# Reinforced Context Order Recovery for Adaptive Reasoning and Planning

**Long Ma**
Academy for Advanced Interdisciplinary Studies,
Peking University, Beijing, China
malong@pku.edu.cn

**Fangwei Zhong**✉
School of Artificial Intelligence,
Beijing Normal University, Beijing, China
fangweizhong@bnu.edu.cn

**Yizhou Wang**
School of Computer Science, Institute for Artificial Intelligence,
State Key Laboratory of General Artificial Intelligence, Peking University, Beijing, China
yizhou.wang@pku.edu.cn

## Abstract

Modern causal language models, followed by rapid developments in discrete diffusion models, can now produce a wide variety of interesting and useful content. However, these families of models are predominantly trained to output tokens with a fixed (left-to-right) or random order, which may deviate from the logical order in which tokens are generated originally. In this paper, we observe that current causal and diffusion models encounter difficulties in problems that require adaptive token generation orders to solve tractably, which we characterize with the $\mathcal{V}$-information framework. Motivated by this, we propose Reinforced Context Order Recovery (ReCOR), a reinforcement-learning-based framework to extract adaptive, data-dependent token generation orders from text data without annotations. Self-supervised by token prediction statistics, ReCOR estimates the hardness of predicting every unfilled token and adaptively selects the next token during both training and inference. Experiments on challenging reasoning and planning datasets demonstrate the superior performance of ReCOR compared with baselines, sometimes outperforming oracle models supervised with the ground-truth order.[1]

## 1 Introduction

Text generation models have seen remarkable advancements in the past few years, with causal language models (CLMs) trained by next-token prediction taking the lead [1, 2, 3], followed by more recent discrete diffusion models [4, 5, 6]. These classes of models have demonstrated impressive capabilities across a wide range of tasks, from writing code to executing tasks as agents [7, 8, 9].

Despite the developments, current models still encounter significant challenges when faced with complex reasoning and planning problems that require a long-horizon and flexible decision-making process. A crucial part of this challenge lies in the token generation order [10]. As illustrated in Fig. 1, in adaptive reasoning tasks like Sudoku, some cells could be hard to predict instantly, depending on other cells to be filled first and provide additional constraints to eliminate candidates. However, CLMs always follow a rigid left-to-right generation paradigm, encountering many intractable tokens along the way. In contrast, humans rarely solve complex reasoning problems in a strictly linear fashion; we tackle the easiest parts first and use those insights to address progressively more challenging ones. Taking note of this issue, existing works either explicitly train the model to predict the easiest next

[1] Project page: https://sites.google.com/view/recor-ai

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

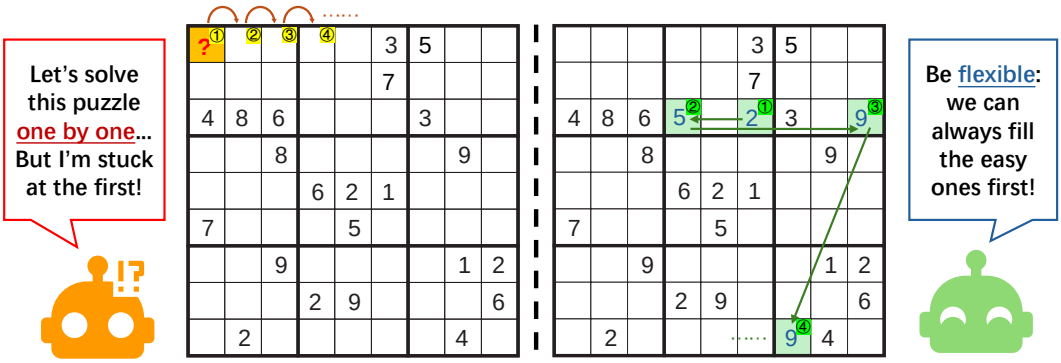



Figure 1: Illustration of ReCOR (right) compared with standard causal language modeling (left). While causal language modeling always tries to produce tokens left-to-right, ReCOR estimates the hardness of each token and adaptively prioritizes the easy ones without external supervision.

token [10] or leverage the properties of masked diffusion models (MDMs) [11, 12] to adaptively decode easy tokens during inference. However, these methods require additional annotations or introduce large distribution shifts between training and inference, hurting performance.

Facing these challenges, in this paper, we introduce Reinforced Context Order Recovery (ReCOR), a self-supervised framework that learns to adaptively determine the optimal token generation order without explicit order annotations. Our approach is motivated by the insight that the hardness of predicting different tokens varies dramatically conditioned on the current context, which we characterize using the framework of predictive $\mathcal{V}$-information. To operationalize the objective derived under this framework, ReCOR casts order prediction as a decision-making problem and trains a policy that adaptively selects the next token (Fig. 1). ReCOR jointly optimizes the token prediction model and the order prediction policy, generating rewards with the former as self-supervision for the latter. Furthermore, unlike previous approaches that apply adaptive strategies for inference only, ReCOR follows the same distribution of order during both training and inference. This ensures that the model not only adapts its generation behavior during inference but also benefits from learning informative, tractable token prediction tasks during training. In the experiments, we demonstrate the effectiveness of ReCOR on a variety of reasoning and planning tasks, including arithmetic problems and logic puzzles, where ReCOR consistently outperforms the state-of-the-art methods.

The contributions of our paper are fourfold: 1) We observe and characterize the token ordering problem using the framework of predictive $\mathcal{V}$-information and propose a corresponding objective; 2) We propose an RL-based formulation and algorithm for optimizing the objective and obtaining an adaptive order predictor; 3) We empirically validate ReCOR on multiple reasoning and planning benchmarks, demonstrating state-of-the-art performance that is competitive or even better than oracle models with access to ground-truth orders; 4) We offer an analysis on specific failure modes of inference-only adaptive methods, showing the necessity of our training-time adaptive designs.

## 2 Related Work

**(Any-order) Autoregressive Models and (Discrete) Diffusion Models.** Autoregressive and diffusion models are currently the two dominant families of generative models in various domains [13, 14, 15]. For textual modeling, in recent years, there has been a seismic rise in popularity for large language models [1, 2, 3] of which the vast majority are trained using the next-token prediction objective, or *causal language modeling*, e.g. GPT [16, 17]. Alternatively, discrete variants [5, 4] of diffusion models [18, 19, 20] like masked diffusion models (MDMs) apply a denoising objective to randomly corrupted data to learn the underlying structures. Furthermore, any-order autoregressive models (AO-ARMs) [21, 22] generalize the causal modeling paradigm by allowing arbitrary orders or masks during training and inference. Compared with these approaches, ReCOR uses a learned adaptive order during both training and inference, avoiding issues with fixed or random orders. We primarily compare to adaptive inference variants of MDMs [11, 12] since AO-ARMs are trained similarly.

**Token Ordering and Reflections on Causal Language Modeling.** Within the active research field of language models, there have long been criticisms and attempts at improvements regarding various aspects of the causal modeling paradigm, including autoregressive inference [23], teacher-forcing

training [24], and the left-to-right order [25]. In particular, current LLMs have been found to struggle with many complex reasoning and planning problems, especially adaptive ones [26, 10, 27, 28]. We focus on token ordering, motivated by the fact that these reasoning problems often require intricate, data-dependent orders to be tractably solved. Recently, adaptive inference methods based on MDMs [11, 12] were proposed to address this problem, combining random masking at training time with selective token inference. We note that adaptive orders are needed during both training and inference, and design ReCOR to automatically recover the correct order with self-supervision.

# 3 Preliminary

## 3.1 Problem Setup

We focus our attention on the classic sequence-to-sequence setting, where the goal is to learn a conditional distribution $p(\mathbf{y} \mid \mathbf{x})$ with $\mathbf{x} \in \mathcal{X} \subseteq \mathcal{T}^*$ as the prompt (space), $\mathbf{y} \in \mathcal{Y} \subseteq \mathcal{T}^*$ as the response (space), and $\mathcal{T}$ as the token vocabulary. For simplicity, we assume prompts of length $N$ and responses of length $M$: $\mathbf{x} = (x_1, x_2, \ldots, x_N) \in \mathcal{X} = \mathcal{T}^N, \mathbf{y} = (y_1, y_2, \ldots, y_M) \in \mathcal{Y} = \mathcal{T}^M$ which can be achieved without loss of generality through padding. A training dataset $D = \{(\mathbf{x}, \mathbf{y})\}$ is available with i.i.d. samples $(\mathbf{x}, \mathbf{y}) \sim p(\mathbf{x})p(\mathbf{y} \mid \mathbf{x})$.

## 3.2 Existing Approaches

In recent years, **causal language models** (CLMs) have witnessed a giant wave of interest, trained using the prevalent **next-token prediction** (NTP) objective:

$$\min_{\theta} \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim D} \left[ -\sum_{i=1}^{M} \log p_\theta(y_i \mid \mathbf{x}, \mathbf{y}_{<i}) \right] \tag{1}$$

with $p_\theta(\cdot \mid \mathbf{x}, \mathbf{y}_{<i})$ parameterized as an autoregressive sequence model (e.g. GPT [16, 17]).

As a generalization of CLMs, **any-order autoregressive models** (AO-ARMs) [21, 22] perform "next-token" predictions under an arbitrary generation order $\boldsymbol{\rho} \in S_M$ instead of the causal order:

$$\min_{\theta} \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim D, \boldsymbol{\rho}\sim\mathcal{U}_{\boldsymbol{\rho}}} \left[ -\sum_{i=1}^{M} \log p_\theta(y_{\rho_i} \mid \mathbf{x}, \mathbf{y}_{\boldsymbol{\rho}_{<i}}, \rho_i) \right] \tag{2}$$

where $\mathcal{U}_{\boldsymbol{\rho}}$ is the distribution of training orders, usually taken as the uniform distribution over all $M$-permutations $S_M$ [21] or further canonicalized using simple rules [22]. Note that $\boldsymbol{\rho}$ is the **generation order** and distinct from the raw **content order**, which is still retained; i.e. for $\mathbf{y} = \texttt{abc}$ and $\rho_1 = 3$, the partially generated response at the first step should be `**c` instead of `c**`.

Alternatively, **masked diffusion models** (MDMs) [5, 4, 29, 12] replace some response tokens with a special `[MASK]` token in a forward process $q_{t|0}(\mathbf{y}^t \mid \mathbf{y})$ that masks each token independently at random, then learn to predict the masked tokens with a denoising objective (constants omitted):

$$\min_{\theta} \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim D, t\sim\mathcal{U}[0,1], \mathbf{y}^t\sim q_{t|0}(\cdot|\mathbf{y})} \left[ -\sum_{i=1}^{M} \mathbb{I}[y_i^t = \texttt{[MASK]}] \log p_\theta(y_i \mid \mathbf{x}, \mathbf{y}^t, t, i) \right] \tag{3}$$

which is also often implemented with a transformer.

# 4 Problem Analysis and Method

This section is structured as follows: We first motivate our focus on generation order by characterizing *token hardness* (Sec. 4.1) and propose to cast order recovery as a decision-making problem (Sec. 4.2). The core training algorithm for ReCOR is presented in Sec. 4.3 with architectural details in Sec. 4.4.

## 4.1 Generation Order and Token Hardness

We begin our discussion about the order problem by noting that prior works mostly handle generation orders passively during training, delegating the issue to a fixed data-independent strategy (Sec. 3.2,

always left-to-right for CLM and uniformly random for AO-ARM and MDM). This is justified from a probabilistic standpoint, where any joint distribution over a series of random variables can be decomposed arbitrarily into the product of a series of conditional distributions via the chain rule: $P(y_1, y_2, \ldots, y_M) = \prod_{i=1}^{M} P(y_{\rho_i} \mid y_{\rho_1}, \ldots, y_{\rho_{i-1}}), \forall \boldsymbol{\rho} \in S_M$. Furthermore, for problems with a unique solution, the prompt can completely determine each response token, and we have $H(Y_{\rho_i} \mid X, \{Y_{\rho_j}\}_{j<i}) \leq H(Y_{\rho_i} \mid X) = 0, I(Y_{\rho_i}; \{Y_{\rho_j}\}_{j<i} \mid X) = 0, \forall \boldsymbol{\rho}, i$, so predicted response tokens do not bring additional information, also implying that order is irrelevant.

However, as illustrated in Fig. 1 and observed by prior works [10, 12, 11], for hard reasoning and planning problems, plain causal modelling frequently encounters computationally intractable intermediate steps and fails dramatically; a tailored generation order is required for each instance to make the problem tractable in practice. Consequently, we choose to explicitly model the generation order $\boldsymbol{\rho}$ as an unobserved variable to be recovered from the plain-text-only training data.

Under such a formulation, the problem then turns to defining and finding a *good* $\boldsymbol{\rho}$. Intuitively, we would like to start with the easy parts of the response and work our way to the harder parts step-by-step, utilizing the partially generated solution as a form of chain-of-thought to guide later generations. We formalize this intuition using the framework of predictive $\mathcal{V}$-information [30]:

**Definition 1** (Predictive $\mathcal{V}$-information [30])**.** Let $\mathcal{V} \subseteq \{f : \mathcal{A} \cup \{\varnothing\} \to \Delta_{\mathcal{B}}\}$ be a **predictive function class** that predicts the distribution of a random variable taking values over $\mathcal{B}$, optionally given the value of another random variable over $\mathcal{A}$ under some regularity conditions.

Define the **predictive conditional $\mathcal{V}$-entropy** of random variables $A, B$ as

$$H_{\mathcal{V}}(B \mid A) = \inf_{f \in \mathcal{V}} \mathbb{E}_{a,b \sim A,B}[-\log f[a](b)] \tag{4}$$

$$H_{\mathcal{V}}(B \mid \varnothing) = \inf_{f \in \mathcal{V}} \mathbb{E}_{b \sim B}[-\log f[\varnothing](b)] \tag{5}$$

Then the **predictive $\mathcal{V}$-information** from random variable $A$ to $B$ is defined as

$$I_{\mathcal{V}}(A \to B) = H_{\mathcal{V}}(B \mid \varnothing) - H_{\mathcal{V}}(B \mid A). \tag{6}$$

Def. 1 captures the *hardness* of a token given the current context under computational constraints, which evades standard probabilistic and information-theoretic arguments. Given prompt $X$ and a set of already predicted response tokens $\{Y_{\rho_j}\}_{j<i}$, an easy token $Y_{\rho_i}$ would have a large $I_{\mathcal{V}}(X, \{Y_{\rho_j}\}_{j<i} \to Y_{\rho_i})$ whereas a hard token would have a small $\mathcal{V}$-information. Building on this characterization, we set our overall objective to maximize the following cumulative predictive $\mathcal{V}$-information using a learned autoregressive $\boldsymbol{\rho}$-generator parameterized by $\theta$:

$$\max_{\theta} \sum_{i=1}^{M} \mathbb{E}_{\rho_i \sim p_\theta(\cdot \mid X, \{Y_{\rho_j}\}_{j<i})} I_{\mathcal{V}}(X, \{Y_{\rho_j}\}_{j<i} \to Y_{\rho_i}) \tag{7}$$

Note that Def. 1 of $\mathcal{V}$-information contains an optimization problem. Naively training a new predictor to solve the optimization problem within $I_{\mathcal{V}}(X, \{Y_{\rho_j}\}_{j<i} \to Y_{\rho_i})$ for every $\boldsymbol{\rho}_{\leq i}$ would require an exponential number of training instances. For tractability, we set the predictive function class $\mathcal{V}$ to the family of autoregressive token generators conditioned on arbitrary contexts and parameterized by $\psi$, and share it across inner subproblems for all $\boldsymbol{\rho}$, yielding the following objective up to constants:

$$\max_{\theta, \psi} \mathbb{E}_{(\mathbf{x}, \mathbf{y}) \sim D} \left[ \sum_{i=1}^{M} \mathbb{E}_{\rho_i \sim p_\theta(\cdot \mid \mathbf{x}, \mathbf{y}_{\boldsymbol{\rho}_{<i}})} \log p_\psi(y_{\rho_i} \mid \mathbf{x}, \mathbf{y}_{\boldsymbol{\rho}_{<i}}, \rho_i) \right] \tag{8}$$

### 4.2 Order Recovery as a Decision-making Problem

We now proceed to solve Eq. 8 for $\theta$ and the associated $\psi$. Although [30] proposed a method for structure learning based on the Chu-Liu algorithm [31], it assumes that the edge weight depends only on the two vertices connected by the edge, while under our setting, the likelihood depends on all random variables in the context with an exponential number of combinations. Furthermore, structure learning requires a uniform structure over the whole dataset, while the correct generation order can be data-dependent and vary between instances (e.g. Sudoku [10] in Fig. 1).

Facing these challenges, we formulate the recovery of $\boldsymbol{\rho}$ as a decision-making problem and employ reinforcement learning (RL) techniques [32, 33, 34] to train the $\rho$-generator. We construct the

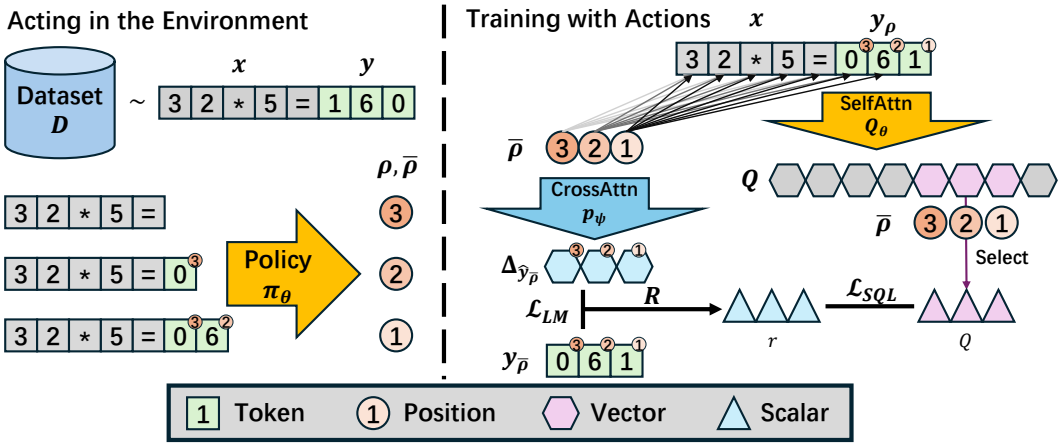



Figure 2: Illustration of the training procedure of ReCOR. ReCOR first rolls out the order prediction policy $\pi_\theta$ autoregressively on sampled data $(\mathbf{x}, \mathbf{y})$ to obtain the actions $\boldsymbol{\rho}, \bar{\boldsymbol{\rho}}$ (left), then performs parallelized training on the sampled actions (right). The token predictor $p_\psi$ is trained to answer queries generated by $\pi_\theta$ using $\mathcal{L}_{\text{LM}}$ while $Q_\theta$ is supervised by reward signals from $p_\psi$ using $\mathcal{L}_{\text{SQL}}$.

following Markov decision process (MDP) $(\mathcal{S}, \mathcal{A}, P, R, p_1, \gamma)$ where $\mathcal{S} := \mathcal{X} \times \cup_{I \subseteq [M]} \mathcal{T}^I$ is the state space with the prompt and a partially generated response; $\mathcal{A} := [M]$ is the action space corresponding to the position of the next token to generate; $P$ is the transition that adds the newly generated token to the state; $R$ is the reward function as in Eq. 14 and 15; $p_1 := p(\mathbf{x}) \times \delta(\varnothing)$ is the initial state distribution consisting of a sampled prompt and an empty response; and $\gamma \in [0, 1]$ is the discount factor. Concretely, at time step $1 \le t \le M$, $s_t = (\mathbf{x}, \mathbf{y}_{\boldsymbol{\rho}_{<t}}), a_t = \rho_t$, which transitions to $(\mathbf{x}, \mathbf{y}_{\boldsymbol{\rho}_{\le t}})$ with $y_{\rho_t}$ taken from the training data or sampled from $p_\psi(\cdot \mid \mathbf{x}, \mathbf{y}_{\boldsymbol{\rho}_{<t}}, \rho_t)$ during inference. Under this formulation, the objective is to train a policy $\pi_\theta : \mathcal{S} \to \Delta_{\mathcal{A}}$ satisfying

$$\max_\theta \mathbb{E}_{s_1 \sim p_1, a_t \sim \pi_\theta(\cdot|s_t), r_t \sim R(s_t,a_t), s_{t+1} \sim P(s_t,a_t)} \left[ \sum_t \gamma^t r_t \right] \tag{9}$$

which recovers Eq. 8 with $\pi_\theta(\cdot \mid s_t) = p_\theta(\cdot \mid \mathbf{x}, \mathbf{y}_{\boldsymbol{\rho}_{<t}}), R(s_t, a_t) = \log p_\psi(y_{\rho_t} \mid \mathbf{x}, \mathbf{y}_{\boldsymbol{\rho}_{<t}}, \rho_t)$ and $\gamma = 1$. This formulation also allows more room for design choices, e.g. RL algorithms, discount factor, and alternative reward functions; see Sec. 4.3 below for details.

Equipped with $\pi_\theta$ for order prediction and $p_\psi$ for token prediction, we can now perform inference by sampling $\rho_t \sim \pi_\theta(\cdot \mid \mathbf{x}, \hat{\mathbf{y}}_{\boldsymbol{\rho}_{<t}})$ and $\hat{y}_{\rho_t} \sim p_\psi(\cdot \mid \mathbf{x}, \hat{\mathbf{y}}_{\boldsymbol{\rho}_{<t}}, \rho_t)$ autoregressively (see Alg. 2).

### 4.3 Reinforced Training for Adaptive Order Prediction

With an RL formulation in place to solve Eq. 9, as illustrated in Fig. 2, we use a discrete version of soft Q-learning [33] that optimizes the following entropy-regularized discounted-return objective:

$$\max_\theta \mathbb{E}_{\pi_\theta} \left[ \sum_t \gamma^t (r_t + \alpha H(\pi_\theta(\cdot \mid s_t))) \right] \tag{10}$$

where $\alpha \ge 0$ is the entropy coefficient. This objective balances exploration and exploitation under a maximum entropy RL framework by requiring the policy to act as stochastically as possible while optimizing returns. In addition, since our action space $\mathcal{A} = [M]$ is discrete and tractable, we can efficiently compute $\pi_\theta$ and $V_\theta$ from $Q_\theta$ without separately learning the policy and value function through function approximation. To optimize this objective, we use soft Bellman update [33] implemented with the following mean squared error loss:

$$\mathcal{L}_{\text{SQL-MSE}}(\theta) := \mathbb{E}_{s,a,r,s'} \left[ (Q_\theta(s,a) - \bar{Q}_\theta(s,a))^2 \right] \tag{11}$$

where $\bar{Q}_\theta(s,a) := r + \gamma V_\theta(s') = r + \gamma\alpha \log \sum_{a'} \exp(Q_\theta(s,a')/\alpha)$ is the Q-value target with gradients detached. Note that we do not use target network techniques.

Alternatively, we can also use the binary cross-entropy loss as value loss:

$$\mathcal{L}_{\text{SQL-BCE}}(\theta) := \mathbb{E}_{s,a,r,s'}\left[-\exp\bar{Q}_\theta(s,a)\cdot Q_\theta(s,a) - (1-\exp\bar{Q}_\theta(s,a))\cdot\log(1-\exp Q_\theta(s,a))\right] \tag{12}$$

which may deliver better robustness to outliers for $0 \le \exp\bar{Q}_\theta(s,a) \le 1$ and is compatible with sparse reward functions, as discussed below.

Finally, we discuss the optimization of the token predictor parameters $\psi$ and the choice of reward function. We train the token predictor $p_\psi$ online along with the policy $\pi_\theta$ with the following (permuted) language modelling loss, echoing Eq. 8:

$$\mathcal{L}_{\text{LM}}(\psi) := \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim D,\boldsymbol{\rho}\sim\pi_\theta}\left[\sum_{i=1}^{M} -\log p_\psi(y_{\rho_i} \mid \mathbf{x}, \mathbf{y}_{\boldsymbol{\rho}_{<i}}, \rho_i)\right] \tag{13}$$

Note that $p_\psi$ is trained on $\boldsymbol{\rho}$ sampled from the current policy $\pi_\theta$ to ensure that it always stays on-policy. As $\mathcal{L}_{\text{LM}}$ is supervised, $p_\psi$ learns much faster than the RL-trained $\pi_\theta$, and we treat $\psi$ as always being near the optimality and approximate $I_{\mathcal{V}}(X, \{Y_{\rho_j}\}_{j<i} \to Y_{\rho_i})$ with the current $\log p_\psi(y_{\rho_i} \mid \mathbf{x}, \mathbf{y}_{\boldsymbol{\rho}_{<i}}, \rho_i)$. As a result, we may set the reward function to the (negated) perplexity:

$$R(s_t, a_t) = R_{\text{ppl}}((\mathbf{x}, \mathbf{y}_{\boldsymbol{\rho}_{<t}}), \rho_t) := \log p_\psi(y_{\rho_t} \mid \mathbf{x}, \mathbf{y}_{\boldsymbol{\rho}_{<t}}, \rho_t) \tag{14}$$

Empirically, we found the following thresholded, sparse reward with a better performance:

$$R(s_t, a_t) = R_{\text{spr}}((\mathbf{x}, \mathbf{y}_{\boldsymbol{\rho}_{<t}}), \rho_t) := \log \mathbb{I}[p_\psi(y_{\rho_t} \mid \mathbf{x}, \mathbf{y}_{\boldsymbol{\rho}_{<t}}, \rho_t) \ge \eta] \tag{15}$$

where $\eta \in (0, 1)$ is the probability threshold. Note, however, that the logarithm is undefined when $p_\psi(y_{\rho_i} \mid \mathbf{x}, \mathbf{y}_{\boldsymbol{\rho}_{<i}}, \rho_i) < \eta$; we use this reward in conjunction with $\mathcal{L}_{\text{SQL-BCE}}$, which only requires $\exp R_{\text{spr}}((\mathbf{x}, \mathbf{y}_{\boldsymbol{\rho}_{<t}}), \rho_t) := \mathbb{I}[p_\psi(y_{\rho_i} \mid \mathbf{x}, \mathbf{y}_{\boldsymbol{\rho}_{<i}}, \rho_i) \ge \eta]$ that is well-defined.

Training pseudocode is presented in Alg. 1. Furthermore, since the dynamics of the MDP we defined are fully known, we can sample potentially multiple actions $\bar{\boldsymbol{\rho}}_{t,1\ldots K}$ at the same state $(\mathbf{x}, \mathbf{y}_{\boldsymbol{\rho}_{<t}})$ and optimize $\mathcal{L}_{\text{LM}}$ and $\mathcal{L}_{\text{SQL}}$ for all of the $K$ actions to improve learning quality. See Sec. 5.6 for details.

### 4.4 Multi-stream Architecture for Order and Token Predictions

With the algorithm of ReCOR described above, we now instantiate the $\rho$-generator $\pi_\theta(\cdot \mid \mathbf{x}, \mathbf{y}_{\boldsymbol{\rho}_{<t}})$ and token predictor $p_\psi(\cdot \mid \mathbf{x}, \mathbf{y}_{\boldsymbol{\rho}_{<t}}, \rho_t)$. For our experiments, we use GPT-2 [16] as the backbone, which could be replaced with other transformer variants. To encode a permuted response, we add the absolute positional embedding of GPT-2 at position $\rho_i$ to token $y_{\rho_i}$, while the prompt $\mathbf{x}$ is encoded as a regular left-to-right sequence before the response tokens. We use a full mask on the prompt $\mathbf{x}$ and a causal mask on the response $\mathbf{y}_{\boldsymbol{\rho}}$ to enable parallelized training and KV-cached inference.

This suffices to handle the sequential inputs $\mathbf{x}, \mathbf{y}_{\boldsymbol{\rho}}$. The token predictor $p_\psi(\cdot \mid \mathbf{x}, \mathbf{y}_{\boldsymbol{\rho}_{<t}}, \rho_t)$ takes an additional scalar query position $\rho_t$ as input at every time step $t$. Ideally, we'd like the queries at every time step not to interfere with each other; consequently, we adopt a multi-stream architecture conceptually similar to XLNet [21] where a **main stream** takes $\mathbf{x}, \mathbf{y}_{\boldsymbol{\rho}}$ as input and performs self-attention as in regular transformers, and a **token query stream** takes $\bar{\boldsymbol{\rho}}$ as queries and cross-attends onto the main stream hidden states as keys and values without self-attention between queries. This guarantees that every query would be independent of others. We produce token predictions via a token head on the outputs of the token query stream. For order predictions $\pi_\theta$, we can directly use the main stream outputs since order predictions do not depend on additional positions. To further enhance expressiveness, we can optionally use an **order query stream** instead of the main stream to perform order predictions. With $C$ learnable order query embeddings $q_{1\ldots C}$ as inputs, we forward the order query stream for $C$ times and concatenate the outputs before projecting to Q values. This design allows us to scale compute for better performance; see Sec. 5.6 for details.

## 5 Experiments

In this section, we aim to answer the following questions with our experiments: 1) Can ReCOR solve arithmetic problems without special data preprocessing? 2) Can ReCOR solve reasoning and planning problems adaptively without annotations? 3) Do we need adaptive orders during training or for inference only? 4) How does ReCOR compare with the state-of-the-art methods under fair inference compute settings? 5) Can the performance of ReCOR scale with more compute?

Table 1: Performance of ReCOR and baselines on arithmetic datasets. ReCOR outperforms baselines and is competitive with the oracle.

| Task | AR-GT | ReCOR | CLM | MDM | AdaMDM |
|---|---|---|---|---|---|
| ARG | $0.994 \pm 0.003$ | $\mathbf{0.987 \pm 0.007}$ | $0.017 \pm 0.002$ | $0.035 \pm 0.010$ | $0.174 \pm 0.036$ |
| MUL | $0.999 \pm 0.001$ | $\mathbf{0.964 \pm 0.007}$ | $0.594 \pm 0.018$ | $0.943 \pm 0.006$ | $0.951 \pm 0.038$ |

Table 2: Performance of ReCOR and baselines on puzzle datasets. ReCOR outperforms all approaches, even including the oracle AR-GT supervised with ground-truth orders.

| Task | AR-GT | ReCOR | CLM | MDM | AdaMDM |
|---|---|---|---|---|---|
| Sudoku | 0.8718 | $\mathbf{0.9017 \pm 0.0004}$ | 0.0973 | 0.0688 | 0.8949 |
| Zebra | 0.9117 | $\mathbf{0.9905 \pm 0.0021}$ | 0.8031 | 0.769 | 0.985 |

### 5.1 Experiment Setup

We chose the following datasets to validate ReCOR's performance at adaptive reasoning: 1) Arithmetic datasets, including synthetic autoregression and multiplication, which are originally generated in reverse order during data generation. 2) Puzzle datasets, including Sudoku and Zebra [10], which may require an adaptive, data-dependent order to solve. We use accuracy (full response match) as the main evaluation metric and report standard deviations over 3 training seeds.

We compare with the following representative baselines: 1) Causal language models (CLM) [16] that always follow a left-to-right order. This baseline serves to demonstrate the necessity of non-left-to-right generation orders. 2) Autoregressive models supervised with ground-truth generation orders (AR-GT) [10] that is an oracle for ReCOR, as ReCOR does not have access to the ground-truth order during both training and inference. 3) Masked diffusion models (MDM) [5, 4] that randomly mask the input and perform denoising, in effect trying to learn arbitrary orders. 4) Adaptive masked diffusion models (AdaMDM) [12, 11] that use recent state-of-the-art adaptive inference methods.

### 5.2 Can ReCOR solve arithmetic problems without special data preprocessing?

We start with arithmetic problems generated using a fixed, non-causal order. We employ a synthetic Autoregression (ARG) task where each response token depends on the prompt and all response tokens after it, and a multiplication (MUL) task between 20-digit and 2-digit numbers. It has been shown that CLMs often need manually reversed training data or additional CoT annotations to solve certain arithmetic problems [35, 36, 37, 38] due to the reverse dependencies between tokens brought by carry digits. In this work, we are interested in whether ReCOR can automatically recover the correct order without manual preprocessing or additional annotations. Consequently, we apply the reversed order only to AR-GT and keep the natural order for the rest of the approaches.

As shown in Tab. 1, ReCOR achieves competitive performance compared with all baselines and outperforms CLMs by a large margin, showing that the plain next-token prediction objective is not sufficient for solving these problems. Notably, ReCOR is competitive with AR-GT, which uses the ground-truth order during both training and inference. This demonstrates its ability to recover order in a self-supervised manner. Furthermore, ReCOR outperforms (Ada)MDM significantly in Autoregression; we compare both methods in more detail in Sec. 5.4.

### 5.3 Can ReCOR solve reasoning and planning problems adaptively without annotations?

In this section, we study classic logic puzzles Sudoku and Zebra [10]. The two datasets are examples of problems that require adaptive, data-dependent reasoning and planning, rendering them difficult for approaches that generate solutions with a fixed order. For example, in Sudoku, we may need to fill certain cells first in order to eliminate candidates for other cells and arrive at the correct answer, as illustrated in Fig. 1. The cell dependency structure varies between instances, posing a great challenge.

We show results on Sudoku and Zebra in Tab. 2. Values without standard deviations are reported by [12]. ReCOR is the best among all approaches, outperforming even the oracle AR-GT. We remark

Table 3: Impact of *suffix unmasking* for MDMs on Autoregression. Suffix unmasking dramatically improves the performance of AdaSufMDM over AdaMDM, but still lags behind ReCOR.

| ReCOR | SufMDM | AdaSufMDM | MDM | AdaMDM |
|---|---|---|---|---|
| $\mathbf{0.987 \pm 0.007}$ | $0.040 \pm 0.005$ | $\underline{0.536 \pm 0.051}$ | $0.035 \pm 0.010$ | $0.174 \pm 0.036$ |

Table 4: Performance of ReCOR and Ada(Suf)MDM under different inference compute settings. MDM underperforms ReCOR even with 10x compute, and is much worse with the same compute. We use AdaMDM on MUL and AdaSufMDM on ARG, as AdaMDM fails on ARG.

| Method | ReCOR | Ada(Suf)MDM | |
|---|---|---|---|
| FLOPs | 1x | 1x($T = 2$) | 10x($T = 20$) |
| ARG | $\mathbf{0.987 \pm 0.007}$ | $0.028 \pm 0.009$ | $0.536 \pm 0.051$ |
| MUL | $\mathbf{0.964 \pm 0.007}$ | $0.875 \pm 0.043$ | $0.951 \pm 0.038$ |

that at every time step, ReCOR can generate an estimated reward for every unfilled cell, which is a denser and more diverse signal than the single next position offered by the ground truth, potentially contributing to its superior performance. CLM and vanilla MDM perform similarly and are both worse than ReCOR and AdaMDM, showing that an adaptive generation order is indeed necessary.

### 5.4 Do we need adaptive orders during training or for inference only?

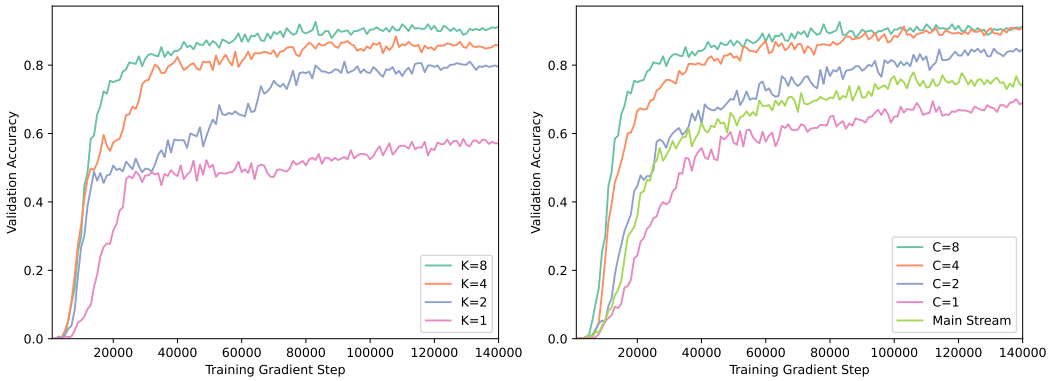
We look closer into the difference between ReCOR and AdaMDM [12, 11], which are recent, state-of-the-art methods also focused on the order problem. Both works showed that MDMs encounter intractable sub-problems due to random masking during training, and proposed to use adaptive decoding strategies during inference to prioritize easy tokens and circumvent these hard scenarios. One of the core differences between ReCOR and AdaMDM is that AdaMDM is trained under random masking while ReCOR follows the recovered order during both training and inference. Thus a problem arises: *do we really need adaptive orders during training?*

We answer the question in the affirmative with the Autoregression task (Tab. 1). In Autoregression, each response token is generated conditioned on *all* response tokens behind it, and consequently, can only be tractably predicted when all of these tokens are present in the context. Similar properties hold for many long-horizon reasoning problems with multiple steps and dense dependencies between the steps. However, under random masking, the probability of such an event grows exponentially smaller with the length of the context, and MDMs rarely see any long, complete contexts during training. As a result, **MDMs are bad at the corresponding sub-problems, even though they are supposedly easy and tractable**, leading to a bad overall performance as evidenced in Tab. 1. In contrast, **ReCOR automatically recognizes and frequently visits these good sub-problems during training**, making sure that the on-policy token predictor $p_\psi$ can reliably solve them.

To further support this analysis, we propose (Ada)SufMDM, a variant of (Ada)MDM that *unmasks* a suffix of length $l \sim \mathcal{U}_{[M]} - 1$ during training. This operation dramatically boosts the probability of each "correct" sub-problem from $O(c^M)$ to $\Omega(M^{-1})$. Note that this fix applies only to ARG, where the correct order is known to be reversed; applying the operation in general would require ground-truth order annotations. The updated results are shown in Tab. 3. SufAdaMDM obtains a large performance improvement and still outperforms SufMDM, demonstrating that correct orders are important during both training and inference. However, there remains a gap between AdaSufMDM and ReCOR, which is likely due to the additional intractable sub-problems brought by random masking, as argued [12]. Our analysis complements their findings in that **random masking leads to not only many intractable sub-problems, but also a dearth of tractable ones**.

### 5.5 How does ReCOR compare with the state-of-the-art methods under fair inference compute settings?

We show another advantage of our autoregressive design by comparing ReCOR with MDMs under different inference compute settings. The standard MDM inference setting used in our primary




(a) Token query scaling with $C = 8$.

(b) Order query scaling with $K = 8$.

Figure 3: ReCOR's performance when scaling the number of token queries $K$ (a) and order queries $C$ (b). Main Stream in (b) denotes using the main stream outputs without a separate order query stream. ReCOR can improve its performance with more computation during training and inference.

experiments employs a large number of diffusion steps; consequently, many transformer forward passes are needed during inference. We note that for inherently serial problems where tokens need to be generated one-by-one, MDMs require diffusion steps to the order of $\Omega(M)$, bringing the overall time complexity to $\Omega(M^3)$. In comparison, since ReCOR is autoregressive and uses a causal mask for the response, we can use KV-caching during inference, reducing the total number of complete transformer forward passes to $2 + C$ where $C$ is the number of order stream queries and a constant with respect to the context length, maintaining an $O(M^2)$ time complexity.

Empirically, we infer MDMs on the arithmetic datasets under alternative inference compute settings in Tab. 4. ReCOR uses $C = 0$ for these datasets, leading to an inference FLOPs equivalent to 2 complete transformer forward passes. When restricting the FLOPs of MDMs to the same level, the number of diffusion steps is limited to $T = 2$, dramatically reducing performance.

### 5.6 Can the performance of ReCOR scale with more compute?

In this section, we demonstrate scaling properties arising from the designs of ReCOR. As described in Sec. 4, in contrast to the standard RL setting, ReCOR can take $K > 1$ actions at the same state and learn from all of these actions, since the MDP we defined can be easily simulated perfectly. Furthermore, we can use $C$ queries to the order query stream to obtain a more expressive Q function. These designs allow us to scale compute to improve performance even with the underlying backbone parameter count fixed.

To demonstrate the scaling properties, we perform ablation experiments on Sudoku that vary one of $K$ or $C$ from the primary setting $K = C = 8$. We show training curves of validation accuracy in Fig. 3. It can be seen that in both groups of experiments, performance improves monotonically with the varying parameter. As a special case, directly using the main stream outputs to compute Q values (Main Stream in Fig. 3b) achieves decent performance between $C = 1$ and 2 without using separate order queries. This makes it a good, lightweight choice that we adopt for the arithmetic datasets.

### 5.7 How does ReCOR compare with large pretrained models?

To further demonstrate the performance of ReCOR, we compare with o3, a state-of-the-art pretrained large reasoning model by OpenAI, on our ARG dataset. Since we could not fine-tune o3, we take the following two setups to evaluate its performance:

1. Prompting o3 with in-context examples and a detailed description of the ground-truth data generation algorithm.
2. Prompting o3 with in-context examples only.

We evaluate 50 random samples from the test set and allocate a token budget of 10000 tokens each. Note that in our main experiments, only the generated training data is available to the models without

the ground-truth generation rule, corresponding to the harder setup 2 above. As a result, under setup 1 with the ground-truth algorithm annotation, o3 achieves a success rate of 0.3, where the majority of failures come from the depletion of token budgets. Even considering only the completely generated solutions, the success rate of o3 is 0.938 and is still lower than ReCOR's 0.987. Furthermore, o3 takes about 4 minutes on average to solve each problem instance, while ReCOR solves the entire 1000-sample test set within 10 seconds. Under setup 2 with no access to ground-truth rules, o3 fails to recover the correct data generation algorithm and has a success rate of 0. We stress again that ReCOR operates under this harder setting without any additional annotations. The results show that even large pretrained reasoning models may struggle to reason about token ordering effectively and efficiently, especially when relevant annotations are missing from the training data.

## 6 Conclusion and Limitations

In this paper, we introduce Reinforced Context Order Recovery (ReCOR), a reinforcement-learning-based algorithm for automatically recovering the correct generation order from textual data without annotations. While modern text generation models are predominantly causal, they have been shown to fail in adaptive reasoning problems due to the presence of intractable tokens. We characterize this phenomenon with the framework of $\mathcal{V}$-information and propose to optimize the corresponding objective to learn a suitable data-dependent generation order. Subsequently, we operationalize this objective by introducing ReCOR with reinforcement learning setups, recovering unobserved generation order from purely textual data in a self-supervised manner. Empirically, ReCOR outperforms strong baselines on various datasets, including recent adaptive inference approaches using masked diffusion models and oracle models supervised with the ground-truth order.

There are certain limitations and future works for ReCOR. We primarily run experiments on reasoning and planning-related datasets, echoing the setups of our baselines. Going forward, we are excited to further scale up ReCOR to larger parameter counts and more diverse problems and datasets with more compute available. Applying ReCOR to modern LLMs and multi-modal models is the path forward to fully realizing its potential. The RL-based formulation also opens up a vast space to integrate with other RL techniques and further improve the performance of ReCOR, which we did not exhaust due to limited resources. Furthermore, there could be more unobserved variables driving the generation of textual data beyond the generation order. The recovery of these variables presents exciting opportunities for future explorations.

## Acknowledgements

This work was supported by the National Science and Technology Major Project (2022ZD0114904), NSFC-6247070125, NSFC-62406010, the State Key Lab of General Artificial Intelligence at Peking University, the Fundamental Research Funds for the Central Universities, and Qualcomm University Research Grant.

## References

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

[2] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

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

[4] Emiel Hoogeboom, Didrik Nielsen, Priyank Jaini, Patrick Forré, and Max Welling. Argmax flows and multinomial diffusion: Learning categorical distributions. *Advances in neural information processing systems*, 34:12454–12465, 2021.

[5] Jacob Austin, Daniel D Johnson, Jonathan Ho, Daniel Tarlow, and Rianne Van Den Berg. Structured denoising diffusion models in discrete state-spaces. *Advances in neural information processing systems*, 34:17981–17993, 2021.

[6] Shen Nie, Fengqi Zhu, Zebin You, Xiaolu Zhang, Jingyang Ou, Jun Hu, Jun Zhou, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. Large language diffusion models. *arXiv preprint arXiv:2502.09992*, 2025.

[7] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

[8] Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, et al. A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18(6):186345, 2024.

[9] Daya Guo, Qihao Zhu, Dejian Yang, Zhenda Xie, Kai Dong, Wentao Zhang, Guanting Chen, Xiao Bi, Yu Wu, YK Li, et al. Deepseek-coder: When the large language model meets programming–the rise of code intelligence. *arXiv preprint arXiv:2401.14196*, 2024.

[10] Kulin Shah, Nishanth Dikkala, Xin Wang, and Rina Panigrahy. Causal language modeling can elicit search and reasoning capabilities on logic puzzles. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[11] Jiacheng Ye, Jiahui Gao, Shansan Gong, Lin Zheng, Xin Jiang, Zhenguo Li, and Lingpeng Kong. Beyond autoregression: Discrete diffusion for complex reasoning and planning. In *The Thirteenth International Conference on Learning Representations*, 2025.

[12] Jaeyeon Kim, Kulin Shah, Vasilis Kontonis, Sham Kakade, and Sitan Chen. Train for the worst, plan for the best: Understanding token ordering in masked diffusions. *arXiv preprint arXiv:2502.06768*, 2025.

[13] Ling Yang, Zhilong Zhang, Yang Song, Shenda Hong, Runsheng Xu, Yue Zhao, Wentao Zhang, Bin Cui, and Ming-Hsuan Yang. Diffusion models: A comprehensive survey of methods and applications. *ACM Computing Surveys*, 56(4):1–39, 2023.

[14] Hanqun Cao, Cheng Tan, Zhangyang Gao, Yilun Xu, Guangyong Chen, Pheng-Ann Heng, and Stan Z Li. A survey on generative diffusion models. *IEEE Transactions on Knowledge and Data Engineering*, 2024.

[15] Jing Xiong, Gongye Liu, Lun Huang, Chengyue Wu, Taiqiang Wu, Yao Mu, Yuan Yao, Hui Shen, Zhongwei Wan, Jinfa Huang, et al. Autoregressive models in vision: A survey. *arXiv preprint arXiv:2411.05902*, 2024.

[16] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[17] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 33:1877–1901, 2020.

[18] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

[19] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in neural information processing systems*, 32, 2019.

[20] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021.

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

[22] Andy Shih, Dorsa Sadigh, and Stefano Ermon. Training and inference on any-order autoregressive models the right way. *Advances in Neural Information Processing Systems*, 35:2762–2775, 2022.

[23] Nouha Dziri, Ximing Lu, Melanie Sclar, Xiang Lorraine Li, Liwei Jiang, Bill Yuchen Lin, Sean Welleck, Peter West, Chandra Bhagavatula, Ronan Le Bras, et al. Faith and fate: Limits of transformers on compositionality. *Advances in Neural Information Processing Systems*, 36:70293–70332, 2023.

[24] Gregor Bachmann and Vaishnavh Nagarajan. The pitfalls of next-token prediction. In *International Conference on Machine Learning*, pages 2296–2318. PMLR, 2024.

[25] Li Du, Hongyuan Mei, and Jason Eisner. Autoregressive modeling with lookahead attention. *arXiv preprint arXiv:2305.12272*, 2023.

[26] Karthik Valmeekam, Matthew Marquez, Sarath Sreedharan, and Subbarao Kambhampati. On the planning abilities of large language models-a critical investigation. *Advances in Neural Information Processing Systems*, 36:75993–76005, 2023.

[27] Subbarao Kambhampati, Karthik Valmeekam, Lin Guan, Mudit Verma, Kaya Stechly, Siddhant Bhambri, Lucas Paul Saldyt, and Anil B Murthy. Position: LLMs can't plan, but can help planning in LLM-modulo frameworks. In *Forty-first International Conference on Machine Learning*, 2024.

[28] Zhiyuan Li, Hong Liu, Denny Zhou, and Tengyu Ma. Chain of thought empowers transformers to solve inherently serial problems. In *The Twelfth International Conference on Learning Representations*, 2024.

[29] Aaron Lou, Chenlin Meng, and Stefano Ermon. Discrete diffusion modeling by estimating the ratios of the data distribution. In *International Conference on Machine Learning*, pages 32819–32848. PMLR, 2024.

[30] Yilun Xu, Shengjia Zhao, Jiaming Song, Russell Stewart, and Stefano Ermon. A theory of usable information under computational constraints. In *International Conference on Learning Representations*, 2020.

[31] Yoeng-Jin Chu. On the shortest arborescence of a directed graph. *Scientia Sinica*, 14:1396–1400, 1965.

[32] Richard S Sutton. Reinforcement learning: An introduction. *A Bradford Book*, 2018.

[33] Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. Reinforcement learning with deep energy-based policies. In *International conference on machine learning*, pages 1352–1361. PMLR, 2017.

[34] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[35] Nayoung Lee, Kartik Sreenivasan, Jason D. Lee, Kangwook Lee, and Dimitris Papailiopoulos. Teaching arithmetic to small transformers. In *The Twelfth International Conference on Learning Representations*, 2024.

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

[37] Daniel Zhang-Li, Nianyi Lin, Jifan Yu, Zheyuan Zhang, Zijun Yao, Xiaokang Zhang, Lei Hou, Jing Zhang, and Juanzi Li. Reverse that number! decoding order matters in arithmetic learning. *arXiv preprint arXiv:2403.05845*, 2024.

[38] Sean McLeish, Arpit Bansal, Alex Stein, Neel Jain, John Kirchenbauer, Brian Bartoldson, Bhavya Kailkhura, Abhinav Bhatele, Jonas Geiping, Avi Schwarzschild, et al. Transformers can do arithmetic with the right embeddings. *Advances in Neural Information Processing Systems*, 37:108012–108041, 2024.

[39] Shengyi Huang and Santiago Ontañón. A closer look at invalid action masking in policy gradient algorithms. *arXiv preprint arXiv:2006.14171*, 2020.

[40] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017.

## NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We support our main claims with extensive analysis in Sec. 4 and experiments in Sec. 5.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss limitations in Sec. 6.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best

judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

   Justification: We do not include theorems.

   Guidelines:

   - The answer NA means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We describe our method and experiment setup in Sec. 4 and 5.1 while additional details (e.g. hyperparameters) are provided in the supplementary material.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We will release our code and data upon acceptance.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: We provide experimental details in the appendix.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We report standard deviations for experiments we ran in the tables in the main text.

   Guidelines:

   - The answer NA means that the paper does not include experiments.

- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We provide relevant details in the appendix.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We confirm compliance with the Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We discuss potential societal impacts in Sec. A.

    Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: We do not produce any high-risk assets.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We cite the original papers in the main text for the code and data we used with details in the supplementary material.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.

- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: We provide details in the supplementary material for the synthetic datasets we constructed.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: We do not perform crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: We do not perform crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [NA]

    Justification: We do not use LLMs for important, original, or non-standard components of the core methods in this research.

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.

# A Broader Impacts

ReCOR aims to enhance the adaptive reasoning and planning capabilities of machine learning models, with potential applications in many fields, e.g. code writing, assisting scientific discoveries, and general LLM agents etc. Embodied agents with real-world tasks also require these capabilities. We regard ReCOR as mostly foundational research while noting that advancements in such capabilities have both positive and negative potential consequences if subject to misuse.

# B Pseudocode of Training and Inference Algorithms for ReCOR

**Algorithm 1** Training of ReCOR.

**Require:** Dataset $D$
**Ensure:** Trained parameters $\theta, \psi$
  Initialize $\theta, \psi$
  **while** Not converged **do**
    Sample mini-batch $B = \{(\mathbf{x}, \mathbf{y})\} \sim D$
    **for** $t = 1 \dots M$ **do** ▷ Rollout $\pi_\theta$ on $B$
      Sample $\rho_t, \bar{\boldsymbol{\rho}}_{t,[K]} \sim \pi_\theta(\cdot \mid \mathbf{x}, \mathbf{y}_{\boldsymbol{\rho}_{<t}})$
    **end for**
    Compute rewards $\mathbf{r}$ using $\psi$ on $B, \boldsymbol{\rho}, \bar{\boldsymbol{\rho}}$
    Update $\psi$ with $\mathcal{L}_{\text{LM}}$ and $\theta$ with $\mathcal{L}_{\text{SQL}}$ on $B, \boldsymbol{\rho}, \bar{\boldsymbol{\rho}}, \mathbf{r}$
  **end while**

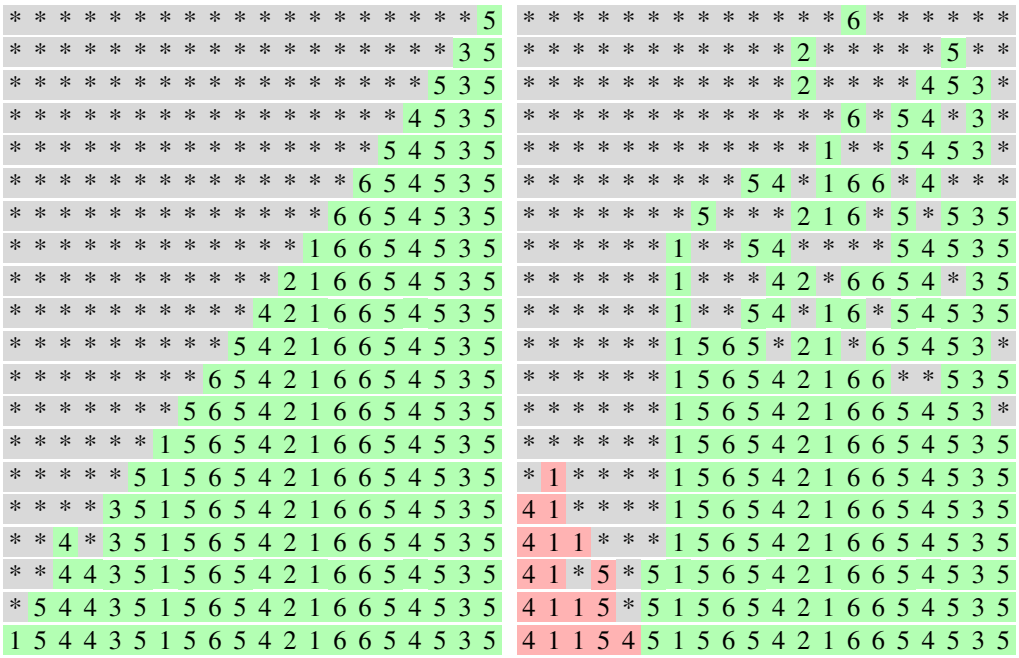
**Algorithm 2** Inference of ReCOR.

**Require:** Evaluation prompt $\mathbf{x}$, parameters $\theta^*, \psi^*$
**Ensure:** Generated response $\hat{\mathbf{y}}$
  **for** $t = 1 \dots M$ **do**
    Sample $\rho_t \sim \pi_{\theta^*}(\cdot \mid \mathbf{x}, \hat{\mathbf{y}}_{\boldsymbol{\rho}_{<t}})$
    Sample $\hat{y}_{\rho_t} \sim p_{\psi^*}(\cdot \mid \mathbf{x}, \hat{\mathbf{y}}_{\boldsymbol{\rho}_{<t}}, \rho_t)$
  **end for**

# C Generation Examples

Here we show generation examples of ReCOR and AdaMDM on Autoregression. The prompt is `64610246434563440135` while the correct response is `15443515654216654535`. Grey * denotes unfilled positions, green denotes correct tokens, and red ones are incorrect tokens.

As observed below, ReCOR (left) recovers the reversed generation order and generates the correct tokens, while AdaMDM (right) encounters difficulties and incorrectly generates later tokens. Note that even though AdaMDM uses adaptive inference techniques, it is confused about which token can actually be predicted correctly, and hallucinates wrong tokens. We analyze and explain this phenomenon in the experiment section in the main text.



# D Experiment Details

## D.1 Evaluation Datasets

Here we describe details about the datasets used in our experiments. For all datasets, we randomly split a small validation set of size 448 from the training set for validation purposes.

### D.1.1 Synthetic Autoregression

We generate a synthetic autoregression dataset (ARG) via the following procedure: For length $L$ and prime modulus $p$, we first generate the prompt $x_i \sim \mathcal{U}_{[p]}, 1 \leq i \leq L$, the generate response $\mathbf{y}$ satisfying the following autoregression formula:

$$y_L = x_L \tag{16}$$

$$y_i = \left( \prod_{j>i} [(y_j + x_i)\%(p-1) + 1] \right) \%p, 1 \leq i < L \tag{17}$$

This results in every $y_i$ depending on all $y_j, j > i$. We use $L = 20, p = 7$ for our experiments, resulting in $N = M = 20$. The size of the training set is $10^6$ while the size of the test set is $10^3$.

### D.1.2 Multiplication

We generate a multiplication dataset with prompts of the form `x*y=` and responses `z`, where $x \in [10^{19}, 10^{20})$ is a 20-digit positive integer, $y \in [2, 100)$ is a positive integer, and $z = x * y$. This leads to $N = 24, M = 22$. Both $x$ and $y$ are uniformly at random within the respective range. We generate a training set of size $10^5$ and a test set of size $10^3$.

### D.1.3 Sudoku

We use the Sudoku dataset from [10]. The prompt is a string of length 81 that contains the flattened initial configuration, where each element is either in $[1, 9]$ denoting a given cell, or 0 indicating that the value in this cell is missing. The response is also a string of length 81 that contains the full solution. $N = M = 81$. The training set contains approximately $1.8 * 10^6$ samples while the test set contains $10^5$ instances.

#### D.1.4 Zebra

We also use the Zebra dataset from [10]. The prompt for Zebra puzzles consists of a set of clues; we tokenize the special words in the clues with corresponding special tokens. After tokenization, $N = 455, M = 42$. The training set contains about $1.5 * 10^6$ puzzles while the test set contains $10^5$.

### D.2 Implementation Details

#### D.2.1 Details of ReCOR

The core algorithm of ReCOR is described in the method section of the main text. We document additional details below.

The ReCOR-related hyperparameters are listed in Tab. 6. For the token query stream and (optional) order query stream, we reuse all parameters of the main stream (including attention projection layers and MLP layers) to ensure that ReCOR always has the same or a smaller parameter count than baselines. During training, for the sampling of $\boldsymbol{\rho}$, we always greedily sample the position with minimal hardness as predicted by $Q_\theta$, while the token queries $\bar{\boldsymbol{\rho}}$ are sampled using `Exploration mode` in Tab. 6 without duplications. `Uniform` means uniform sampling $\bar{\boldsymbol{\rho}} \sim \mathcal{U}_{\mathcal{A}}$, while $\alpha = \ldots$ means using the corresponding entropy coefficient for exploration. For inference, we always greedily sample both $\rho_t$ and $y_{\rho_t}$. We use action masking [39] and never pick actions at already filled positions. We employ $\gamma = 0$ across the board, in effect solving a contextual bandit problem. Since our evaluation datasets require an exact match of all response tokens, the correctness of every token counts, and we do not need the trade-off between short-term and long-term rewards introduced by $\gamma$. $\gamma = 0$ simplifies the learning procedure by eliminating additional value function evaluations while still achieving superior performance, as evidenced in our experiments. Furthermore, we use the thresholded, sparse reward since this choice slightly improves performance (see additional ablations below), which we also attribute to the exact-match evaluation metric. Note that the binary cross-entropy loss associated with this reward actually computes an upper bound of the true Q function via Jensen's inequality. However, since we use $\gamma = 0$, the expectation for the Q value concentrates at a single point, rendering the bound tight.

Optionally, we use focal-like techniques [40] with focal $\gamma$ listed in Tab. 6 as `Focal coefficient`. We use focal loss for the RL value loss $\mathcal{L}_{\text{SQL}}$ instead of the token (language modeling) loss $\mathcal{L}_{\text{LM}}$, differing from similar techniques used in [11]. We remark that, as argued in the main text, certain token prediction sub-problems are inherently hard, and we should not expect the model to *solve* them or penalize it for these failures; rather, we simply want the model to *recognize* that these sub-problems are bad through the value loss.

#### D.2.2 Hyperparameters

We list hyperparameters on Autoregression and Multiplication in Tab. 5 while describing ReCOR-related ones on all datasets in Tab. 6. Baseline performances for Sudoku and Zebra are reported by [12]. In Autoregression and Multiplication, compared with ReCOR, we double the batch size and number of epochs for baselines to match the amount of compute per iteration and number of gradient steps of ReCOR to ensure a fair comparison.

#### D.2.3 Other Training Details

For all experiments, we use mixed-precision training with bfloat16. The model parameters are kept in full precision (float32). On Autoregression, Multiplication, and Sudoku, we use the tiny version of GPT-2 with 3 layers, 384 hidden dimensions, and 12 attention heads, resulting in approximately 6M parameters. On Zebra, we use the nano version with double the depth and parameter count. Each experiment takes a couple of hours using a single NVIDIA RTX4090 on Autoregression and Multiplication, and less than 2 days using a single NVIDIA A100-80G on Sudoku and Zebra.

Table 5: Hyperparameters for ReCOR and baselines on Autoregression and Multiplication.

| Hyperparameter | ReCOR | CLM | AR-GT | (Ada)MDM |
|---|---|---|---|---|
| # Model parameters | 6M | 6M | 6M | 6M |
| Batch size | 1024 | 2048 | 2048 | 2048 |
| # Epochs | 100 | 200 | 200 | 200 |
| Optimizer | AdamW | AdamW | AdamW | AdamW |
| Learning rate | $10^{-3}$ | $10^{-3}$ | $10^{-3}$ | $10^{-3}$ |
| LR scheduler | Cosine | Cosine | Cosine | Cosine |
| Weight decay | 0.1 | 0.1 | 0.1 | 0.1 |
| Diffusion steps | N/A | N/A | N/A | 20 |

Table 6: ReCOR-related hyperparameters on our evaluation datasets.

| Hyperparameter | ARG | MUL | Sudoku | Zebra |
|---|---|---|---|---|
| # Model parameters | 6M | 6M | 6M | 11M |
| Batch size | 1024 | 1024 | 512 | 512 |
| # Epochs | 100 | 100 | 40 | 50 |
| Optimizer | AdamW | AdamW | AdamW | AdamW |
| Learning rate | $10^{-3}$ | $10^{-3}$ | $3 * 10^{-4}$ | $3 * 10^{-4}$ |
| LR scheduler | Cosine | Cosine | Cosine | Cosine |
| Weight decay | 0.1 | 0.1 | 0.1 | 0.1 |
| Discount factor $\gamma$ | 0.0 | 0.0 | 0.0 | 0.0 |
| Focal coefficient | 2.0 | 2.0 | 0.0 | 0.0 |
| Threshold $\eta$ | 0.8 | 0.8 | 0.8 | 0.8 |
| # Token queries $K$ | 1 | 1 | 8 | 2 |
| # Order queries $C$ | 0 | 0 | 8 | 2 |
| Exploration mode | Uniform | Uniform | $\alpha = 0.1$ | $\alpha = 0.1$ |

# E Additional Experiments

## E.1 Exploration Strategies

We present additional ablation experiments regarding alternative exploration strategies for token queries $\bar{\rho}$ on Sudoku in Fig. 4a. `alpha=...` corresponds to an entropy-regularized exploration with the designated entropy coefficient, `Uniform` denotes uniform sampling, and `TopK` denotes sampling the $K$ positions with the largest Q-values. We find ReCOR pretty robust to alternative exploration strategies and opt for $\alpha = 0.1$ in our main experiments.

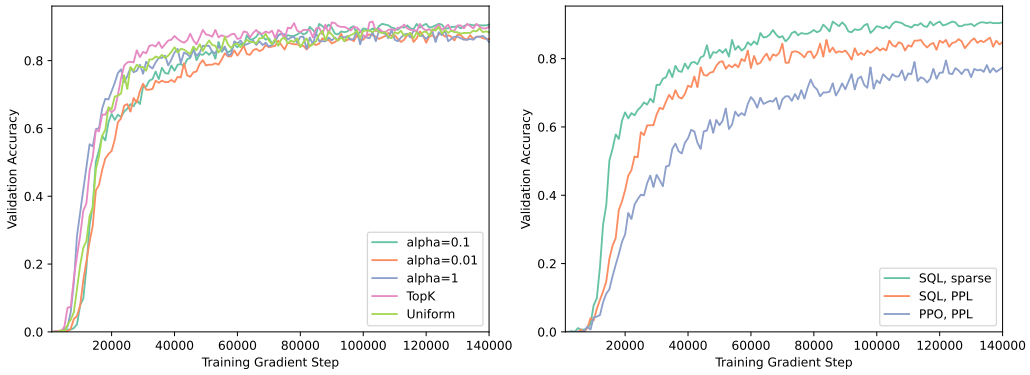
## E.2 Reward Functions and RL algorithms

Here we study the effects of alternative reward functions and RL algorithms on Sudoku. We compare our primary setting (Soft Q-learning, sparse reward with threshold $\eta = 0.8$) to alternative settings with negated perplexity as rewards and policy-gradient class of RL algorithms. Note that the sparse reward cannot be used with policy-gradient algorithms. We use the following loss to implement a PPO [34]-like algorithm, replacing $\mathcal{L}_{\text{SQL}}$:

$$\mathcal{L}_{\text{PPO}}(\theta) = \mathbb{E}_{s,a,r,s'}\left[\frac{\pi_\theta(a \mid s)}{\pi_{\theta_{\text{old}}}(a \mid s)}\hat{A} - \alpha H(\pi_\theta(\cdot \mid s))\right] \tag{18}$$

where $\hat{A}$ is the batch-normalized advantage. Note that since ReCOR always stays exactly on-policy and does not reuse actions, $\theta_{\text{old}}$ is the detached version of the current $\theta$, and the ratio is always 1, so we do not need the clipping in standard PPO.

As shown in Fig. 4b, our value-based choice of RL algorithm with a sparse reward achieves the best overall performance. Switching to dense rewards still yields decent performance, validating our motivation. We found a value-based algorithm to slightly outperform policy-gradient methods,




(a) Exploration strategy ablation with sparse rewards. (b) Reward function and RL algorithm ablation.

Figure 4: Additional ablation experiments regarding the exploration strategy of $\bar{\boldsymbol{\rho}}$ (a) and the choice of reward function and RL algorithm (b).

Table 7: Performance of ReCOR and baselines on the mixed dataset between ARG and MUL. ReCOR can perform uniformly well when trained on both datasets simultaneously.

| ReCOR | CLM | MDM | AdaMDM |
|---|---|---|---|
| **0.945** | 0.265 | 0.398 | 0.558 |

potentially due to the fact that it can make full use of return signals while policy-based methods only model the relative difference between actions.

### E.3 Handling Multiple Tasks

To validate the generalization capability of ReCOR across different types of tasks, we conduct a mixed-dataset experiment that trains ReCOR and baselines on ARG and MUL simultaneously. The average evaluation accuracies on both test sets are shown in Tab. 7. We can see that ReCOR learns from both datasets and achieves uniformly good performance, while baselines still struggle.