., Kamar, E., Lee, P., Lee, Y. T., Li, Y., Lundberg, S., et al. Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv preprint arXiv:2303.12712*, 2023.

Burtsev, M. S., Kuratov, Y., Peganov, A., and Sapunov, G. V. Memory transformer. *arXiv preprint arXiv:2006.11527*, 2020.

Chang, K., Krishnamurthy, A., Agarwal, A., III, H. D., and Langford, J. Learning to search better than your teacher. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, 2015.

Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., Hesse, C., and Schulman, J. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Daumé III, H., Langford, J., and Marcu, D. Search-based structured prediction. *Mach. Learn.*, 75(3):297–325, 2009.

Dawid, A. and LeCun, Y. Introduction to latent variable energy-based models: A path towards autonomous machine intelligence. *arXiv preprint arXiv:2306.02572*, 2023.

Du, L., Mei, H., and Eisner, J. Autoregressive modeling with lookahead attention. *arXiv preprint arXiv:2305.12272*, 2023a.

Du, L., Torroba Hennigen, L., Pimentel, T., Meister, C., Eisner, J., and Cotterell, R. A measure-theoretic characterization of tight language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Toronto, Canada, 2023b. Association for Computational Linguistics.

Dziri, N., Lu, X., Sclar, M., Li, X. L., Jiang, L., Lin, B. Y., Welleck, S., West, P., Bhagavatula, C., Le Bras, R., et al. Faith and fate: Limits of transformers on compositionality. *Advances in Neural Information Processing Systems*, 36, 2024.

Feng, G., Zhang, B., Gu, Y., Ye, H., He, D., and Wang, L. Towards revealing the mystery behind chain of thought: a theoretical perspective. *Advances in Neural Information Processing Systems*, 36, 2023.

Glasmachers, T. Limits of end-to-end learning. In Zhang, M. and Noh, Y. (eds.), *Proceedings of The 9th Asian Conference on Machine Learning, ACML 2017*, volume 77 of *Proceedings of Machine Learning Research*, pp. 17–32. PMLR, 2017.

Gokaslan, A. and Cohen, V. Openwebtext corpus. `http://Skylion007.github.io/OpenWebTextCorpus`, 2019.

Gong, S., Li, M., Feng, J., Wu, Z., and Kong, L. Diffuseq: Sequence to sequence text generation with diffusion models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023.

Goyal, A., Lamb, A., Zhang, Y., Zhang, S., Courville, A. C., and Bengio, Y. Professor forcing: A new algorithm for training recurrent networks. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016*, pp. 4601–4609, 2016.

Goyal, S., Ji, Z., Rawat, A. S., Menon, A. K., Kumar, S., and Nagarajan, V. Think before you speak: Training language models with pause tokens. *The Twelfth International Conference on Learning Representations, ICLR 2024*, 20234.

Gu, A. and Dao, T. Mamba: Linear-time sequence modeling with selective state spaces, 2023.

Gu, J., Bradbury, J., Xiong, C., Li, V. O. K., and Socher, R. Non-autoregressive neural machine translation. In *6th International Conference on Learning Representations, ICLR 2018, Conference Track Proceedings*. OpenReview.net, 2018.

Gülçehre, Ç. and Bengio, Y. Knowledge matters: Importance of prior information for optimization. *J. Mach. Learn. Res.*, 17:8:1–8:32, 2016.

Gurnee, W., Nanda, N., Pauly, M., Harvey, K., Troitskii, D., and Bertsimas, D. Finding neurons in a haystack: Case studies with sparse probing. *arXiv preprint arXiv:2305.01610*, 2023.

Havrilla, A., Du, Y., Raparthy, S. C., Nalmpantis, C., Dwivedi-Yu, J., Zhuravinskyi, M., Hambro, E., Sukhbaatar, S., and Raileanu, R. Teaching large language models to reason with reinforcement learning, 2024.

Hsieh, C.-Y., Li, C.-L., Yeh, C.-K., Nakhost, H., Fujii, Y., Ratner, A., Krishna, R., Lee, C.-Y., and Pfister, T. Distilling step-by-step! outperforming larger language models with less training data and smaller model sizes. *arXiv preprint arXiv:2305.02301*, 2023.

Huang, W., Xia, F., Xiao, T., Chan, H., Liang, J., Florence, P., Zeng, A., Tompson, J., Mordatch, I., Chebotar, Y., Sermanet, P., Jackson, T., Brown, N., Luu, L., Levine, S., Hausman, K., and Ichter, B. Inner monologue: Embodied reasoning through planning with language models. In *Conference on Robot Learning, CoRL 2022, 14-18 December 2022, Auckland, New Zealand*, volume 205 of *Proceedings of Machine Learning Research*, pp. 1769–1782. PMLR, 2022.

Jiang, A. Q., Sablayrolles, A., Mensch, A., Bamford, C., Chaplot, D. S., de las Casas, D., Bressand, F., Lengyel, G., Lample, G., Saulnier, L., Lavaud, L. R., Lachaux, M.-A., Stock, P., Scao, T. L., Lavril, T., Wang, T., Lacroix, T., and Sayed, W. E. Mistral 7b, 2023.

Kääriäinen, M. Lower bounds for reductions. In *Atomic Learning Workshop*, 2006.

Kahneman, D. *Thinking, fast and slow*. Farrar, Straus and Giroux, 2011.

Kojima, T., Gu, S. S., Reid, M., Matsuo, Y., and Iwasawa, Y. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35, 2022.

Lai, Y., Zhang, C., Feng, Y., Huang, Q., and Zhao, D. Why machine reading comprehension models learn shortcuts?

In *Findings of the Association for Computational Linguistics: ACL/IJCNLP 2021, Online Event, August 1-6, 2021*, volume ACL/IJCNLP 2021 of *Findings of ACL*, pp. 989–1002. Association for Computational Linguistics, 2021.

LeCun, Y. Do large language models need sensory grounding for meaning and understanding? University Lecture, 2024.

Lee, N., Sreenivasan, K., Lee, J. D., Lee, K., and Papailiopoulos, D. Teaching arithmetic to small transformers. *arXiv preprint arXiv:2307.03381*, 2023.

Li, X., Trabucco, B., Park, D. H., Luo, M., Shen, S., Darrell, T., and Gao, Y. Discovering non-monotonic autoregressive orderings with variational inference. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL `https://openreview.net/forum?id=jP1vTH3inC`.

Li, Y., Huang, Y., Ildiz, M. E., Rawat, A. S., and Oymak, S. Mechanics of next token prediction with self-attention. In *27th International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2024.

Lin, C., Jaech, A., Li, X., Gormley, M. R., and Eisner, J. 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, NAACL-HLT 2021, Online, June 6-11, 2021*, pp. 5147–5173. Association for Computational Linguistics, 2021.

Ling, W., Yogatama, D., Dyer, C., and Blunsom, P. Program induction by rationale generation: Learning to solve and explain algebraic word problems. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pp. 158–167. Association for Computational Linguistics, 2017.

Liu, B., Ash, J. T., Goel, S., Krishnamurthy, A., and Zhang, C. Transformers learn shortcuts to automata. In *The Eleventh International Conference on Learning Representations, ICLR 2023*, 2023.

Liu, F. et al. Learning to summarize from human feedback. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020.

Loshchilov, I. and Hutter, F. Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

Lv, A., Zhang, K., Xie, S., Tu, Q., Chen, Y., Wen, J.-R., and Yan, R. Are we falling in a middle-intelligence trap? an analysis and mitigation of the reversal curse. *arXiv preprint arXiv:2311.07468*, 2023.

Madaan, A., Tandon, N., Gupta, P., Hallinan, S., Gao, L., Wiegreffe, S., Alon, U., Dziri, N., Prabhumoye, S., Yang, Y., et al. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, 36, 2024.

Malach, E. Auto-regressive next-token predictors are universal learners. *arXiv preprint arXiv:2309.06979*, 2023.

McCoy, R. T., Yao, S., Friedman, D., Hardy, M., and Griffiths, T. L. Embers of autoregression: Understanding large language models through the problem they are trained to solve. *arXiv preprint arXiv:2309.13638*, 2023.

Meng, K., Bau, D., Andonian, A., and Belinkov, Y. Locating and editing factual associations in GPT. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022*, 2022.

Merrill, W. and Sabharwal, A. The parallelism tradeoff: Limitations of log-precision transformers, 2023.

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

Momennejad, I., Hasanbeig, H., Frujeri, F. V., Sharma, H., Ness, R. O., Jojic, N., Palangi, H., and Larson, J. Evaluating cognitive maps and planning in large language models with cogeval. *Advances in Neural Information Processing Systems*, 36, 2023.

Monea, G., Joulin, A., and Grave, E. Pass: Parallel speculative sampling. *3rd Workshop on Efficient Natural Language and Speech Processing (NeurIPS 2023)*, 2023.

Nye, M. I., Andreassen, A. J., Gur-Ari, G., Michalewski, H., Austin, J., Bieber, D., Dohan, D., Lewkowycz, A., Bosma, M., Luan, D., Sutton, C., and Odena, A. Show your work: Scratchpads for intermediate computation with language models. *arXiv preprint arXiv:2112.00114*, 2021.

Olsson, C., Elhage, N., Nanda, N., Joseph, N., DasSarma, N., Henighan, T., Mann, B., Askell, A., Bai, Y., Chen, A., et al. In-context learning and induction heads. *arXiv preprint arXiv:2209.11895*, 2022.

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C. L., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., Schulman, J., Hilton, J., Kelton, F., Miller, L., Simens, M., Askell, A., Welinder, P., Christiano, P. F., Leike, J., and Lowe, R. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022*, 2022.

Pal, K., Sun, J., Yuan, A., Wallace, B. C., and Bau, D. Future lens: Anticipating subsequent tokens from a single hidden state. In *Proceedings of the 27th Conference on Computational Natural Language Learning, CoNLL 2023*. Association for Computational Linguistics, 2023.

Papadopoulos, V., Wenger, J., and Hongler, C. Arrows of time for large language models, 2024.

Paulus, R., Xiong, C., and Socher, R. A deep reinforced model for abstractive summarization. In *6th International Conference on Learning Representations, ICLR 2018, Conference Track Proceedings*. OpenReview.net, 2018.

Pezeshki, M., Kaba, S., Bengio, Y., Courville, A. C., Precup, D., and Lajoie, G. Gradient starvation: A learning proclivity in neural networks. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pp. 1256–1272, 2021.

Pfau, J., Infanger, A., Sheshadri, A., Panda, A., Michael, J., and Huebner, C. Eliciting language model behaviors using reverse language models. In *Socially Responsible Language Modelling Research*, 2023.

Pfungst, O. and Rahn, C. L. *Clever Hans (the horse of Mr. Von Osten) a contribution to experimental animal and human psychology*. New York, H. Holt and company, 1911.

Piekos, P., Malinowski, M., and Michalewski, H. Measuring and improving bert's mathematical abilities by predicting the order of reasoning. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 2: Short Papers), Virtual Event, August 1-6, 2021*, pp. 383–394. Association for Computational Linguistics, 2021.

Power, A., Burda, Y., Edwards, H., Babuschkin, I., and Misra, V. Grokking: Generalization beyond overfitting on small algorithmic datasets, 2022.

Qi, W., Yan, Y., Gong, Y., Liu, D., Duan, N., Chen, J., Zhang, R., and Zhou, M. Prophetnet: Predicting future n-gram for sequence-to-sequence pre-training. In *Findings of the Association for Computational Linguistics: EMNLP 2020, Online Event, 16-20 November 2020*, volume EMNLP 2020 of *Findings of ACL*, pp. 2401–2410, 2020.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. Language models are unsupervised multitask learners. 2019.

Ranaldi, L. and Zanzotto, F. M. Hans, are you clever? clever hans effect analysis of neural systems, 2023.

Ranzato, M., Chopra, S., Auli, M., and Zaremba, W. Sequence level training with recurrent neural networks. In *4th International Conference on Learning Representations, ICLR 2016, Conference Track Proceedings*, 2016.

Recchia, G. Teaching autoregressive language models complex tasks by demonstration. *arXiv preprint arXiv:2109.02102*, 2021.

Reynolds, L. and McDonell, K. Prompt programming for large language models: Beyond the few-shot paradigm. In *Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems*, 2021.

Ross, S. and Bagnell, D. Efficient reductions for imitation learning. In Teh, Y. W. and Titterington, D. M. (eds.), *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2010, Chia Laguna Resort, Sardinia, Italy, May 13-15, 2010*, volume 9 of *JMLR Proceedings*, 2010.

Ross, S. and Bagnell, J. A. Reinforcement and imitation learning via interactive no-regret learning. abs/1406.5979, 2014.

Ross, S., Gordon, G. J., and Bagnell, D. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2011*, JMLR Proceedings, 2011.

Sanford, C., Hsu, D., and Telgarsky, M. Transformers, parallel computation, and logarithmic depth. *arXiv preprint arXiv:2402.09268*, 2024.

Shah, H., Tamuly, K., Raghunathan, A., Jain, P., and Netrapalli, P. The pitfalls of simplicity bias in neural networks. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

Shalev-Shwartz, S. and Shashua, A. On the sample complexity of end-to-end training vs. semantic abstraction training. *arXiv preprint arXiv:1604.06915*, 2016.

Shalev-Shwartz, S., Shamir, O., and Shammah, S. Failures of gradient-based deep learning. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pp. 3067–3075. PMLR, 2017.

Shannon, C. E. A mathematical theory of communication. *The Bell System Technical Journal*, 27(3):379–423, 1948.

Shannon, C. E. Prediction and entropy of printed english. *The Bell System Technical Journal*, 30(1):50–64, 1951.

Shen, R., Bubeck, S., Eldan, R., Lee, Y. T., Li, Y., and Zhang, Y. Positional description matters for transformers arithmetic. *arXiv preprint arXiv:2311.14737*, 2023.

Shinn, N., Cassano, F., Berman, E., Gopinath, A., Narasimhan, K., and Yao, S. Reflexion: Language agents with verbal reinforcement learning, 2023.

Shlegeris, B., Roger, F., Chan, L., and McLean, E. Language models are better than humans at next-token prediction. *arXiv preprint arXiv:2212.11281*, 2022.

Shridhar, K., Stolfo, A., and Sachan, M. Distilling reasoning capabilities into smaller language models. *arXiv preprint arXiv:2212.00193*, 2022.

Springer, J. M., Kotha, S., Fried, D., Neubig, G., and Raghunathan, A. Repetition improves language model embeddings, 2024.

Stern, M., Shazeer, N., and Uszkoreit, J. Blockwise parallel decoding for deep autoregressive models. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, 2018.

Thrampoulidis, C. Implicit bias of next-token prediction, 2024.

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

Tschannen, M., Kumar, M., Steiner, A., Zhai, X., Houlsby, N., and Beyer, L. Image captioners are scalable vision learners too. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023.

Valmeekam, K., Marquez, M., and Kambhampati, S. Can large language models really improve by self-critiquing their own plans? *arXiv preprint arXiv:2310.08118*, 2023a.

Valmeekam, K., Marquez, M., Olmo, A., Sreedharan, S., and Kambhampati, S. Planbench: An extensible benchmark for evaluating large language models on planning and reasoning about change, 2023b.

Valmeekam, K., Marquez, M., Sreedharan, S., and Kambhampati, S. On the planning abilities of large language models - A critical investigation. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023c.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 5998–6008, 2017.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E. H., Le, Q. V., and Zhou, D. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022*, 2022.

Welleck, S., Kulikov, I., Kim, J., Pang, R. Y., and Cho, K. Consistency of a recurrent language model with respect to incomplete decoding. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, November 2020.

Wies, N., Levine, Y., and Shashua, A. Sub-task decomposition enables learning in sequence to sequence tasks. In *The Eleventh International Conference on Learning Representations, ICLR 2023*, 2023.

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

Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.

Xue, F., Likhosherstov, V., Arnab, A., Houlsby, N., Dehghani, M., and You, Y. Adaptive computation with elastic input sequence. In *International Conference on Machine Learning, ICML 2023*, Proceedings of Machine Learning Research. PMLR, 2023.

Yao, S., Yu, D., Zhao, J., Shafran, I., Griffiths, T., Cao, Y., and Narasimhan, K. Tree of thoughts: Deliberate problem solving with large language models. *Advances in Neural Information Processing Systems*, 36, 2023a.

Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K. R., and Cao, Y. React: Synergizing reasoning and acting in language models. In *The Eleventh International*

Conference on Learning Representations, ICLR 2023, 2023b.

Young, T. and You, Y. On the inconsistencies of conditionals learned by masked language models. *arXiv preprint arXiv:2301.00068*, 2022.

Zelikman, E., Wu, Y., Mu, J., and Goodman, N. D. Star: Bootstrapping reasoning with reasoning. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022.

Zhang, H., Li, L. H., Meng, T., Chang, K., and den Broeck, G. V. On the paradox of learning to reason from data. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI 2023, 19th-25th August 2023, Macao, SAR, China*, pp. 3365–3373. ijcai.org, 2023.

Zhao, T. Z., Kumar, V., Levine, S., and Finn, C. Learning fine-grained bimanual manipulation with low-cost hardware. In *Robotics: Science and Systems XIX, Daegu, Republic of Korea, July 10-14, 2023*, 2023.

Ziegler, D. M., Stiennon, N., Wu, J., Brown, T. B., Radford, A., Amodei, D., Christiano, P., and Irving, G. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*, 2019.

A. Limitations

1. Our arguments are empirical and conceptual. We have not provided a formal proof for our arguments.

2. We have also not demonstrated failure for very large models such as `Llama2` (Touvron et al., 2023) or `Mistral` (Jiang et al., 2023).

3. We note that there may be specific workarounds to make the Transformer (efficiently) learn the path-star task. For example, it may become solvable with other pre-trained models which may have been taught path-finding with step-by-step supervision (see Remark 1). The task may also be solvable via other workarounds such as in-context learning or multi-modal learning (where the model visually processes the image of the graph). Furthermore, since the path-star problem composes the same subroutine over itself, it allows efficient parallelizable solutions Sanford et al. (2024) that may not require learning a sequential composition of discrete subroutines. This can break the assumption of Proposition 3, making the optimization tractable. We warn the reader that these are however only specialized workarounds for this specific task; our broader point is that (a) next-token-prediction/teacher-forcing may still be relatively inefficient compared to teacherless training and (b) there may still be other novel tasks not seen during pre-training, or not solvable visually, or where no paralleizable solutions exist, for which similar failures may occur. We discuss this in detail in Remark 1.

4. Nevertheless, beyond the minimal path-finding setting, we have not demonstrated or characterized the range of problems where teacher-forcing-induced failure may occur. We only intuitively believe it should extend to other problem-solving tasks and creative-writing tasks that require lookahead (see § C).

5. It is also unclear if this failure generalizes to run-of-the-mill text-generation tasks.

B. Teacher-Forcing Failure and Snowballing Failure are Distinct

We emphasize that, while both the Clever Hans failure mode and the Snowball mode are both indicative of the inability to plan, these failure modes are also orthogonal to each other, and demand different solutions. We make this a bit more formal:

Proposition 4. *In the path finding problem of §4.1, there exists a next-token predictor that experiences Failures 2a, 2b due to teacher-forcing, but not the snowballing error Failure 1 due to autoregressive inference. Conversely, there exists a next-token predictor that experiences the latter failure but not the former.*

Proof. Consider the model learned via teacher-forcing on the graph problem. During inference, we saw that it suffers a debilitating error right in the first step (with accuracy of $1/d$ for degree d of the start node). Thus, during inference the error that is experienced is not from an accumulation over length. In fact, if only the first node is set correctly during inference, a model with the perfect Clever Hans cheat, would achieve 100% accuracy rate. Such a model does not experience snowballing errors.

On the other hand, consider a model, that in each step predicts the correct next vertex with a high accuracy of $1 - \epsilon$ for small ϵ. Such a model clearly has learned the correct plan, albeit with minor errors in each token. These errors however can snowball during inference. Thus, this model has no failure due to teacher-forcing, but will fail during autoregressive inference, if the path length is long. □

Differing solutions. Based on the above simple illustration, we note that the two failures need different solution approaches. Specifically, while snowballing errors may be fixable via "backtracking-and-planning" wrappers, teacher-forcing failures is a pathology that cannot be solved post-hoc.

C. An Illustration via Story-Telling

Can a teacher-forced model merely trained on thousands of stories learn to write plot twists? Indeed, Bubeck et al. (2023) report instances where models can fail to accomplish tasks involving creative-writing (e.g., poems). We speculatively extend our discussion in §4 to reason about this scenario. Consider for example, teacher-forcing on the following story that follows an often-used plot outline:

- `Event 1 (Setup):` Alex and Bob, who are friends, are trying to defeat the Evil King.

- `Event 2 (Conflict):` One day, surprisingly, Bob turns against Alex, and tries to thwart Alex's plans, *albeit unsuccessfully*.
- `Event 3:` Alex thinks Bob is evil too, defeats Bob first.
- `Event 4 (Backstory):` Losing the battle, Bob reveals he is a double-agent. In his final words, Bob explains he was ordered to defeat Alex.
- `Event 5 (Resolution):` To preserve the King's trust, Bob obeyed the command, but also *deliberately* failed at it. Bob then relays critical information he extracted from the King's inner circles.
- `Event 6:` Alex uses Bob's insider information to defeat the King.

Evidently, this story requires a plan: `Event 5` is a key plot resolution that the narrator must have planned before methodically generating parts of the setup in `Event 1` (introducing Bob as a friend) and the conflict in `Event 2` (Bob's turning against Alex, and failing at it). While training, the model must thus treat the story as a whole, and tease apart these dependencies between the events, some of which may be anti-chronological (akin to how, in the path-star graph, the model must learn that the problem is straightforwardly solvable when viewed from right-to-left).

However, we hypothesize that a teacher-forced model would take a rigid chronological (left-to-right) view. First, it would use the Clever Hans cheat to easily fit the plot resolution in `Event 5`: the model would use the facts of `Event 4` and `Event 2` (revealed as input) to fit the content of Bob's final words. Thus, the content of `Event 5` would no longer be available as supervision to guide how the model fits `Event 1` and `Event 2`. When the model tries to fit these earlier events, these events would become Indecipherable Tokens — the model would simply learn to fit them as arbitrary events. Thus, we conjecture that a model trained via teacher-forcing merely on raw, unannotated texts of stories — however many stories they may be — would not learn to plan its stories, and would instead create arbitrary twists and turns during inference, and improvize upon that.

D. Other Remarks

Remark 1. ***(Conditions under which first token becomes decipherable end-to-end)*** *There are certain corner cases where teacher-forcing can learn the (otherwise indecipherable) first token efficiently, without having to brute-force search an exponential space of algorithms. We enumerate these below. (Note though that even if the problem does become tractable, it can still be the case in these problems, next-token prediction is less efficient than predicting multiple future tokens.)*

1. ***Lucky prior biases:*** *If the model happens to have been exposed to certain relevant kinds of supervision during pre-training, then the model will be biased towards a favorable part of the search space, and chance upon the right algorithm much quicker.*

   (a) *If the model had witnessed the same task but with the correct step-by-step supervision, then the prior would assign high probability to the correct algorithm. (One can imagine that the the true end-to-end algorithm itself becomes a readily available subroutine in this case.)*

   (b) *Or, in the specific path-star example, if the prior bias assigns high probability to all l subroutines being identical, then the search only needs $O(|\mathcal{C}|)$ time (where $\mathcal{C}$ is the set of candidate subroutines.) For illustration, see the Transformer construction for k-hop problems in (Sanford et al., 2024).*

   Note that in such a case, one may still be able to demonstrate intractability by constructing slight variations of the tasks that defy such prior biases (e.g., tasks where the subroutines are not identical).

2. ***Small graph size.*** *If the number of edges is very small (say $|E|$) in proportion to the training data, then the model can learn alternative solutions that "memorize" the problem:*

   (a) ***Naive memorization:*** *If the vocabulary has only $|\mathcal{V}|$ possible node ID's, then there are only $O(|\mathcal{V}|^{|E|})$ possible adjacency lists the model can see. So, if the training data has at least $\Omega(|\mathcal{V}|^{|E|})$ datapoints, then, every test example is seen with high probability during training. Here, the model merely needs to look up exact replicas from training, and regurgitate the subsequent values from the training string.*

   (b) ***Node-ID-agnostic memorization:*** *If the number of training data is $\Omega(|E|!)$, the model can still implement a form of memorization, but this requires a cleverer strategy that is agnostic to the node ID's. Concretely, for a fixed assignment of node ID's, there are only $O(|E|!)$ ways to permute the adjacency list. Assume that the model sees* *all such permutations during training (albeit with varying instantiations of the node IDs). During inference, the model can first look up whether the test input corresponds to an existing permutation it had witnessed (possibly with different node ID's). For example* $1 \to 2; 2 \to 3; 4 \to 1$ *would correspond to* $10 \to 20; 20 \to 30; 40 \to 10$. *Then the model merely needs to recall from its memory, the index where the target token is located in the input for that permutation. The model can then look at the same index in the test input and output the node ID located there. Thus, if the target token was* 4 *for the first sequence above, the model can output* 40 *for the other sequence.*

Note that this doesn't contradict Proposition 3 because the proposition only precludes learning the "true" path-finding algorithm, not spurious memorizing solutions.

3. ***Small path length.*** *If the path length* l *is small, then either the search space of algorithms (which is about as large as* $|\mathcal{C}|^l$*) becomes tractable.*

   (a) *For example, if* $l = 3$*, the first node is the only intermediate node between the start and goal token. Here, the model can easily learn the "right-to-left" solution that the desired node is the only node preceding the goal node.*

Remark 2. ***(Mechanism implemented by teacherless model)*** *The (hypothetical) solution that the teacherless model must implement is a fairly difficult one to implement — yet the model surprisingly learns to implement it. Recall that our hypothesis is that the teacherless model automatically learns to fit the targets in the reverse order, since the path from* $v_{\mathtt{goal}}$ *is unique. This is indeed what we find in Fig 4, where the accuracies of the later tokens become higher earlier. Note though that this is a fairly difficult computation to implement. First, when the model predicts* v_i*, it must require the identity of* v_{i+1}*. However, this identity is not fed as input to the model, in the absence of the teacher. Thus the model must have computed* v_{i+1} *and crucially, stored that in one of its its internal representations. Then, by induction, when predicting the first node* v_1*, the model must know the identity of all the other nodes in the path. In other words, the model must have (a) computed and (b) stored the whole solution in its hidden representations before it outputs the first token. This is a substantial type of lookahead that some of our models are able to achieve under teacherless training.*

E. Experiment Notations

E.1. Verifying In-Distribution Performance

We simply compute exact match with the ground truth path as follows:

$$\mathtt{Acc}_{\mathtt{ag}}(\mathtt{LM}_\theta) := \mathbb{P}(\hat{\boldsymbol{r}} = \boldsymbol{r}), \qquad \boldsymbol{p}, \boldsymbol{r} \sim \mathcal{D}, \;\; \hat{\boldsymbol{r}} \overset{\mathtt{ag}}{\sim} \mathtt{LM}_\theta. \tag{5}$$

E.2. Quantifying the Clever Hans Cheat

Formally, let $\mathtt{Unif}(\mathcal{N}(v_{\mathtt{start}}))$ denote a uniform distribution over the set of adjacent nodes of $v_{\mathtt{start}}$. For any node v in the graph, denote by $\mathtt{path}(v)$ the path emanating from v and going outwards, away from the start node. Notice that except for $v = v_{\mathtt{start}}$, this path is unique. We thus measure

$$\mathtt{Acc}_{\mathtt{cheat}}(\mathtt{LM}_\theta) := \mathbb{P}\left(\hat{\boldsymbol{r}}_{1<} = \mathtt{path}(v_1')\right) \tag{6}$$

$$\begin{aligned} \text{where} \quad & \boldsymbol{p}, \boldsymbol{r} \sim \mathcal{D}, \;\; \hat{\boldsymbol{r}}_{1<} \overset{\mathtt{ag}}{\sim} \mathtt{LM}_\theta(\cdot; \boldsymbol{p}, v_{\mathtt{start}}, v_1') \\ & v_1' \sim \mathtt{Unif}(\mathcal{N}(v_{\mathtt{start}})). \end{aligned}$$

E.3. Quantifying the Indecipherable Token Failure

To quantify the Indecipherable Token failure, in our path-finding task, we measure the accuracy in predicting the first token after the start node.

$$\mathtt{Acc}_{\mathtt{1st}}(\mathtt{LM}_\theta) = \mathbb{P}\left(\hat{r}_1 = r_1\right), \qquad \boldsymbol{p}, \boldsymbol{r} \sim \mathcal{D}, \hat{\boldsymbol{r}} \overset{\mathtt{ag}}{\sim} \mathtt{LM}_\theta(\cdot; \boldsymbol{p}). \tag{7}$$

E.4. Inference in Teacherless Training

In teacherless training, we make use of an uninformative input $\boldsymbol{r}^{\$}$ that simply corresponds to a series of dummy tokens denotes by $\$$. During inference, instead of autoregressing on the model's own output, we use this uninformative input. We formalize this below:

$$\hat{\boldsymbol{r}} \overset{\$}{\sim} \texttt{LM}^{\$}_{\theta}(\cdot;\boldsymbol{p}) \text{ where } \hat{r}_i \sim \texttt{LM}^{\$}_{\theta}(\cdot;\boldsymbol{p},\boldsymbol{r}^{\$}_{<i}). \tag{8}$$

We then denote the accuracy of the model as follows:

$$\texttt{Acc}_{\$}(\texttt{LM}^{\$}_{\theta}) = \mathbb{P}\left(\hat{\boldsymbol{r}} = \boldsymbol{r}\right) \qquad\qquad \boldsymbol{p},\boldsymbol{r} \sim \mathcal{D},\;\; \hat{\boldsymbol{r}} \overset{\$}{\sim} \texttt{LM}^{\$}_{\theta}(\cdot;\boldsymbol{p}). \tag{9}$$

E.5. Reversed Training

Notationally, in reversed training we let $\texttt{LM}^{\texttt{rev}}_{\theta}$ be the model trained to maximize $\mathcal{J}_{\texttt{next-token}}$ with the targets (and the teacher-forced inputs) set to $\boldsymbol{r}^{\texttt{rev}} = r_{L_{\texttt{resp}}}, \ldots r_1$, the reversal of $\boldsymbol{r}$. We then measure the autoregressive accuracy by comparing against $\boldsymbol{r}^{\texttt{rev}}$:

$$\texttt{Acc}_{\texttt{rev}}(\texttt{LM}^{\texttt{rev}}_{\theta}) = \mathbb{P}\left(\hat{\boldsymbol{r}} = \boldsymbol{r}^{\texttt{rev}}\right), \qquad\qquad \boldsymbol{p},\boldsymbol{r} \sim \mathcal{D}, \hat{\boldsymbol{r}} \overset{\text{ag}}{\sim} \texttt{LM}^{\texttt{rev}}_{\theta}(\cdot;\boldsymbol{p}) \tag{10}$$

F. More Experimental Results

F.1. Snowball Failure

To explicitly measure to what degree the model falls victim to the snowball effect, we train *GPT-Mini* on graphs of various path lengths l. In order to remove the failure stemming from the difficult first token, we teacher-force the model for the first token and then check how accurate the generations are for subsequent tokens. More concretely, we evaluate

$$\texttt{Acc}_{\texttt{sb}}(\texttt{LM}_{\theta}) = \mathbb{P}\left(\hat{\boldsymbol{r}}_{1<} = \boldsymbol{r}_{1<}\right) \tag{11}$$

$$\text{where} \quad \boldsymbol{p},\boldsymbol{r} \sim \mathcal{D},\;\; \hat{\boldsymbol{r}}_{1<} \overset{\text{ag}}{\sim} \texttt{LM}_{\theta}(\cdot;\boldsymbol{p},r_1)$$

If $\texttt{Acc}_{\texttt{sb}}(\texttt{LM}_{\theta})$ is ≈ 1, then *Failure 1* is not prominent in our task. If $\texttt{Acc}_{\texttt{sb}}(\texttt{LM}_{\theta}) \ll 1$, then clearly teacher-forcing is responsible for surpressing errors in generation, strongly hinting at the fact that *Failure 1* is at play. We display the results in Fig. 5 (left). We observe that the accuracy $\texttt{Acc}_{\texttt{sb}}$ is barely affected even for graphs with very long paths $L = 40$.

As another metric, we proceed token by token during inference, and evaluate the probability of correctly predicting all tokens up to the current one. We report this for $G_{2,40}$ in Fig. 5 (right). Similarly, while the success probability does decay for larger length (at an exponential rate), it remains very high due to the failure events being so unlikely. We thus conclude that *Failure 1* is not as prominent in this setting.

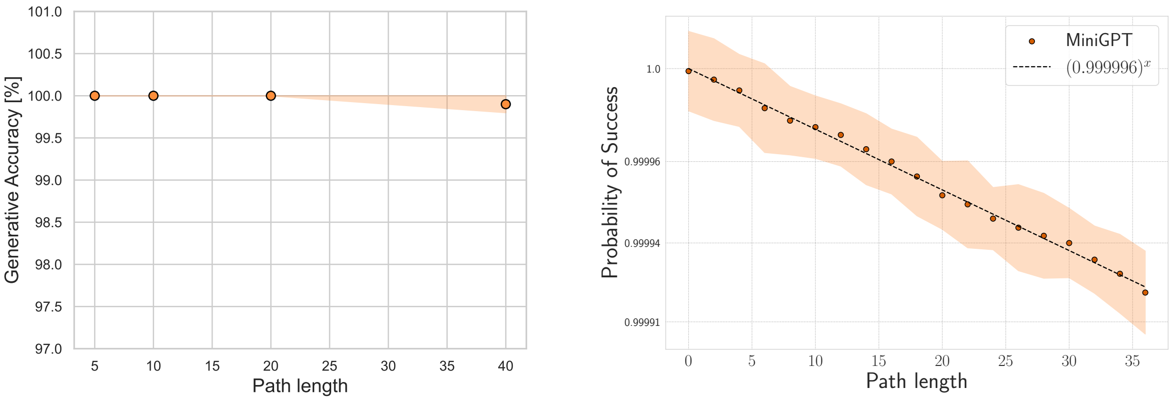

Figure 5. Accuracy of $\texttt{LM}_{\theta}$ when conditioned on the first difficult token (left) for graphs of various length. Probability of correct prediction of $\texttt{LM}_{\theta}$ as a function of current token position on $G_{2,40}$, as we walk towards the goal.

F.2. Clever Hans Cheating Accuracies

In Table 1 we display the Clever Hans cheating accuracies $\texttt{Acc}_{\texttt{cheat}}(\texttt{LM}_\theta)$. We observe that in almost all cases, all the models achieve nearly perfect cheating accuracies. The only exception is the high-degree graph $G_{20,5}$ where all models struggle to even fit the training data.

	$G_{2,5}$	$G_{2,20}$	$G_{5,5}$	$G_{10,5}$	$G_{20,5}$
GPT-MINI	99.7	100	100	99.8	0.0
GPT2-LARGE	99.8	99.7	100	99.8	0.0
MAMBA	97.6	98.3	99.5	95.9	0.0

Table 1. Evaluating Clever Hans cheating accuracies $\texttt{Acc}_{\texttt{cheat}}(\texttt{LM}_\theta)$ (in percent %) for different types of graphs.

F.3. More Detailed Accuracies

We report more detailed accuracy values per model in the following tables. We display standard accuracy $\texttt{Acc}_{\texttt{ag}}(\texttt{LM}_\theta)$ in Table. 2, teacherless accuracy $\texttt{Acc}_{\$}(\texttt{LM}_\theta)$ in Table. 3 and reverse accuracy $\texttt{Acc}_{\texttt{rev}}(\texttt{LM}_\theta)$ in Table. 4. In general we observe that solving the task with standard next-token prediction is very tough and performance is limited to $\frac{1}{d}$ where d is the degree of the graph $G_{d,l}$.

	$G_{2,5}$	$G_{2,20}$	$G_{5,5}$	$G_{10,5}$	$G_{20,5}$
GPT-MINI	49.8	49.1	19.1	8.1	0.0
GPT2-LARGE	48.9	49.2	19.4	10.3	3.5
MAMBA	48.5	48.7	20.2	9.3	0.0

Table 2. Autoregressive accuracies $\texttt{Acc}_{\texttt{ag}}(\texttt{LM}_\theta)$ (in percent %) for different types of graphs.

Teacherless training on the other hand works very well with GPT2-Large, allowing it to solve most graph tasks perfectly. From-scratch models however also struggle to learn the task in this fashion (except for GPT-Mini on the simplest graph, $G_{2,5}$).

	$G_{2,5}$	$G_{2,10}$	$G_{2,20}$	$G_{5,5}$	$G_{10,5}$	$G_{20,5}$
GPT-MINI	99.9	0.0	0.0	0.0	0.0	0.0
GPT2-L	99.9	98.8	0.0	99.0	97.8	0.0
MAMBA	0.0	0.0	0.0	0.0	0.0	0.0

Table 3. Autoregressive accuracy $\texttt{Acc}_{\$}$ when using a teacherless response.

Finally, reversing the sequence significantly simplifies the problem for all the models, allowing near perfect accuracies across all graphs.

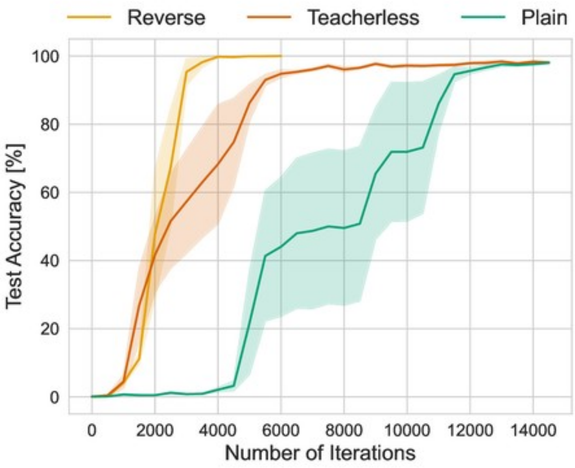

Figure 6. Test accuracies for 3-digit addition for standard, reversed and teacherless training.

	$G_{2,5}$	$G_{2,20}$	$G_{5,5}$	$G_{10,5}$	$G_{20,5}$
GPT-MINI	99.7	99.8	100	99.8	0.0
GPT2-LARGE	99.9	99.9	99.6	99.8	99.9
MAMBA	98.5	96.2	99.1	99.5	0.0

Table 4. Autoregressive accuracy $\texttt{Acc}_{\texttt{rev}}$ when reversing the response $\boldsymbol{r}$.

F.4. Arithmetic Tasks

To further highlight that the identified failure modes are relevant for realistic tasks, we study the task of addition. We consider 3-digit addition, where samples are of the form

$$x_1x_2x_3 + y_1y_2y_3 = z_1z_2z_3z_4$$

We use the natural encoding (i.e. use the digits themselves as encodings) and pad shorter numbers with leading zeros to ensure fixed lengths. It has been observed in prior work Lee et al. (2023); Shen et al. (2023) that reversing the result (i.e. $z_4z_3z_2z_1$) is very beneficial for training, leading to significantly more sample-efficient learning. Here we study if our teacherless training strategy can lead to similar gains over the standard encoding, i.e. we encode the inputs as

$$x_1x_2x_3 + y_1y_2y_3 = \$\$\$\$$$

while keeping all other aspects of the training pipeline (such as the targets and the objective) the same as in standard training. We again train a *GPT-Mini*-style transformer with *AdamW* using a learning rate of 5e-4. We display the resulting test accuracies for both standard, reversed and teacherless training in Fig. 6, as a function of training iterations. Here we follow previous works and at every iteration, sample with replacement from all the possible 3-digit configurations (aside from a test set put aside before). We can indeed see that teacherless training leads to more sample-efficient learning compared to the standard format, but to slightly less efficient learning compared to the reverse format. This again hints at the fact that some form of CleverHans cheating is picked up when learning addition.

In this case though, it is more difficult to precisely characterize the form of the cheat in contrast to the path-finding problem. One possible form of cheating may be that in addition, knowing the leading digits of the answer can provide information about the subsequent digits. For example, when we add two single-digit numbers (picked uniformly at random), knowing that the $10'th$ place is a 1 rather than a 0, tells us that the 0th place is more likely to be a smaller digit like 0.

F.5. Predicting Only the Indecipherable Token

To study the difficulty of the first token in isolation from the Clever Hans cheat, we reduce the task to solely predicting this token, removing the rest of the path. The problem thus effectively becomes a classification problem. We use the same learning setup as for previous experiments and keep the same number of examples. We display the results in Table 5. Our results indicate that the model fails to learn the first token in isolation.

This reinforces our understanding of the role played by the Clever Hans cheat in bringing about failure. One possibility could have been that the first node in isolation is easy to learn, but the Clever Hans cheat makes it harder somehow (e.g., it attracts the model to a bad local minimum). The other possibility (the one we put forth) is that the Clever Hans cheat reduces the problem to learning the first token in isolation, which we argued is hard. These experiments confirm the second possibility.

	$G_{2,5}$	$G_{2,20}$	$G_{5,5}$	$G_{10,5}$	$G_{20,5}$
GPT2-LARGE	50.2	50.4	18.9	10.4	4.5

Table 5. Accuracy when solely learning to predict the difficult token r_1

G. Other experimental details

G.1. Tokenization

We tokenize the graph in the following manner: (1) we first tokenize the randomly shuffled edge list as “$|v_1\ v_2|v_3\ v_4|$...” where the first vertex in each edge is the one closest to $v_{\texttt{start}}$, (2) then append start and goal node as “$/v_{\texttt{start}}\ v_{\texttt{goal}} =$ ” and (3) then append the full path repeating start and goal node, ”$v_{\texttt{start}}\ v_{i_1} \dots v_{i_{l-1}}\ v_{\texttt{goal}}$”. Note that (1) and (2) make up the prefix $\boldsymbol{p}$, which the model does not learn to predict. Then, (3) is the target sequence that the model aims to learn. The vocabulary size is thus given by $N + 3$, where we add entries for the special tokens “$|$”, “$/$” and “ $=$ ”. When using the pre-trained models GPT2 we use the tokenizer that was employed for pre-training, in this case the *Byte-Pair tokenizer* (Radford et al., 2019).

G.2. Models

When training Transformer models from scratch, we use a small model consisting of $n_{\text{layers}} = 12$ blocks with embedding dimension $e_{\text{dim}} = 384$, $n_{\text{heads}} = 6$ attention heads and MLP expansion factor $e = 4$, coined *GPT-Mini*. For pre-trained models, we consider GPT2-Large with $n_{\text{layers}} = 36$, $e_{\text{dim}} = 1280$, $n_{\text{heads}} = 20$ and expansion factor $e = 4$ (Radford et al., 2019). To further evaluate purely recurrent models, we perform experiments with the recent Mamba model (Gu & Dao, 2023). We train the Mamba models from scratch with 12 layers and embedding dimension 784. We train all the models with the *AdamW* optimizer (Loshchilov & Hutter, 2019). For models trained from scratch we use a learning rate of $\eta = 0.0005$ while for pre-trained models we use a smaller one of $\eta = 0.0001$. In both cases we use weight decay of strength 0.01. Models from scratch are trained for up to 500 epochs in order to ensure convergence. Pre-trained models require less training time and we usually fit the training data perfectly after 10 epochs.

H. More Related Work

Other arguments about next-token prediction. We note that the works of Kääriäinen (2006); Ross & Bagnell (2010) capture a stronger notion of snowballing, wherein, once an erroneuous sub-optimal action is committed, the model is more likely to commit more sub-optimal actions since it has wandered into territories that it was not trained on. Implicitly, the error here is not evaluated as an exact match of the response (i.e., $\boldsymbol{r} \neq \hat{\boldsymbol{r}}$) but as a cumulative notion of error over all steps (e.g., $\sum \mathbf{1}[r_i \neq \hat{r}_i]$). In this setting, there is an additional cause of failure called *exposure bias*: the teacher-forced model has only been trained on correct trajectories, and has not learned how to recover from poor trajectories. Nevertheless, even this notion of snowballing assumes that that teacher-forcing has learned an accurate next-token predictor in the first place, which our failure mode challenges.

A closely-related criticism (Bubeck et al., 2023; Dawid & LeCun, 2023; LeCun, 2024; Du et al., 2023a) is that to model human thinking, we need to model two types of thinking as outlined in Kahneman (2011): a fast (System 1) thinking process that is also guided by a slower (System 2) thinking process. Theoretically, Lin et al. (2021) show that there are formal languages for which expressing some next-tokens may require super-polynomial time or parameter count during *inference*. These arguments however only suggest that some tokens require more computation; not that they are specifically problematic under left-to-right learning. However, Du et al. (2023a) informally note that some next tokens can be hard to *learn* as they require a global understanding of what will be uttered in the future (but see Remark 3 below).

Remark 3. ***(Locally Unlearnable Token vs Indecipherable Token)*** *We note that the "locally unlearnable" hypothesis of Du et al. (2023a) is related to, but not the same as the Indecipherable Token failure. The hypothesis in Du et al. (2023a) is that when we learn tokens left-to-right, some tokens simply cannot be learned since crucial information becomes available only in subsequent tokens. This hypothesized failure may happen regardless of whether the supervision from subsequent tokens is lost to a Clever Hans cheat. In contrast, in our path-star graph, the Indecipherable Token becomes unlearnable only because of the Clever Hans cheat. For example, the (first) Indecipherable Token in the path-star problem is locally learnable by the teacherless model (where, the crucial information is still only presented after this token). This token becomes unlearnable only in the teacher-forced model where the Clever Hans cheat emerges.*

We survey related arguments of next-token prediction, orthogonal to our main discussion regarding planning. Allen-Zhu & Li (2023); Lv et al. (2023) report that language models that are trained on `A equals B` are unable to infer `B equals A`, which Allen-Zhu & Li (2023) suggest is due to autoregressive left-right training. Du et al. (2023b); Welleck et al. (2020) formalize the limitation that autoregressive models may potentially assign non-zero probability to infinite-length strings, thus leading to non-terminating inference. Li et al. (2024) provide a Transformer-specific analysis of how self-attention affects the optimization geometry of next-token prediction. Thrampoulidis (2024) provide an analysis of the implict bias of optimization with next-token prediction for linear models.

Other limitations of Transformers Merrill & Sabharwal (2023) identify limitations of the representative power of Transformer architecture when the arithmetic precision is logarithmic in the number of input tokens. Bender et al. (2021) criticize GPT-like language models as simply parroting out training data with minor stochasticity, while Arkoudas (2023) report that such models struggle with reasoning, even if not a stochastic parrot. Young & You (2022) study masked language (T5, BERT) models (not causally-trained) and argue there are inconsistencies in the probabilities that they assign. E.g., when conditioned on '`white`', the probability of '`rice`' may be higher '`bread`' but the probability of '`white bread`' and '`white rice`' are the opposite. Artetxe et al. (2022) empirically analyze the effect of bidirectional attention and bidirectional supervision (as in masked language modeling) during pretraining on the ability of the model to do various things, including next-token prediction. Springer et al. (2024) argue that autoregressive Transformers compute sub-optimal embeddings that can be improved by repeating the input text twice.

Finally, we note that (Ranaldi & Zanzotto, 2023) use the term Clever Hans effect to denote how models can pick up spurious correlations between the position of a choice in a multiple-choice question, and the correctness of the answer. We note that the above correlation is inherent to the distribution, and independent of teacher-forcing. We distinguish this from the Clever Hans *cheating* which happens under the guidance of teacher-forcing.

End-to-end reasoning and chain-of-thought supervision. In our path-star graph, learning the Indecipherable Token (the first node v_1) can be thought of as a task whose end target is v_1, but whose implicit intermediate targets (or "chain-of-thought") correspond to the unique path starting from $v_{\texttt{goal}}$ headed towards v_1 (although this is only provided as supervision after the first token). In this terminology, we can rephrase our claim as the model failing to learn the end target once the intermediate targets are lost to the Clever Hans Cheat.

Such limits of end-to-end learning have been echoed in literature on learning with chain-of-thought-type supervision. Recent theoretical works have shown broad classes of tasks (e.g., any function efficiently computed by a Turing machine) where prepending CoT to the end target allows efficiently learning tasks; yet, there are "multi-hop reasoning" tasks that are unlearnable end-to-end (i.e., without intermediate supervision) either due to computational hardness (Wies et al., 2023) or representational limits (Malach, 2023)). Earlier theoretical works Shalev-Shwartz et al. (2017); Shalev-Shwartz & Shashua (2016) have similarly proven negative results for end-to-end learning in similar settings. Similar empirical arguments have been made in neural network literature (Gülçehre & Bengio, 2016; Glasmachers, 2017) and also more recently, in language models on complex reasoning and math problems (Nye et al., 2021; Ling et al., 2017; Cobbe et al., 2021; Piekos et al., 2021; Zelikman et al., 2022; Recchia, 2021; Cobbe et al., 2021; Hsieh et al., 2023; Shridhar et al., 2022).

Remark 4. ***(Chain-of-thought before vs. after end target.)*** *It is worth noting though that the above lines of work are concerned with chain-of-thought that is present before the end target; in our setup, this supervision is presented only after* *the end target. Surprisingly, some of our teacherless models manage to utilize even such hindsight chain-of-thought. This success is not fully explained by existing positive results about chain-of-thought supervision, such as Wies et al. (2023); Malach (2023), where supervision is provided before the end target.*

Going beyond next-token prediction. Inference-time techniques like chain-of-thought (Reynolds & McDonell, 2021; Wei et al., 2022; Kojima et al., 2022) and its variants (Yao et al., 2023a; Besta et al., 2024; Yao et al., 2023b) or those that elicit feedback from the model (Madaan et al., 2024; Huang et al., 2022; Shinn et al., 2023) can be thought of as going beyond conventional form of inference by allowing the model to think more before producing its final answer. However, the backbone in these models are still trained by standard teacher-forcing. While other techniques (Burtsev et al., 2020; Xue et al., 2023; Goyal et al., 20234) train the model to explicitly think more, even these boil down to next-token prediction during training.

One may argue that reinforcement learning-based training (Ranzato et al., 2016; Wu et al., 2016; Bahdanau et al., 2017; Paulus et al., 2018; Ziegler et al., 2019; Liu et al., 2020; Ouyang et al., 2022) is another way to build backbones that go beyond teacher-forcing. However, it is worth noting that the gradients in these techniques boil down to teacher-forcing on the model's own generated answer. Furthermore, if we desire that the model be able to generate a solution that can plan ahead of time, it is unclear how a model can go from a complete inability to plan (that may assign near-zero probability to the true plan in an exponential space of solutions), to discovering the correct plan simply through preference-based feedback (see (Havrilla et al., 2024) for related empirical evidence).

Another line of work — spanning language (Bengio et al., 2015; Goyal et al., 2016), imitation learning (Ross et al., 2011; Ross & Bagnell, 2010; 2014) and structured prediction (Daumé III et al., 2009; Chang et al., 2015) — has been aimed at addressing the Snowball Failure, under the assumption that the model has otherwise learned an accurate next-step predictor. Broadly, the idea is to train the model on a mixture of the ground truth sequences and the model-generated sequences themselves, as a way to ensure that the test-time and training-time distributions are as similar as possible. These techniques however do not address the failure to learn a good next-step predictor in the first place.

As for reversal-based training, Lee et al. (2023); Shen et al. (2023) observe that addition tasks become much simpler when the digits are reversed. Their argument is that this explicitly assists the model to learn a simpler algorithm. When it comes to natural language however, Papadopoulos et al. (2024) find that reversing hurts the model's perplexity.

Predicting future tokens. Some works (Gurnee et al., 2023; Meng et al., 2022; Pal et al., 2023) aim to recover future tokens that an already-trained model may predict based on the internal layers of the current token. Note that the success of this does not imply that the model necessarily plans well. This only means that it is possible to recover what the already-trained model wants to generate in the future (which may simply be a bad plan). Pfau et al. (2023) train a language model to predict in reverse with the orthogonal goal of finding prefixes that elicit certain behaviors.

Shortcut-learning in language models. A line of work has empirically and theoretically analyzed how Transformer-based language models learn superficial shortcuts to (partially) solve tasks such as learning multiplication (Dziri et al., 2024), logic (Zhang et al., 2023), automata (Liu et al., 2023), recursion (Young & You, 2022), reading comprehension (Lai et al., 2021) and multiple-choice questions (Ranaldi & Zanzotto, 2023) However, these shortcuts must *not* be confused with the Clever Hans cheating induced by teacher-forcing as elaborated below.

Remark 5. ***(Difference between Clever Hans cheating and known shortcut-learning failures in Transformers.)*** *First, these aforementioned shortcuts exist independent of teacher-forcing: these are correlations between the prefix (such as the initial digits of two multiplicands) and the final answer (the initial digits of the product) in the underlying training distribution. But Clever Hans cheats arise only upon teacher-forcing: these are correlations between the prefixes of the answer itself to the rest of the answer. Second, the above shortcuts only fail out-of-distribution (such as when the number of multiplied digits is increased, where the failure is in length generalization (Anil et al., 2022)). In contrast, the Clever Hans cheat is more severe as it causes in-distribution failure. Thirdly, the aforementioned empirical observations are specific to Transformers, and the theoretical arguments rely crucially on properties of the Transformer (such as its non-recurrence and convolution, or its self-attention modules). Our argument however only relies on the teacher-forcing objective with no reliance on the Transformer architecture, and is demonstrated even for the recurrent Mamba architecture.*