PROPHECY TRANSFORMER

Anonymous authors
Paper under double-blind review

ABSTRACT

Autoregressive transformers trained on next-token prediction suffer from a key weakness; early decisions can restrict the model to suboptimal continuations. Furthermore, they do not learn a meta-model of how their mistakes correlate with ground truth, nor how to recover. These issues are critical in the context of reasoning, where such approaches often become trapped in irreconcilable dead-ends without a way to update their predictions based on these failures. To address this, we introduce Prophecy Transformer, an architecture which conditions its generations on a nonparametric latent variable encoding the model's internal activations produced during an initial rollout following its own policy. By attending to this latent variable from the start of generation, our model gets a view into its possible future internal state which reveals information that would only become apparent *after* committing to certain decisions. Tested on algorithmic reasoning tasks such as 3-SAT, graph coloring, TSP, and countdown, our method significantly outperforms autoregressive models due to its prospective and self-corrective capabilities. Across tasks, Prophecy Transformer acquires self-referential representations encoding the quality of its predictions and the ability to update the model's decisions based on its predicted future state, unlocking large gains in reasoning ability.

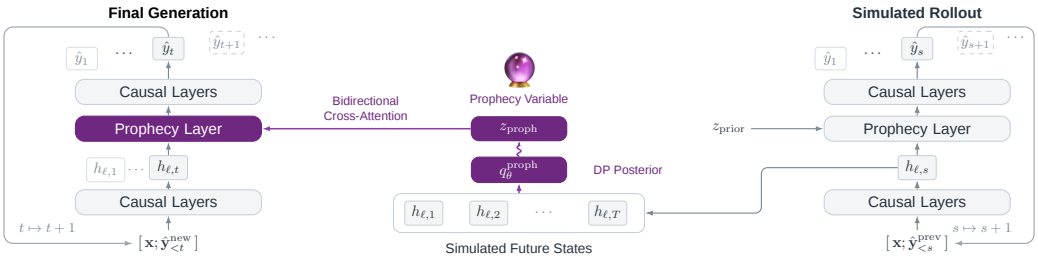

Figure 1: **Prophecy Transformer architecture.** Following a simulated rollout, the resulting hidden states are projected into the parameters of a posterior Dirichlet Process from which a mixture distribution that we call *prophecy variable* is sampled. Attending to this mixture distribution allows the final generation to condition on its possible future beliefs produced during simulated attempts.

1 INTRODUCTION

The ability to roll out candidate actions and alter one's policy in retrospect is a hallmark of intelligent behavior. In the context of machine learning, it is one of the main principles underlying reinforcement learning and the success of reasoning LLMs (Sutton & Barto, 2018; Guo et al., 2025). In the study of algorithms more broadly, this idea is fundamental to search and backtracking (Biere et al., 2009). The underlying principle is that given some initial set of decisions, it is possible to make more optimal choices by observing the consequences of those decisions, particularly when they are hard to predict due to the combinatorics of the search space.

Traditional autoregressive transformer architectures trained on next-token prediction suffer from short-sightedness due to the fact that they need to commit to choices early (Bachmann & Nagarajan, 2024; Shih et al., 2023); furthermore, these choices cannot be amended when they are found undesirable later on. Autoregressive models often find themselves trapped in irreconcilable branches inside solution spaces, leading to *regretful compromise*, a situation where no continuation is able to

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

satisfy an already predicted prefix and generation capitulates to an unsound answer (Ye et al., 2025). Such conflicts can often be detected retrospectively via metrics associated to the generative model, e.g., by measuring uncertainty in the token predictions (Kang et al., 2025). In principle, the internal states of the model which produced these predictions can be inspected to retrieve information about the model's beliefs (such as uncertainty) and be used to generate a revised answer. However, to the best of our knowledge, existing work has not yet demonstrated a method which applies this idea to a general training method for transformer architectures that achieves improved performance on reasoning tasks.

To address these issues, we introduce a novel architecture which conditions its generations on a nonparametric latent variable encoding possible future states produced by its own predicted rollouts. Concretely, we train models to perform next-token prediction using two attention functions, standard causal attention, which accesses the memory of its prefix, and cross attention to such memories produced during a previous complete attempt by the same model. This allows models to condition predictions at early time steps on a sequence of predictions at future time steps, and learn correlations between the ground truth and their own internal states produced during potentially faulty solution attempts. Inspired by *prophecy variables* used in program verification (Abadi & Lamport, 1991; Jung et al., 2019), we call our architecture **Prophecy Transformer**.

Training such self-referential models suffers from pervasive superficial correlations between the two rollouts, which we address by adding a nonparametric variational information bottleneck to the attention over the initial rollout. This regularizer encourages the model to rely on autoregressive generation, and only use the representation of the simulated future when it is needed to disambiguate possible continuations. We find that giving autoregressive transformers access to this simulated future information drastically improves their performance on algorithmic reasoning tasks such as 3-SAT, graph coloring, traveling salesman problem, and countdown compared to baselines.

We find that exposing models to the distribution of their inference-time internal states in this way leads to emergent self-corrective behavior. Furthermore, our analysis shows that our models map solution attempts to latent representations encoding correctness, and infer representations at the start of generation which encode self-referential features about its own eventual success. A key property of our method is that such corrections need not be local in nature; our models exhibit corrective behavior that leads to global pivoting of solution strategies that would not arise from spot-fixing previous attempts.

In summary, our contributions are:

- We introduce Prophecy Transformer, a novel architecture which conditions its generations on a nonparametric latent variable encoding the model's own internal activations produced during simulated rollouts.
- We demonstrate significantly improved performance on 3-SAT, graph coloring, traveling salesman problem, and countdown compared to standard autoregressive baselines, including inference-time search methods.
- We show that our nonparametric variational treatment of Prophecy Transformer's latent variable leads to abstract representations that encode correctness of previous generations, and estimate the success of their final generations in advance.

2 Related Work

Two-pass neural networks. Previous work in neural machine translation and text summarization has incorporated two-pass neural networks known as Deliberation Networks (DNs); here, two separate decoders are trained, and the second decoder conditions on the predictions and hidden states of the first (Xia et al., 2017; Hassan et al., 2018). Look-ahead attention (Du et al., 2023) similarly trains a transformer decoder to attend to representations encoded by a separate model. While the principle behind conditioning predictions on an initial generation is similar, our method differs in that it never trains a separate decoder and instead conditions its predictions on its *own* internal states produced during a simulated rollout. This avoids the issue of joint/independent training of DNs (Dou & Gales, 2022), and more importantly, exposes our model to representations encoding distributional information about itself (discussed in §4.1).

Sequential methods for autoregressive reasoning. LLMs exhibit the capacity to reason through difficult problems by relying on extensive CoT (Wei et al., 2022), and these abilities have been further enhanced via post-training reinforcement learning with verifiable rewards (RLVR) (Guo et al., 2025). A plethora of RL methods for LLM post-training have emerged in recent years (Shao et al., 2024; Ahmadian et al., 2024; Yu et al., 2026; Kazemnejad et al., 2025), and these methods overcome the shortcomings of autoregressive reasoning by expending a large amount of tokens to sequentially attempt varying approaches until a solution is found. However, recent work has shown that models trained in this way suffer from overthinking and getting stuck in endless loops (Chen et al., 2024; Pipis et al., 2026). In contrast to RLVR, our method does not require a reward function beyond the standard next-token prediction objective, and is able to learn how to attempt solutions and recover from mistakes in a self-supervised manner in domains where only final answer artifacts exist without associated CoT labels (discussed in §5.3).

Iterative search methods for optimal inference. Iterative search methods which seek to find optimal sequences by repeatedly sampling possible continuations through expansion and backtracking include Tree-of-Thought (Yao et al., 2023), self-consistency (Wang et al., 2023), guided decoding (Xie et al., 2023), and Monte-Carlo Tree Search (Hao et al., 2023). While these methods are often able to recover close to optimal sequences through extensive enumeration, they incur large computational cost which can become intractable as the sequence length grows (Snell et al., 2025; Wu et al., 2025).

Inference methods for improved planning and foresight. Previous work has proposed inference methods that address the short-sightedness of autoregressive sequence models while mitigating the large cost of iterative enumeration. Such methods include Predictive Decoding (Ma et al., 2025) and ϕ-Decoding (Xu et al., 2025), which make use of *foresight sampling*, a technique where decisions in the present are based on the evaluation of simulated rollouts into the future.

While the motivating principle behind foresight sampling is similar to our method, previous methods differ in three important ways: 1) they rely on a separate handcrafted step value estimation function, while our method requires no such function 2) their step value estimation is chosen a priori, while our method is able to learn correlations between simulated rollouts and optimal action sequences from data 3) they necessitate repeated simulated rollouts to compute relative advantages between possible futures, while our method only requires one.

3 Background

3.1 Autoregressive Modeling

The standard approach to modeling sequences is to use the product rule to factorize the joint distribution $P(\mathbf{y}) = P(y_1, y_2, ..., y_t)$ into a product of conditionals parameterized by θ:

$$P_\theta(\mathbf{y}) = P_\theta(y_1) \prod_{t=2}^{N} P_\theta(y_{t+1} \mid \mathbf{y}_{1:t})$$

The conditional distributions are learned by minimizing the negative log-likelihood:

$$\mathcal{L}(\theta) = -\mathbb{E} \log P_\theta(\mathbf{y})$$

3.2 Incorporating Latent Variables

It is often easier to model the relationship between observed outcomes and some latent variable. We can consider the joint distribution $P(\mathbf{y}, z) = P(y_1, y_2, ..., y_t, z)$ between sequences and a latent variable z. Then an analogous factorization yields:

$$P_\theta(\mathbf{y}, z) = P(z) P_\theta(y_1 \mid z) \prod_{t=2}^{N} P_\theta(y_{t+1} \mid \mathbf{y}_{1:t}, z)$$

We can approximate this distribution by minimizing variational free energy (a.k.a. the ELBO):

$$\mathcal{F}(\theta) = D_{KL}\left(q_\theta(z \mid \mathbf{y}) \parallel P(z)\right) - \mathbb{E}_{z \sim q_\theta}\left(P_\theta(\mathbf{y} \mid z)\right) \tag{1}$$

where $\{q_\theta\}$ is some family of functions used to approximate the posterior distribution over z. The KL-divergence between the posterior $q_\theta(z \mid \mathbf{y})$ and the prior $P(z)$ adds a variational information bottleneck (VIB) (Alemi et al., 2017) to the task loss $P_\theta(\mathbf{y} \mid z)$, which results in a compressed representation in the latent variable.

3.3 Nonparametric latent variables

Our method uses a family of nonparametric distributions known as Dirichlet Processes (DP) as its posterior over latent variables. As opposed to parametric distributions, e.g. Gaussian distributions, whose samples have some fixed number of components, nonparametric distributions emit samples with an arbitrary number of components.

Nonparamtetrics are desirable when the amount of information carried by the latent variable depends on the complexity of the input, or is unbounded; for example, when encoding the meaning of a sentence, we would like the capacity of the latent variable to grow as a function of the sentence length. Previous work has shown that nonparametric variational methods using mixture distributions sampled from DPs as their latent representations lead to improved performance on text summarization and reasoning tasks (Fehr & Henderson, 2024; Shirakami & Henderson, 2026).

Dirichlet Process details. DPs are parameterized by a base distribution G and concentration parameter α, denoted $z \sim \mathrm{DP}(G, \alpha)$. The sampled object z is a set of atoms drawn from the base distribution with associated probabilities that sum to 1; in other words, DPs are distributions over distributions. When the base distribution is a Gaussian, the sampled z are mixtures of impulse distributions in vector space. Henderson & Fehr (2023) show a theoretical equivalence between the attention operation and computing the posterior mean of such a mixture distribution (discussed in App. A.1), which allows us to apply this Bayesian nonparametric view to transformer architectures. Henderson & Fehr (2023) propose a nonparametric variational information bottleneck (NVIB) which compresses the representations learned by an attention layer.

4 Prophecy Transformer

This section describes our main methodological contributions, namely the introduction of prophecy variables and their integration into transformers. Our prophecy variables provide attention-based access to our model's possible future states produced during simulated rollouts. An NVIB regularizer helps prophecy variables learn abstract representations of possible futures which generalize well.

4.1 Prophecy variables

Consider drawing z in the following way:

$$z \sim q_\theta^{\text{proph}} \quad \text{where} \quad q_\theta^{\text{proph}} \coloneqq f_\theta(\hat{\mathbf{y}}), \quad \hat{\mathbf{y}} \sim P_\theta \tag{2}$$

We call this special latent variable a *prophecy variable* and denote it as z_{proph}. We can sample a new prediction $\hat{\mathbf{y}}^{\text{new}}$ as follows:

$$\hat{\mathbf{y}}^{\text{new}} \sim P_\theta(\cdot \mid z_{\text{proph}}) \quad \text{where} \quad z_{\text{proph}} \sim q_\theta^{\text{proph}}, \quad q_\theta^{\text{proph}} = f_\theta(\hat{\mathbf{y}}^{\text{prev}}), \quad \hat{\mathbf{y}}^{\text{prev}} \sim P_\theta \tag{3}$$

Notice that in the case of autogressive sequence modeling, a prediction $\hat{y}_{t+1}^{\text{new}}$ now depends on $\hat{\mathbf{y}}^{\text{prev}}$ in its entirety: $\hat{y}_{t+1}^{\text{new}} \sim P_\theta(\cdot \mid \hat{\mathbf{y}}_{1:t}^{\text{new}}, \hat{\mathbf{y}}^{\text{prev}})$. This means that unlike traditional autoregressive modeling, new predictions at position t depend on previous predictions at positions $t' > t$. A useful piece of intuition is to imagine the prophecy variable as a crude representation of what may happen at future time steps during autoregressive inference. By utilizing prophecy variables, inference no longer unfolds 'blindly' step by step; it can make use of future information to guide its generation.

Self-referential prophecy variables. Certain choices of $f_\theta(\hat{\mathbf{y}})$ allow z_{proph} to encode information about the distribution from which $\hat{\mathbf{y}}$ is sampled. For example, consider the case where f_θ simulates the autoregressive generation process used to sample $\hat{\mathbf{y}} \sim P_\theta$. Suppose that f_θ then outputs a distribution q_θ^{proph} parameterized by the set of reconstructed conditional distributions $\{P_\theta(\cdot|\hat{\mathbf{y}}_{1:t})\}_{1 \le t \le T}$. Then, $z_{\text{proph}} \sim q_\theta^{\text{proph}}$ encodes distributional information about P_θ that would not be available by observing $\hat{\mathbf{y}}$ on its own. We call prophecy variables encoding information about P_θ *self-referential*.

Using self-referential prophecy variables, the sampling process described in §4.1 allows P_θ to condition its generations on distributional information produced during simulated rollouts from P_θ (itself); for instance, P_θ could condition its distribution over y_t on its own future uncertainty over some sampled, on-policy trajectory $\hat{\mathbf{y}}_{>t}$. More generally, self-referential prophecy variables allow conditioning in the following way:

$$y_{t+1} \sim P_\theta\left(\cdot \mid \mathbf{y}_{1:t}, P_\theta(\cdot \mid \hat{\mathbf{y}}_1), P_\theta(\cdot \mid \hat{\mathbf{y}}_{1:2}), ..., P_\theta(\cdot \mid \hat{\mathbf{y}}_{1:T})\right) \quad \text{where} \quad \hat{\mathbf{y}} \sim P_\theta \tag{4}$$

Here, $\{P_\theta(\cdot \mid \hat{\mathbf{y}}_{1:i})\}_{1 \le i \le T}$ are random measures whose stochasticity arises from sampling $\hat{\mathbf{y}}$.

4.2 Architecture

Fig. 1 shows how we incorporate prophecy variables into transformer architectures. Similarly to traditional autoregressive transformer sequence models, our method takes a prefix sequence as input and outputs a distribution over the next token. It differs in the fact that it also performs a parallel forward pass taking a previously generated sequence as input, where the resulting model activations produced at layer l are encoded into a nonparametric latent variable z_{proph} and accessed by the model via a bidirectional cross-attention mechanism at layer $l+1$. We follow Henderson & Fehr (2023) for their theory and implementation of accessing nonparametric latent variables via attention. More details are given in App. A.1.

More explicitly, using our notation from §4.1, what we call $f_\theta(\hat{\mathbf{y}})$ is implemented by the first l layers of a transformer followed by learned projections (details found in App. A.2), and this operation infers the posterior DP q_θ^{proph} from which the prophecy variable z_{proph} is sampled. This means that the specific activation sequence that is produced at layer l during the generation of $\hat{\mathbf{y}}$ is encoded into the prophecy variable which the model conditions on when generating its new sequence. As a consequence, z_{proph} in Prophecy Transformer can be considered self-referential according to the definition given in §4.1.

4.3 Training

Our models are trained by minimizing the variational free energy introduced in §3.2. We jointly optimize the negative log-likelihood of the data conditioned on a sampled prophecy variable and the KL-divergence between the prior DP and the inferred posterior DP. Thus the loss is:

$$\mathcal{L}(\theta) = D_{KL}\left(q_\theta(z \mid \hat{\mathbf{y}}^{\text{prev}}) \parallel P(z)\right) - \log P_\theta(\mathbf{y} \mid z_{\text{proph}}) \quad \text{where} \quad z_{\text{proph}} \sim q_\theta(z \mid \hat{\mathbf{y}}^{\text{prev}}) \tag{5}$$

Here $\mathbf{y}$ is the ground truth sequence, and $\hat{\mathbf{y}}^{\text{prev}}$ is a prior generation from $P_\theta(\cdot \mid \mathbf{y}_{1:t}, z_{\text{prior}})$ where $\mathbf{y}_{1:t}$ is some optional ground truth prefix and z_{prior} is sampled from the prior DP. Our method uses a standard next-token prediction objective with cross-entropy loss to compute the negative log-likelihood. To compute the KL-divergence between the prior and posterior DPs, we follow the NVIB of Henderson & Fehr (2023).

5 Algorithmic Reasoning

In this section, we test Prophecy Transformer's ability to solve constraint satisfaction and combinatorial optimization problems. We choose tasks which are combinatorial in nature and have the property that local coherence provides limited information about progress towards the overall objective. Problems of this nature allow us to study our model's ability to infer correct approaches from an exponentially large space of locally plausible continuations; an ability we hypothesize will be improved through the use of prophecy variables. We first describe the synthetic tasks considered in our experiments. Further details on task implementations are discussed in App. A.3.

5.1 Overview of Tasks

3-SAT. The Boolean satisfiability problem (also known as SAT) is a fundamental decision problem in computer science and was the first to be shown NP-complete (Cook, 1971). While the general form of the problem asks if an arbitrary formula in propositional logic can be made true by some assignment of logical values to its variables, 3-SAT restricts the space of formulas to those in conjunctive normal form where each clause contains a disjunction of exactly 3 variables. 3-SAT is

also an NP-complete decision problem, and has the desired property that the validity of a partial assignment does not guarantee its containment in a satisfying assignment.

Setup. In our experiments, we consider a witness generation variant of the problem, where instead of asking whether a given formula is satisfiable, the objective is to generate a sequence of true/false variable assignments that satisfies the formula. We produce our training and test sets by randomly sampling formulas with 62 clauses containing 3 literals each over 12 variables, rejecting unsatisfiable formulas. We choose the number of clauses following the cross-over point formula (Crawford & Auton, 1996) used in previous 3-SAT benchmarks (Ye et al., 2025) to ensure that we generate relatively difficult problems. We train on 500k training examples and evaluate on 10k test examples.

Graph coloring. A graph coloring is an assignment of colors to vertices of a graph such that no two neighboring vertices share the same color assignment. A k-coloring is such an assignment using exactly k colors. The problem of whether an arbitrary graph admits a k-coloring for a given k is NP-complete for $k \geq 3$.

Setup. In this task, we generate 4-colorable graphs and ask for a valid coloring using exactly 4 colors. In other words, the objective of the task is to generate a witness to the 4-colorability of a given graph. We consider graphs containing 16 vertices and 36 edges. Graphs are generated by first assigning 4 vertices to each color, and constructing bipartite graphs with 6 edges between every color-class pair and taking their union. As such, the graphs we generate are always 4-colorable, fixed density, and have balanced colorings. We generate 500k training examples and evaluate on 10k test examples.

Traveling salesman problem. The traveling salesman problem is a combinatorial optimization problem on weighted undirected graphs. Given such a graph, it asks for the lowest cost path which visits every vertex exactly once and returns to the starting vertex. The problem is NP-hard in its general form.

Setup. Similarly to our 3-SAT experiments, we consider a witness generation variant of the problem where the objective is to generate a path which realizes the optimal cost. Our generated training examples are weighted graphs with 16 vertices and integer weights sampled uniformly between [0-9]. To ensure feasibility, we plant a Hamiltonian cycle by taking a random permutation of vertices. The remaining edges are randomly sampled with probability 0.08. For each generated graph, we run an exact solver to retrieve the true optimal path and store it as the solution. We generate 500k training and 10k test examples.

Countdown. Countdown is a mathematical reasoning task where given a set of natural numbers and a target number, the objective is to apply elementary arithmetic operations $\{+, -, \times, \div\}$ to the given numbers and intermediate results to reach the target number. For example, given a set of numbers $\{5, 11, 3, 2\}$ and target 26, a valid solution is $5 + 11 = 16, 16 - 3 = 13, 13 \times 2 = 26$. Every number in the starting set must be consumed, and cannot be reused.

Setup. We generate problems with 4 starting integers with solutions requiring 3 arithmetic operations to reach the target. We sample starting numbers from [1, 99] without replacement and targets from [10, 100]. For a given target, we use rejection sampling to resample starting numbers until depth-first search finds a valid solution. We generate 500k training examples and 10k test examples with disjoint targets. For example, if 10 appears as a target in any problem in the test set, then no problem in the training set will have that target.

5.2 Experimental setup

Prophecy Transformer. For all our experiments, we train a Prophecy Transformer (PT) instance for 25k steps on the training set. We use greedy autoregressive decoding for inference. Generation is carried out by first generating a prior-conditioned solution (prior generation) and using it to encode a prophecy variable which is then used to generate the final answer. Training configuration details are given in App. A.4.

Baselines. We train a baseline autoregressive model (AR baseline) on the same training examples and evaluate its performance using 3 search-based inference methods which have been proposed to overcome the limitations of autoregressive generation. In addition to standard autoregressive inference, we evaluate using Tree-of-Thought (ToT)/beam search (Yao et al., 2023), Monte-Carlo

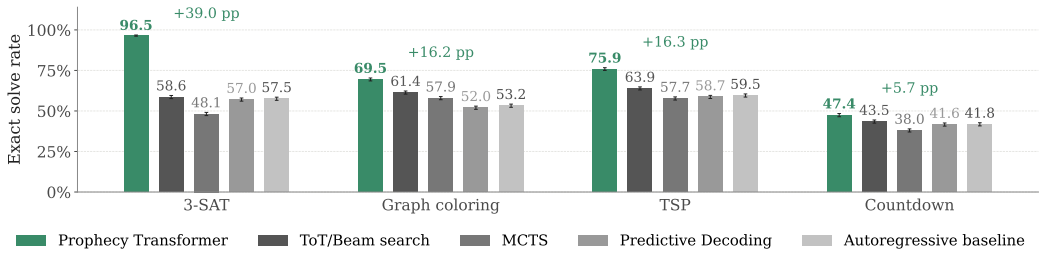

Figure 2: **Accuracy on algorithmic reasoning tasks.** We plot exact solve rates across 4 tasks for Prophecy Transformer and each of the baselines we tested. Error bars are 95% Wilson score intervals. Our results show that Prophecy Transformer outperforms all of the baselines we evaluated across all tasks.

Tree Search (MCTS) (Hao et al., 2023), and Predictive Decoding (Ma et al., 2025). In our synthetic task settings, ToT, MCTS and Predictive Decoding treat each proposed solution token as a reasoning step and use the model likelihood as the score for partial solutions. Further baseline configuration details are given in App. A.4.

5.3 Results on algorithmic reasoning

Prophecy Transformer outperforms search baselines on algorithmic reasoning. Our results for algorithmic reasoning are shown in Fig. 2. We see that Prophecy Transformer significantly outperforms all of the baselines we tested across all tasks. The largest gain is in 3-SAT, where our method achieves 96.5% accuracy versus the AR baseline scoring 57.5%, a 39.0 percentage point improvement. We observe 16.2 and 16.3 percentage point improvements against AR baselines on graph coloring and TSP respectively. For countdown, our method outperforms the AR baseline by 5.7 percentage points.

Focusing on the additional inference methods we tested, we find that ToT/Beam search and MCTS both improve accuracy compared to the AR baseline on graph coloring. We also find that ToT/Beam search improves accuracy over the AR baseline on TSP. Across all tasks, Prophecy Transformer performs better than the best performing baseline inference method.

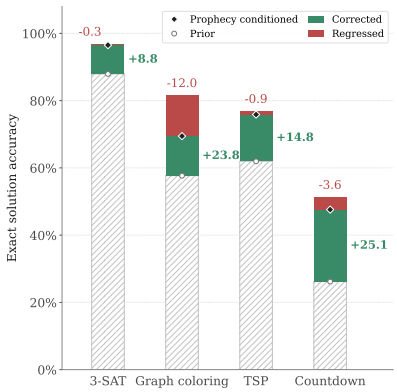

Figure 3: **Self-corrective behavior of Prophecy Transformer.** Comparison of solution accuracy between prior-conditioned and prophecy-variable-conditioned generations. Diagonal fill indicates the accuracy achieved by prior generations. Red fill indicates solutions that regressed during final generation, and green fill indicates net improvement.

Self-corrective behavior of Prophecy Transformer. Fig. 3 shows our results comparing the accuracy of Prophecy Transformer's prior generations to its final prophecy-variable-conditioned generations. Across all tasks, PT demonstrates the ability to condition on incorrect prior generations and produce corrected final generations. On 3-SAT, our method produces a net accuracy improvement of 8.8 percentage points. For graph coloring, TSP and countdown, we see a net improvement of 23.8, 14.8, and 25.1 percentage points respectively. Interestingly, while some correct prior generations are erroneously made incorrect during final generation, the number of successful corrections significantly outweigh such cases and we observe large net improvements across all tasks.

Visualizing and probing prophecy variables. Prophecy Transformer is trained on next-token prediction conditioned on a prophecy variable encoding its prior generation, and the weights responsible for this encoding are learned jointly with the rest of the model to minimize the overall loss. We thus hypothesize that our training objective encourages our models to encode information in its internal states that capture information about the correctness of its previous attempts, in order to be able to correct them.

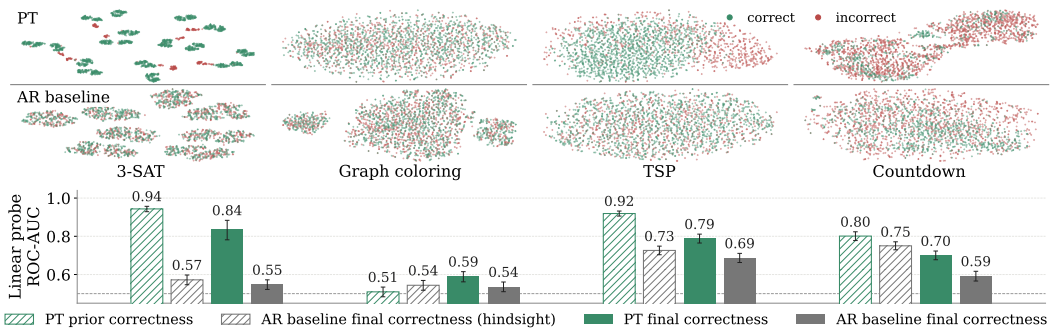

Figure 4: **Visualizing and probing prophecy variables.** **Above:** tSNE plots of Prophecy Transformer's pooled prophecy variable and baseline embeddings each encoding instances of its own generated solutions. Embeddings are colored green/red depending on ground truth correctness of the encoded solution. **Below:** ROC-AUC metrics for linear probes trained to predict the correctness of generated solutions in two settings: dashed green fill indicates probe metrics for discriminating prior generation correctness taking prophecy variable instances as input, solid green fill indicates probe performance on final generation correctness taking the conditioned prompt embedding as input. Dashed grey and solid fill indicate corresponding experiments for AR baseline.

To investigate this hypothesis, we first visualize the latent space learned by Prophecy Transformer (upper half of Fig. 4). The visualizations corresponding to PT are tSNE plots of the weighted average over its nonparametric latent variable's components, where the latent variable encodes the model's prior-conditioned solution attempts on each task. The plotted embeddings are colored by the ground truth correctness of the embedded solutions. For the AR baseline, we take the simple average over hidden representations at the corresponding layer, and plot them analogously. These results show visible separation between the PT embeddings of correct and incorrect solutions for 3-SAT, graph coloring and countdown, with less if any separation for AR embeddings.

Next, in order to quantify the presence of correctness features, we train linear probes on these embeddings for each task (for both PT and baseline), where the objective is to classify the correctness of the generation used to compute the embedding. The ROC-AUC metrics of the resulting classifiers are plotted in the bottom half of Fig. 4 (indicated by bars filled with dashed lines). This analysis shows that, for 3-SAT, TSP, and countdown, PT's prophecy variable encodes more linearly separable information about the correctness of the encoded prior generation than the AR baseline. We note that the metric reported for the AR baseline is labeled *hindsight* since it is computed from the sequence of hidden representations produced after generating the entire final answer.

Lastly, we train linear probes to classify the correctness of *final* generations resulting from prefix embeddings extracted from PT and AR baseline, respectively. Linear probe metrics are presented in the bottom half of Fig. 4 indicated by solid green for PT and solid grey for AR baseline. We find that PT's prefix embeddings predict its final generation correctness above random chance across tasks. Furthermore, PT's prefix embeddings lead to significantly better correctness prediction compared to AR baseline on all tasks, indicating that features relevant to introspection are more robustly represented in our model.

Taken together, these results suggest the presence of self-referential modeling abilities unique to Prophecy Transformer.

5.4 Further Ablations

Investigating the role of variational nonparametrics. Central to Prophecy Transformer is its nonparametric posterior and variational Bayesian training objective (ELBO) used to encourage compressed latent representations embedded in a smooth and disentangled latent space. A natural ablation on this idea is to train a model identical to PT in all aspects but with its probablistic treatment of its latent variable removed. This would entail directly attending to the raw hidden states produced during the first-pass generation, without any sort of compression or sampling induced by our variational components.

Setup. We train our direct attention ablation on 3-SAT using the same dataset and training configuration as before. The direct attention model has the same architecture as PT except instead of projecting the first pass hidden states into parameters of a DP, the hidden states are directly accessed via bi-directional cross-attention during the second pass.

In addition, similarly to our previous experiments, we train linear probes on the pooled embeddings resulting from the direct attention model's first-pass solution attempts, and test their ability to predict ground truth correctness of these solutions.

Results. Our results are presented in Fig. 5. We see that on 3-SAT, the direct attention ablation performs significantly worse than PT. Furthermore, our linear probe experiments show that its hidden states contain less recoverable information about solution correctness compared to prophecy variables.

Comparisons with Deliberation Networks. Another key property of Prophecy Transformer is that its first and second-pass generations are produced by the same model. This is not the case for previous methods such as Deliberation Networks (Xia et al., 2017) and Look-ahead attention (Du et al., 2023), which train a separate decoder for each pass.

We hypothesize that our parameter sharing approach plays an important role in our models learning their introspective representations discussed in §5.3. Because the parameters encoding the first-pass hidden states are the same parameters used to make predictions in the second pass, these representations are in the same space and each have a dual use; one is to predict the next token, and the other is to provide information about the correctness of the current solution approach. We believe that this encourages representations in the prophecy-variable-conditioned pass to also model its own future correctness.

To investigate the relevance of parameter sharing, we train and evaluate a 'separate decoder' baseline designed to emulate previous approaches such as Deliberation Networks.

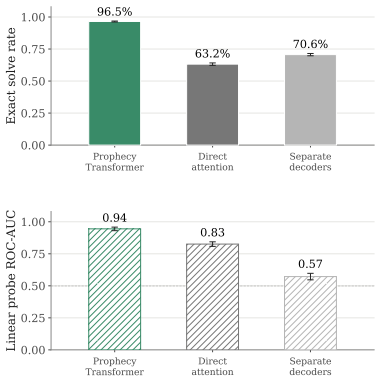

Figure 5: **Ablating variational components and parameter sharing. Above:** Comparison of solution accuracy between PT, direct attention and separate decoder ablations. **Below:** Linear probe performance on predicting ground truth first-pass generation correctness from pooled first-pass hidden states.

Setup. For our separate decoder baseline, we take a frozen baseline checkpoint and train another decoder to attend to its raw hidden states at the same layer as PT. We train on the same 3-SAT data using the same training configuration as previous experiments. After training our separate decoder baseline, we once again train linear probes to predict ground truth solution correctness taking its first-pass embeddings as input.

Results. Our results are shown in Fig. 5. We see that the separate decoder baseline significantly under-performs PT on 3-SAT. Looking at the linear probe performance, we see that its hidden states do not represent correctness as robustly as PT's prophecy variable.

6 Conclusion

In this paper, we introduced Prophecy Transformer, an architecture for autoregressive sequence modeling that conditions its generations on a nonparametric latent variable encoding its own internal states produced during simulated generation attempts. We find that our method leads to significant improvements on algorithmic reasoning tasks with combinatorial search spaces such as 3-SAT, graph coloring, traveling salesman problem and countdown. In addition, we observe that Prophecy Transformer learns how to correct its own errors, and infers representations that encode information about its own correctness on reasoning problems. Taken together, these results demonstrate the emergence of prospective and self-referential modeling abilities in Prophecy Transformer, and indicates a promising novel approach to autoregressive sequence modeling.

AI USE STATEMENT

In this work, we used generative AI tools for code and LaTeXfigure syntax. We have not used generative AI tools for any of the writing in the main text of the paper. Additionally, we used generative AI tools as a search engine for literature review, where relevant papers were manually investigated and cited appropriately. We have reviewed all AI-assisted work. LLM-generated code was verified and tested for correctness. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

The work presented here contributes to foundational AI research and therefore does not raise any immediate or potential ethical concerns.

REPRODUCIBILITY STATEMENT

In order to make our work reproducible, we have included experimental details for all of our experiments in the main text and supplementary materials. See App. A.4 for more information. We provide model and training code in our supplementary materials.

REFERENCES

Martín Abadi and Leslie Lamport. The existence of refinement mappings. *Theoretical Computer Science*, 82(2):253–284, 1991. ISSN 0304-3975. doi: https://doi.org/10.1016/0304-3975(91)90224-P. URL https://www.sciencedirect.com/science/article/pii/030439759190224P.

Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. Back to basics: Revisiting reinforce-style optimization for learning from human feedback in llms. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12248–12267, 2024.

Alexander A. Alemi, Ian Fischer, Joshua V. Dillon, and Kevin Murphy. Deep variational information bottleneck. In *5th International Conference on Learning Representations, ICLR 2017, Conference Track Proceedings*, Toulon, France, 2017. OpenReview.net. URL https://openreview.net/forum?id=HyxQzBceg.

Gregor Bachmann and Vaishnavh Nagarajan. The pitfalls of next-token prediction. *arXiv preprint arXiv:2403.06963*, 2024.

Armin Biere, Hans {van Maaren}, T. Walsh, and M. Heule (eds.). *Handbook of Satisfiability*. IOS Press, February 2009.

Xingyu Chen, Jiahao Xu, Tian Liang, Zhiwei He, Jianhui Pang, Dian Yu, Linfeng Song, Qiuzhi Liu, Mengfei Zhou, Zhuosheng Zhang, et al. Do not think that much for 2+ 3=? on the overthinking of o1-like llms. *arXiv preprint arXiv:2412.21187*, 2024.

Stephen A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the Third Annual ACM Symposium on Theory of Computing*, STOC '71, pp. 151–158, New York, NY, USA, 1971. Association for Computing Machinery. ISBN 9781450374644. doi: 10.1145/800157.805047. URL https://doi.org/10.1145/800157.805047.

James M. Crawford and Larry D. Auton. Experimental results on the crossover point in random 3-sat. *Artificial Intelligence*, 81(1):31–57, 1996. ISSN 0004-3702. doi: https://doi.org/10.1016/0004-3702(95)00046-1. URL https://www.sciencedirect.com/science/article/pii/0004370295000461. Frontiers in Problem Solving: Phase Transitions and Complexity.

Qingyun Dou and Mark Gales. Deliberation networks and how to train them. *arXiv preprint arXiv:2211.03217*, 2022.

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

Fabio James Fehr and James Henderson. Nonparametric variational regularisation of pretrained transformers. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=Zu8OWNUC0u.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Hanwei Xu, Honghui Ding, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jingchang Chen, Jingyang Yuan, Jinhao Tu, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaichao You, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingxu Zhou, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645 (8081):633–638, September 2025. ISSN 1476-4687. doi: 10.1038/s41586-025-09422-z. URL http://dx.doi.org/10.1038/s41586-025-09422-z.

Shibo Hao, Yi Gu, Haodi Ma, Joshua Hong, Zhen Wang, Daisy Wang, and Zhiting Hu. Reasoning with language model is planning with world model. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pp. 8154–8173, 2023.

Hany Hassan, Anthony Aue, Chang Chen, Vishal Chowdhary, Jonathan Clark, Christian Federmann, Xuedong Huang, Marcin Junczys-Dowmunt, William Lewis, Mu Li, et al. Achieving human parity on automatic chinese to english news translation. *arXiv preprint arXiv:1803.05567*, 2018.

James Henderson and Fabio James Fehr. A VAE for transformers with nonparametric variational information bottleneck. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=6QkjC_cs03X.

Ralf Jung, Rodolphe Lepigre, Gaurav Parthasarathy, Marianna Rapoport, Amin Timany, Derek Dreyer, and Bart Jacobs. The future is ours: prophecy variables in separation logic. *Proc. ACM Program. Lang.*, 4(POPL), December 2019. doi: 10.1145/3371113. URL https://doi.org/10.1145/3371113.

Katie Kang, Eric Wallace, Claire Tomlin, Aviral Kumar, and Sergey Levine. Unfamiliar finetuning examples control how language models hallucinate. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 3600–3612, 2025.

Amirhossein Kazemnejad, Milad Aghajohari, Eva Portelance, Alessandro Sordoni, Siva Reddy, Aaron Courville, and Nicolas Le Roux. VinePPO: Refining credit assignment in RL training of LLMs. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=Myx2kJFzAn.

Chang Ma, Haiteng Zhao, Junlei Zhang, Junxian He, and Lingpeng Kong. Non-myopic generation of language models for reasoning and planning. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 35031–35058, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/56b694fb10c02b177e75a41f45825a74-Paper-Conference.pdf.

Charilaos Pipis, Shivam Garg, Vasilis Kontonis, Vaishnavi Shrivastava, Akshay Krishnamurthy, and Dimitris Papailiopoulos. Wait, wait, wait... why do reasoning models loop?, 2026. URL https://arxiv.org/abs/2512.12895.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Andy Shih, Dorsa Sadigh, and Stefano Ermon. Long horizon temperature scaling. In *International conference on machine learning*, pp. 31422–31434. PMLR, 2023.

Haruki Shirakami and James Henderson. Compositional generalization through gradient search in nonparametric latent space. In *International Conference on Learning Representations*, volume 2026, pp. 63622–63643, 2026.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling parameters for reasoning. In *International Conference on Learning Representations*, volume 2025, pp. 10131–10165, 2025.

Richard S. Sutton and Andrew G. Barto. *Reinforcement learning: an introduction*. Adaptive computation and machine learning series. The MIT Press, Cambridge, Massachusetts, second edition edition, 2018. ISBN 978-0-262-03924-6.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=1PL1NIMMrw.

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.

Yangzhen Wu, Zhiqing Sun, Shanda Li, Sean Welleck, and Yiming Yang. Inference scaling laws: An empirical analysis of compute-optimal inference for llm problem-solving. In *International Conference on Learning Representations*, volume 2025, pp. 55824–55845, 2025.

Yingce Xia, Fei Tian, Lijun Wu, Jianxin Lin, Tao Qin, Nenghai Yu, and Tie-Yan Liu. Deliberation networks: Sequence generation beyond one-pass decoding. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/c6036a69be21cb660499b75718a3ef24-Paper.pdf.

Yuxi Xie, Kenji Kawaguchi, Yiran Zhao, James Xu Zhao, Min-Yen Kan, Junxian He, and Michael Xie. Self-evaluation guided beam search for reasoning. *Advances in Neural Information Processing Systems*, 36:41618–41650, 2023.

Fangzhi Xu, Hang Yan, Chang Ma, Haiteng Zhao, Jun Liu, Qika Lin, and Zhiyong Wu. ϕ-decoding: Adaptive foresight sampling for balanced inference-time exploration and exploitation. *arXiv preprint arXiv:2503.13288*, 2025.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 11809–11822. Curran Associates, Inc., 2023. doi: 10.52202/075280-0517. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/271db9922b8d1f4dd7aaef84ed5ac703-Paper-Conference.pdf.

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 *International Conference on Learning Representations*, volume 2025, pp. 77875–77898, 2025.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. Dapo: An open-source llm reinforcement learning system at scale. *Advances in Neural Information Processing Systems*, 38:113222–113244, 2026.

A Appendix

A.1 Cross-attention to nonparametric latent variables

In order to access nonparametric latent variable representations, we use denoising attention (Henderson & Fehr, 2023). While standard attention considers a query accessing a set of vectors, denoising attention generalizes the operation over sets of vectors to probability distributions over vectors.

Comparison to standard attention. In standard attention, we project the input $u' \in \mathbb{R}^{1\times p}$ to the query through $W^Q \in \mathbb{R}^{p\times d}$. Keys and value are obtained by projecting the set of vectors $Z \in \mathbb{R}^{n\times p}$ through $W^K, W^V \in \mathbb{R}^{p\times d}$ respectively. We can group projections to map inputs to the space of Z:

$$\text{Attention}(u', Z; W^Q, W^K, W^V) = \text{Attn}(u'W^Q(W^K)^\top, Z)W^V = \text{Attn}(u, Z)W^V \tag{6}$$

where $u = u'W^Q(W^K)^\top, u \in \mathbb{R}^{1\times p}$. This new operation $\text{Attn}(u, Z)$ can be viewed as both a sum over vectors $z_i \in Z$, or as an integral over a distribution with non-zero probability at z_i. Explicitly, we have the following equivalence:

$$\text{Attn}(u, Z) = \text{softmax}\left(\tfrac{1}{\sqrt{d}} u Z^T\right) Z = \text{DAttn}(u; F_Z) \tag{7}$$

$$F_Z = \sum_{i=1}^{n} \frac{\exp\left(\frac{1}{2\sqrt{d}}\|z_i\|^2\right)}{\sum_{i=1}^{n}\exp\left(\frac{1}{2\sqrt{d}}\|z_i\|^2\right)} \delta_{z_i} \tag{8}$$

$$\text{DAttn}(u; F) = \frac{\int_v f(v)\, g(u; v, \sqrt{d}I)\, v\, dv}{\int_v f(v)\, g(u; v, \sqrt{d}I)\, dv} \tag{9}$$

Here, δ_{z_i} are Dirac delta functions at z_i, $f(\cdot)$ represents the density of distribution F, and $g(u;, v, \sqrt{d}I)$ is a Gaussian distribution. Thus, the attention operation can be viewed as taking the posterior mean of a corresponding mixture of impulses (and vice versa).

A.2 Inferring posterior DPs in transformers

Taking the output of the first l layers of the transformer $V \in \mathbb{R}^{n\times p}$, we linearly project these representations to pre-activation parameters $\boldsymbol{\mu} \in \mathbb{R}^{n\times p}$, $\log(\boldsymbol{\sigma^2}) \in \mathbb{R}^{n\times p}$, $\boldsymbol{\alpha} \in \mathbb{R}^n$. To ensure positive values, we apply an exponential activation function to our log-variance parameters and ReLU to our concentration parameters following (Henderson & Fehr, 2023). This returns one component $\langle \boldsymbol{\mu}_i, \boldsymbol{\sigma}_i, \alpha_i \rangle$ per transformer output, which parameterize our posterior Dirichlet Process.

A.3 ALGORITHMIC TASK DETAILS

Details regarding task configuration and example tokenization are given in Table. A.3.

Table 1: Synthetic task configurations and example tokenizations. P^8 denotes eight consecutive <pause> tokens.

Task	Configuration		Example tokenization
3-SAT	Variables:	12	Input: C 1 !3 12 C !2 4 7 ... P^8
	Clauses:	62	Output: 1 0 1 0 1 0 1 0 1 0 1 0
Graph coloring	Vertices:	16	Input: V 16 E 0 , 3 ; 0 , 7 ; ... P^8
	Colors:	4	Output: A B C D A B C D ...
	Edges:	36	
	Edges/class pair:	6	
	Min. degree:	2	
Sparse TSP	Vertices:	16	Input: V 16 E 0 3 W 5 E 3 7 W 1 ... P^8
	Edges:	16–32	Output: 0 3 7 12 ...
	Degree range:	2–4	
	Edge prob.:	0.08	
	Edge weights:	1–9	
Countdown	Starting numbers:	4	Input: 2 3 4 5 14 P^8
	Starting range:	1–99	Output: 2+3=5, 4+5=9, 5+9=14
	Target range:	10–100	
	Operations:	$+, -, \times, \div$	

A.4 TRAINING CONFIGURATION DETAILS

Model architecture. For both PT and AR baseline, we use a Qwen3 (Yang et al., 2025) style transformer architecture configured according to Table. A.4. In order to infer the posterior DP, PT possesses an additional projection matrix $\texttt{mu_proj} \in \mathbb{R}^{\texttt{hidden dim.} \times \texttt{hidden dim.}}$ as well as projection vectors $\texttt{var_proj} \in \mathbb{R}^{\texttt{hidden dim.}}$ and $\texttt{alpha_proj} \in \mathbb{R}^{\texttt{hidden dim.}}$, which follow the implementation used by (Henderson & Fehr, 2023).

Training configuration. Table. A.4 shows our training configuration across tasks, which is shared between PT and AR baseline. Hyperparamters specfic to PT are shown in Table. A.4.

Table 2: Model architecture configuration.

Property	**Value**
Decoder layers	12
Hidden dim.	768
SwiGLU intermediate size	3,072
Query heads / KV heads	12 / 12
Head dimension	128
Normalization	RMSNorm, epsilon 10^{-6}
Activation	SiLU
Positional encoding	RoPE, theta 10^6

Table 3: Training configuration across tasks.

Configuration	3-SAT	Graph coloring	Sparse TSP	Countdown
Training examples	500,000			
Maximum sequence length	280	192	256	128
Optimizer	AdamW			
Adam (β_1, β_2)	$(0.9, 0.95)$			
Adam ϵ	10^{-8}			
Learning rate (matrix / embedding / output)	6×10^{-4}			
Learning rate (scalar / normalization)	3×10^{-4}			
Learning-rate schedule	Linear warmup			
Learning-rate warmup	500 steps			
Weight decay (matrix parameters)	0.1			
Weight decay (embedding / output head)	0.001 / 0.01			
Gradient clipping (global norm)	1.0			
Global batch size	512 examples			
Microbatch size per GPU	128 examples			
GPUs	4			
Gradient accumulation	1			
Training steps	25,000			

Table 4: PT training hyperparameters.

Hyperparam.	Value
Gaussian KL coefficient	10^{-4}
Dirichlet KL coefficient	10^{-4}
DP Projections learning rate	6×10^{-4}
δ	1.0
Prior generation temperature	1.0