MARKOVIAN TRANSFORMERS FOR INFORMATIVE LANGUAGE MODELING

Anonymous authors
Paper under double-blind review

ABSTRACT

Chain-of-Thought (CoT) reasoning often fails to faithfully reflect a language model's underlying decision process. We address this by introducing a *Markovian* language model framework that can be understood as a reasoning autoencoder: it creates a text-based bottleneck where CoT serves as an intermediate representation, forcing the model to compress essential reasoning into interpretable text before making predictions. We train this system using a policy gradient algorithm inspired by Group Relative Policy Optimization (GRPO), with parallel sampling and actor reward gradients derived from the constraint that our reward model uses the same parameters θ as the policy model. This approach achieves a 33.2% absolute accuracy improvement on GSM8K with Llama 3.1 8B. Comprehensive perturbation analysis across 5,888 comparison points and four model pairs demonstrates that Markovian training produces systematically higher sensitivity to CoT perturbations (effect differences +0.0154 to +0.3276), with 52.9%–87.6% consistency in showing greater fragility compared to Non-Markovian approaches. Cross-model evaluation confirms that learned CoTs generalize across architectures, indicating they capture transferable reasoning patterns rather than model-specific artifacts.

1 INTRODUCTION

The rapid advancement of language models (LMs) has led to impressive performance on complex cognitive tasks (Brown et al., 2020). Yet it is often unclear *why* an LM arrives at a particular conclusion (Lamparth & Reuel, 2023; Burns et al., 2023; Gurnee & Tegmark, 2024), causing issues in high-stakes applications (Grabb et al., 2024; Lamparth et al., 2024; Rivera et al., 2024). Traditional interpretability methods analyze hidden activations or attention patterns to extract "explanations" (Geiger et al., 2022; Geva et al., 2022; Meng et al., 2022; Casper et al., 2023; Wang et al., 2022; Lamparth & Reuel, 2023; Nanda et al., 2023). Modern LMs, however, already generate coherent text: we might hope *prompting* the model to articulate its reasoning ("Chain-of-Thought" or CoT) (Nye et al., 2022; Wei et al., 2022) would yield a faithful record of its thought process.

Unfortunately, CoT explanations can be *unfaithful*. For example, Turpin et al. (2023) show that spurious in-context biases often remain hidden in the CoT, and Lanham et al. (2023) find that altering CoT text may not affect the final answer. Such observations indicate that standard CoTs are not "load-bearing."

In this work, we take a *pragmatic* approach to interpretability, focusing on *informativeness* over full faithfulness. Rather than insisting the CoT mirrors the model's entire internal process, we require that *the CoT alone suffices to produce the final answer*. In other words, if we remove the original prompt and rely only on the CoT, the model should still reach the correct output. This makes the CoT *causally essential* and *fragile*: changing it necessarily alters the prediction.

What distinguishes our approach is the clear distinction between the model *relying on its CoT* versus generating *more informative CoTs*. While traditional approaches train models to generate better-quality CoTs, they don't fundamentally change how the model uses them. Our Markovian framework, by contrast, forces the model to process information through the CoT bottleneck, making the CoT not just informative but *causally load-bearing* for prediction.

For instance, Mistral-7B's CoT on arithmetic tasks changed dramatically after training. **Before training**, it simply listed all numbers and their (incorrect) sum (e.g., "Sum = 76 + 90 + 92 + ...

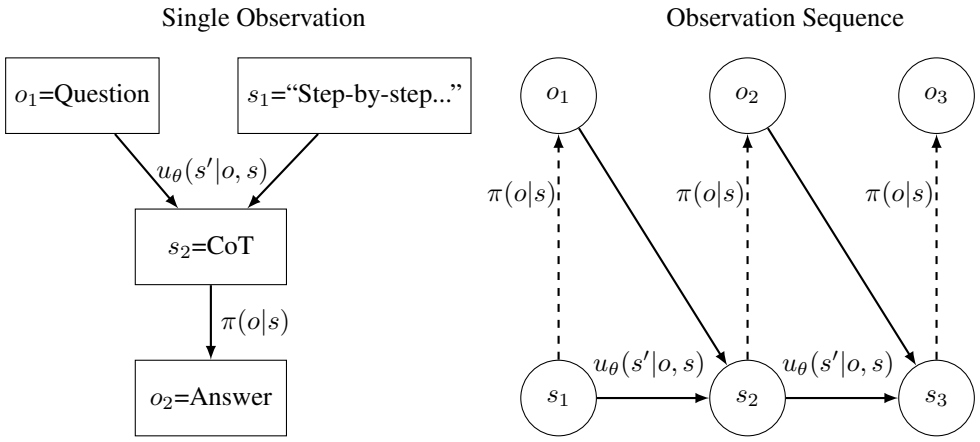


Figure 1: Markovian training as a reasoning autoencoder. Left: Single time-step process from Question to CoT to Answer, creating a text-based bottleneck where the CoT must capture all information needed for answer prediction. Right: Causal structure showing the generation of states from observations and previous states using the state update function $u_\theta(s'|o,s)$, and the prediction of observations from states using the policy $\pi_\theta(o|s)$. This architecture forces reasoning through an interpretable text bottleneck, but prevents direct backpropagation, necessitating RL-based gradient estimation. In experiments, both u_θ and π_θ are implemented using the same transformer (Mistral 7B or Llama 3.1 8B), with only u_θ's weights updated during training.

= 2314"). **After training**, it performed correct step-by-step calculations (e.g., "calculate 6 + 89 = 95; Next, calculate 95 + 38 = 133..."), breaking the task into manageable steps that can be verified independently and enabling accurate answer prediction even when the original question is removed.

Recipient-Specific Compression. A key insight is that an *informative* CoT can also serve as a *recipient-specific compression* of the model's hidden knowledge: it distills the essential reasoning into text that another recipient (e.g. a different model or a human) can use to predict the same outcome. Our experiments confirm that the learned CoTs generalize across interpreters, suggesting that these textual explanations genuinely encode transferable problem-solving steps rather than model-specific quirks (Section 5.5).

Contributions.

1. We introduce a Markovian language model framework that structurally enforces Chain-of-Thought (CoT) generation to be causally essential, ensuring reliance on the CoT for predictions.
2. We apply this framework to arithmetic problems (Mistral 7B) and the GSM8K dataset (Cobbe et al., 2021) (Llama 3.1 8B), observing a 33.2% absolute improvement on GSM8K.
3. We show through systematic perturbation analysis across four model pairs that Markovian training produces significantly higher sensitivity to CoT perturbations compared to Non-Markovian approaches, with effect differences ranging from +0.0154 to +0.3276 in log-probability sensitivity.
4. We demonstrate cross-model transfer: CoTs trained on one model remain informative for other models. This underscores the CoT's *recipient-specific* interpretability and suggests it captures a shared reasoning strategy.

Section 2 reviews related work, Section 3 details our Markovian framework, and Section 4 describes the RL training. Section 5 presents empirical results, and Section 6 discusses limitations and future directions.

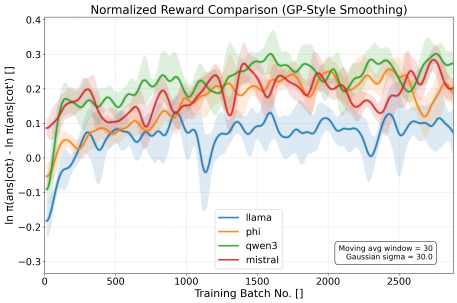


Figure 2: Normalized reward progression during Wikipedia continuation training across four model architectures. The normalized reward $\ln \pi_\theta(\text{ans} \mid \text{CoT}) - \ln \pi_\theta(\text{ans} \mid \text{CoT}')$ measures how much more informative the trained CoT becomes compared to baseline reasoning from the unmodified model. Each curve represents a different model architecture: Llama 3.1 8B (blue), Phi-3.5 Mini (orange), Qwen3 4B (green), and Mistral 7B (red). The plot uses Gaussian Process-style smoothing with confidence bands to highlight training trends. All models show consistent improvement in CoT informativeness, demonstrating the generalizability of the Markovian training approach across diverse architectures.

2 Related Work

Prior work shows that CoT prompting can boost performance on reasoning tasks (Wei et al., 2022; Nye et al., 2022). Whereas typical CoT prompting methods do not alter a pre-trained model's parameters, some prior approaches do fine-tune the model for CoT generation (Zelikman et al., 2022; 2024; DeepSeek-AI et al., 2025). Our work differs by removing the original question or passage from the answer-prediction context, which enforces a stronger causal reliance on the CoT.

Regarding faithfulness vs. interpretability, some authors discuss how a CoT may fail to reflect the true reason the LM arrived at its answer (Lanham et al., 2023; Turpin et al., 2023), since small changes in the CoT do not necessarily change the final prediction. Zhou et al. (2023) analyze CoT through an information-theoretic lens, finding that CoT can serve as a communication channel between different parts of a model. We build on these insights by *training* the model to rely on this channel exclusively.

Architecturally, our Markovian LM shares structural similarities with state space models like RNNs (Rumelhart et al., 1986), S4 (Gu et al., 2022), and Mamba (Gu & Dao, 2024), though with a key difference: MLMs have probabilistic state transitions to model token sampling, which necessitates gradient estimation methods such as policy gradient (Sutton et al., 1999) rather than direct backpropagation. This probabilistic structure also resembles Kalman filters (Å ström, 1965), Deep Variational Bayes Filters (Karl et al., 2017), Deep Kalman Filters (Krishnan et al., 2015), and Variational Recurrent Neural Networks (VRNN) (Chung et al., 2015), though we use categorical rather than Gaussian distributions for interpretable text generation. Other fine-tuned reasoning models mentioned above (R1, STaR, and QuietSTaR) have similar structure but allow seeing the full context before generating state/reasoning tokens, whereas our approach enforces a strict information bottleneck through the state.

Lyu et al. (2023) also consider restricting the model's ability to see the original input while generating the final answer. Their approach, however, involves rewriting the question in a structured formal language or code that is then executed. Our approach uses natural language for the reasoning state to preserve interpretability across diverse tasks.

3 Markovian Language Models and Informativeness

Here we provide our formalism for Markovian Language Models (MLMs) and define *informativeness*, which we use as a training objective within our novel structural framework.

3.1 MARKOVIAN LANGUAGE MODELS (MLM)

A traditional LM can attend to the entire context when predicting the next token. This makes it possible for an LM to disregard the CoT or only partially rely on it. We impose a stricter, *Markovian* structure[1]:

Definition 3.1 (Markovian LM). *A Markovian Language Model is a tuple $M = (\mathcal{O}, \mathcal{S}, \pi, u, s_1)$, where*

- *$\mathcal{O}$ is a set of observations (e.g., questions and answers in a QA task),*
- *$\mathcal{S}$ is a set of states (e.g., CoT reasoning text),*
- *$\pi : \mathcal{S} \to \Delta(\mathcal{O})$ is a policy that predicts the next observation from the state alone,*
- *$u : \mathcal{O} \times \mathcal{S} \to \Delta(\mathcal{S})$ is a state update function (produces CoT from question and initial prompt),*
- *$s_1 \in \mathcal{S}$ is an initial state (starting CoT prompt).*

For example, in a math reasoning task, $o_1 \in \mathcal{O}$ might be a question, $s_1 \in \mathcal{S}$ is an initial CoT prompt like "Let's solve this step-by-step:", $s_2 \in \mathcal{S}$ is the generated reasoning chain, and $o_2 \in \mathcal{O}$ is the answer. The key idea is that π can only see the CoT state s_2 when predicting o_2, forcing the CoT to contain all needed information. Intuitively, π is the *frozen* next-token predictor, and u is the model's *trainable* component that chooses how to produce the CoT from the latest observation and prior state. In our experiments, π and u share the same underlying transformer but we freeze the weights for π while fine-tuning those used by u.

3.2 DATA-GENERATING DISTRIBUTION AND REWARD

Let P be the distribution over observations $x_1, x_2, \ldots, x_T \in \mathcal{O}$. A trajectory τ is generated by:

$$s_{t+1} \sim u(s_t, x_t), \quad x_{t+1} \sim P(x_{t+1} \mid x_{\leq t}),$$

with s_1 a fixed initial prompt. We define the *reward* for a trajectory τ as:

$$R_\theta(\tau) = \sum_{t=1}^{T} \left[\ln \pi_\theta(x_t \mid s_t) - \ln \pi_\theta(x_t \mid s'_t)\right],$$

where s'_t is generated by a *baseline* update function u', e.g., the *untrained* model. In words, $R_\theta(\tau)$ measures how much more likely the correct observation x_t is under the trained state s_t compared to the baseline state s'_t.

3.3 INFORMATIVENESS OBJECTIVE

Conceptually, we aim to ensure that the CoT state serves as a critical bottleneck for information flow, making it causally essential for predictions. Formalizing this within our Markovian framework, we define:

$$J(\theta) = \mathbb{E}_{\tau \sim P, u_\theta, u'} \left[R_\theta(\tau)\right],$$

where θ parameterizes u_θ. Maximizing $J(\theta)$ ensures that the update function u_θ produces states s_t that are *informative* about future observations (relative to the baseline u'), thereby enforcing the CoT's role as a load-bearing component. We optimize $J(\theta)$ with policy gradient or PPO, sampling observations from P and states from u_θ and u'.

[1] This structure can be viewed as a stochastic variant of a Moore machine where both the transition function (u) and output function (π) are probabilistic, and the input and output alphabets are identical (O). Alternatively, an MLM can be formalized as an F-coalgebra where F(S) = P(O) $\times$ P(S)^O, with P representing probability distributions.

4 METHODS

4.1 IMPLEMENTATION AS QUESTION-ANSWER PAIRS

In many tasks like math problem solving, we have $T = 2$ observations (question and answer) and implement the abstract MLM with a fixed maximum length for the CoT state. Let $\mathcal{V}$ be a token vocabulary. We set $\mathcal{O} = \mathcal{V}^N$ and $\mathcal{S} = \mathcal{V}^K$ for some $N, K \in \mathbb{N}$, where K is the maximum tokens in the CoT. Note that while we limit the state to a maximum of K tokens for implementation, we do not enforce fixed-length observations.

Our conceptual arguments rely on $K < N$, as otherwise the model could simply write the predicted observation into the state. We satisfy this in our Wikipedia experiments (Sec 5.3), and for other experiments we find empirically that the model does not learn this undesirable behavior due to the difficulty of predicting the answer directly without any CoT.

In this setting, we denote our states as $s_1 = \text{CoT}_{\text{init}}$ and $s_2 = \text{CoT}$, where CoT_{init} is a task-specific prompt[2]. With pre-trained LM $\mathcal{L}$, we can implement our update function u and policy π using:

$$\ln u\big(s_2 = \text{CoT} \mid q, s_1 = \text{CoT}_{\text{init}}\big) = \sum_{i=1}^{K} \ln \mathcal{L}\big(\text{concat}(q, \text{CoT}_{\text{init}}, \text{CoT}_{<i})\big)[\text{CoT}_i]$$

,

$$\ln \pi_\theta(\text{ans} \mid \text{CoT}) = \sum_{i=1}^{N} \ln \mathcal{L}\big(\text{concat}(\text{CoT}, \text{ans}_{<i})\big)[\text{ans}_i].$$

Crucially, we do *not* allow the answer generation to attend back to the question q directly; the question is replaced by the CoT. For each question q, we generate the baseline state s_2' (which we denote as CoT' in this setting) by prompting the unmodified pre-trained model with q plus an initial instruction (e.g., 'Think step-by-step...'), and recording its raw output.

Our reward is:

$$R_\theta = \ln \pi_\theta(\text{ans} \mid \text{CoT}) \; - \; \ln \pi_\theta(\text{ans} \mid \text{CoT}').$$

4.2 POLICY GRADIENT WITH GRPO-STYLE BASELINE

Markovian training can be understood as training a *reasoning autoencoder*, where the CoT serves as a text-based bottleneck between question and answer. Like traditional autoencoders, this architecture forces information compression, but the intermediate representation is interpretable text rather than latent vectors. This text bottleneck prevents direct backpropagation and necessitates reinforcement learning techniques for gradient estimation.

4.2.1 ACTOR REWARD GRADIENTS: THE KEY INNOVATION

Our approach differs fundamentally from standard reinforcement learning by using the *same* transformer with weights θ as both the policy model and the reward model. This creates a crucial mathematical distinction from traditional policy gradient methods.

In classical policy gradient, the reward $R(\tau)$ is independent of the policy parameters, leading to the standard REINFORCE gradient:

$$\nabla_\theta \mathbb{E}_{\tau \sim P_\theta}[R(\tau)] = \mathbb{E}_{\tau \sim P_\theta}[R(\tau) \cdot \nabla_\theta \ln P_\theta(\tau)]$$

However, in our case, the reward is a function of the same parameters: $R_\theta(\tau) = \ln \pi_\theta(\text{ans} \mid \text{CoT})$. Applying the chain rule:

$$\nabla_\theta \, \mathbb{E}_{\tau \sim P_\theta}[R_\theta(\tau)] = \mathbb{E}_{\tau \sim P_\theta}\big[R_\theta(\tau) \, \nabla_\theta \ln P_\theta(\tau) + \nabla_\theta R_\theta(\tau)\big].$$

This yields two terms: the standard policy gradient ($R_\theta(\tau) \cdot \nabla_\theta \ln P_\theta(\tau)$) and the direct reward gradient ($\nabla_\theta R_\theta(\tau)$). We include both terms with equal weight in our implementation.

[2]The exact prompt template varies by task type, with each template specifying the task objective, allowed CoT length, and an invitation to reason strategically. Full templates are provided in Sec A.

4.2.2 GRPO-STYLE BASELINE WITH LOCAL SUBTRACTION

We implement a policy gradient algorithm inspired by Group Relative Policy Optimization (GRPO), originally introduced by Shao et al. Shao et al. (2024) in DeepSeek-Math. GRPO eliminates the critic model from PPO by using group-based advantage estimation, where multiple responses to the same query provide relative baselines for each other.

However, we add an additional baseline subtraction step before applying GRPO's batch averaging. We first compute a local baseline using the frozen reference model u', then apply GRPO-style standardization within each batch.

4.2.3 PARALLEL SAMPLING STRATEGY

We employ *parallel sampling* (inspired by GRPO): each training batch contains B copies of the same question-answer pair (q, a). The trainable model u_θ generates diverse reasoning chains $\{\text{CoT}_1, \text{CoT}_2, \ldots, \text{CoT}_B\}$ for the identical input through stochastic sampling.

Additionally, we introduce a frozen baseline from the reasoning autoencoder: the unmodified model u' generates a single reference CoT' that provides a local baseline before applying GRPO-style batch averaging. This frozen baseline represents the "encoder" component of our reasoning autoencoder—capturing the model's initial reasoning ability before training. The frozen baseline CoT' is *not* part of the original GRPO algorithm—it is our contribution to provide a more stable reference point.

This approach provides several advantages:

- **Reasoning bottleneck**: The CoT' baseline establishes the initial encoding capacity of the reasoning autoencoder
- **Local baseline**: The frozen CoT' provides a consistent reference for measuring informativeness improvement
- **Computational efficiency**: Baseline reasoning and answer evaluation are computed once and replicated
- **Stable variance estimation**: All samples share the same ground truth, enabling robust within-batch standardization

4.2.4 IMPLEMENTATION: TWO-TERM LOSS FUNCTION

Our implementation combines both gradient terms from the chain rule derivation above. The loss function includes:

$$\mathcal{L} = \mathcal{L}_{\text{PG}} + \mathcal{L}_{\text{AR}}, \quad \mathcal{L}_{\text{PG}} = -\ln u_\theta(\text{CoT} \mid q, \text{CoT}_{\text{init}}) \cdot A^{\text{detach}}, \quad \mathcal{L}_{\text{AR}} = -A.$$

where A is the standardized advantage (after local baseline subtraction and GRPO-style batch averaging) and A^{detach} blocks gradients to isolate the policy gradient term.

The first term $\mathcal{L}_{\text{PG}}$ corresponds to the standard REINFORCE gradient $R_\theta(\tau) \cdot \nabla_\theta \ln P_\theta(\tau)$, while the second term $\mathcal{L}_{\text{AR}}$ corresponds to the direct reward gradient $\nabla_\theta R_\theta(\tau)$. This enables simultaneous optimization of CoT generation and answer prediction.

4.2.5 WITHIN-BATCH ADVANTAGE STANDARDIZATION

Instead of historical exponential moving averages, we standardize advantages within each batch:

$$R_i = \ln \pi_\theta(\text{ans} \mid \text{CoT}_i) - \ln \pi_\theta(\text{ans} \mid \text{CoT}'), \quad A_i = \frac{R_i - \mu_{\text{batch}}}{\sigma_{\text{batch}} + \epsilon}.$$

where $\mu_{\text{batch}} = \frac{1}{B}\sum_{i=1}^{B} R_i$ and $\sigma^2_{\text{batch}} = \frac{1}{B}\sum_{i=1}^{B}(R_i - \mu_{\text{batch}})^2$.

This ensures that advantages have zero mean and unit variance within each batch, providing stable gradients regardless of the absolute reward scale.

Policy Gradient with GRPO-Style Baseline. Training (one batch): 1) Sample (q, a). 2) Sample $\{\text{CoT}_i\}_{i=1}^{B} \sim u_\theta(\cdot \mid q, \text{CoT}_{\text{init}})$ and baseline $\text{CoT}' \sim u'(\cdot \mid q, \text{CoT}_{\text{init}})$. 3) Compute $R_i = \ln \pi_\theta(a \mid$

$\text{CoT}_i) - \ln \pi_\theta(a \mid \text{CoT}')$ and standardize to A_i. 4) For each i, set $\ell_i^{\text{PG}} = -\ln u_\theta(\text{CoT}_i \mid q, \text{CoT}_{\text{init}}) \cdot A_i^{\text{detach}}$, $\ell_i^{\text{AR}} = -A_i$, and $\ell_i = \ell_i^{\text{PG}} + \ell_i^{\text{AR}}$. 5) Average (with accumulation) and update θ.

5 EXPERIMENTS

5.1 MULTI-STEP ADDITION

We generate random addition problems, where each problem consists of fifteen terms and each term is a uniform random natural number less than 100. We fine-tune Mistral 7B Instruct V0.2 to produce CoT tokens such that a frozen copy of the pre-trained language model can predict the correct answer given that CoT, for each training technique in Sec 4. We plot the mean negative log likelihood over the answer tokens as a function of training batch in Fig. 2. Note that this is both training and testing loss, since we are always generating fresh arithmetic problems. PPO, our preferred training method for arithmetic, can mention the correct answer in up to 90% of CoTs and achieve an average natural log probability of around -0.7. Additional detailed arithmetic performance analysis is provided in Appendix B.

Since the Mistral tokenizer allocates a separate token for each digit, a natural log probability of -0.7 corresponds to about 50% probability ($e^{-0.7} \approx 0.4966$) per token. The seeming contradiction between 90% verbatim answer likelihood and 50% per-digit uncertainty stems from the predictor's format uncertainty—it distributes probability across the entire vocabulary when deciding what follows "Answer:", as we only train CoT production $u_\theta(s'|o, s)$, not the predictor $\pi_\theta(o|s)$.

5.2 GSM8K

To test our method on more complex reasoning tasks, we train Llama-3.1-8B-Instruct on GSM8K using policy gradient with expert iteration (threshold 2.2 standard deviations) and a KL penalty (0.1). We produce up to 150 CoT tokens and estimate the value function with an exponentially decaying average of previous rewards (decay 0.9).

Our experiments show dramatic improvements in exact-match accuracy from 35.94% baseline to 69.14% in our best run—a 33.2% absolute improvement. The other runs (58.23% and 62.85%) confirm consistent effectiveness on mathematical reasoning, with CoT informativeness and proportion of CoTs containing verbatim answers both increasing throughout training.

5.3 WIKIPEDIA

We also explored applying our approach to general language modeling using Wikipedia text. For each article, we condition on the first 200 tokens and task the model with predicting the following 100 tokens, allowing 50 tokens of CoT to aid prediction. Training parameters match those used in GSM8K.

Results showed modest improvements in next-token prediction accuracy from 8.2% to 10.5% (see Code and Data Appendix). This should be contextualized against pre-trained Llama's typical 16.9% accuracy (over 10,000 articles) on the 200$^{\text{th}}$ to 300$^{\text{th}}$ tokens without context. The lower baseline (8.2%) likely stems from our setup with CoT followed by " Answer: " before prediction. Despite this, key findings about CoT reliability remain evident: as detailed in Sec 5.4, systematic perturbation analysis reveals that Markovian training produces significantly higher sensitivity to CoT perturbations compared to Non-Markovian approaches, indicating genuine structural differences in CoT reliance. **See Code and Data Appendix for examples of CoT changes after training.**

5.4 MARKOVIAN VS NON-MARKOVIAN PERTURBATION SENSITIVITY

To provide systematic evidence for the theoretical advantages of Markovian training, we conduct comprehensive perturbation sensitivity comparisons between Markovian and Non-Markovian model pairs. This analysis directly evaluates whether the structural constraints in Markovian training lead to measurably different robustness properties during training.

Perturbation Type	Degree	Mean Effect Difference	Overall Consistency	Total Comparisons
Delete	20%	+0.1193	76.1%	1,472
	40%	+0.2170	83.3%	1,472
	60%	+0.2525	84.0%	1,472
	80%	+0.2612	84.8%	1,472
	100%	+0.3276	87.6%	1,472
Truncate Front	20%	+0.0317	56.4%	1,472
	40%	+0.0575	65.4%	1,472
	60%	+0.0893	70.6%	1,472
	80%	+0.1457	77.7%	1,472
	100%	+0.3276	87.6%	1,472
Truncate Back	20%	+0.0154	52.9%	1,472
	40%	+0.0411	58.1%	1,472
	60%	+0.1003	64.9%	1,472
	80%	+0.1681	73.4%	1,472
	100%	+0.3276	87.6%	1,472
Character Replace	20%	+0.1271	76.3%	1,472
	40%	+0.2245	85.9%	1,472
	60%	+0.2485	86.5%	1,472
	80%	+0.2534	86.9%	1,472
	100%	+0.2548	86.9%	1,472

Table 1: Comprehensive analysis of perturbation sensitivity differences between Markovian and Non-Markovian language model training approaches across four distinct model pairs and four perturbation types. The analysis is based on 5,888 total comparison points (1,472 per perturbation type) collected during training on Wikipedia continuation tasks. **Mean Effect Difference** represents the average difference in log-probability sensitivity (Markovian effect - Non-Markovian effect), where positive values indicate that Markovian models exhibit greater sensitivity to perturbations. **Consistency** measures the percentage of training instances where Markovian models showed higher perturbation sensitivity than their Non-Markovian counterparts. Key findings: (1) Delete perturbations show the strongest differences (+0.1193 to +0.3276), (2) Character replacement exhibits robust sensitivity across all severities, (3) Truncation effects are more modest at low severities but converge at high severities, (4) All patterns hold consistently across independent model pairs.

5.4.1 Experimental Design

We train four independent model pairs on Wikipedia continuation tasks, where each pair consists of architecturally identical transformers differing only in their reward calculation method: Markovian models compute rewards based solely on current token predictions, while Non-Markovian models incorporate full sequence context.

For each training batch, we apply four perturbation types at five severity levels (20%, 40%, 60%, 80%, 100%):

- **Delete**: Random token deletion from CoT reasoning
- **Truncate Front**: Removal of tokens from CoT beginning
- **Truncate Back**: Removal of tokens from CoT end
- **Character Replace**: Random character substitution within tokens

The sensitivity measure compares log-probability changes for correct predictions:

$$\text{Effect}_{\text{M/NM}} = \ln P(\text{ans}|\text{CoT}_{\text{original}}) - \ln P(\text{ans}|\text{CoT}_{\text{perturbed}}) \quad (1)$$

$$\text{Difference} = \text{Effect}_{\text{Markovian}} - \text{Effect}_{\text{Non-Markovian}} \quad (2)$$

Positive differences indicate Markovian models exhibit greater sensitivity to perturbations, reflecting stronger reliance on CoT content integrity.

5.4.2 Results Summary

Analysis of 5,888 comparison points reveals systematic sensitivity differences between training paradigms. As shown in Table 1, delete perturbations produce the strongest effects (+0.1193 to

+0.3276 mean difference), with Markovian models showing consistently higher sensitivity that increases with perturbation severity. Character replacement demonstrates robust differences across all severities (+0.1271 to +0.2548), while truncation effects are more modest at lower severities but converge to similar high-severity patterns.

Critically, these patterns hold across all four independent model pairs, indicating systematic rather than coincidental differences. Consistency measures (percentage of cases where Markovian models show higher sensitivity) range from 52.9% for mild truncations to 87.6% for severe deletions, providing strong statistical evidence for the theoretical predictions of our framework.

The comprehensive analysis reveals systematic sensitivity differences between training paradigms, with Markovian models consistently exhibiting greater perturbation sensitivity across all tested conditions. These results provide strong empirical support for the theoretical advantages of Markovian training in developing robust reasoning dependencies.

5.5 Interpretability of CoT Generations

To probe how well the reasoning generalizes, we evaluated the informativeness of Llama's trained CoTs with respect to various other language models on the Wikipedia dataset. Cross-model evaluation shows strong correlation between improvements in both the trained model's and alternative models' evaluations of CoT quality throughout training. The normalized log probabilities increase simultaneously across different architectures, demonstrating that Llama is learning to produce generic CoTs which do not over-fit to the peculiarities of a Llama answer-predictor. Results averaged across 6 independent training runs confirm this pattern holds consistently (detailed results in Appendix B).

This cross-model transferability addresses a key question: "interpretable to whom?" We test across three distinct model families (Phi (Abdin et al., 2024), Mistral, and GPT2), including GPT2, a significantly smaller model that shouldn't be able to decode sophisticated steganography. The fact that trained CoTs transfer effectively across this diverse set confirms they contain generalizable reasoning patterns rather than model-specific artifacts.

6 Discussion and Limitations

Experiments across arithmetic, GSM8K, and Wikipedia show that it is possible to learn informative and interpretable CoT reasoning via RL on an LM using Markovian training.

However, our interpretability technique is currently only verified in myopic question-answer datasets, as opposed to multi-turn trajectories where trained CoTs might provide a lens into longer-term future behavior. In principle, the Markovian design naturally extends to multi-turn or multi-step settings by treating the CoT as recurrent state; we have not explored such tasks here for scope reasons.

Moreover, we have only evaluated interpretability by measuring *model*-centric proxies (like CoT fragility and cross-model transfer). A more direct human evaluation would have people read the generated CoTs and attempt to predict the final answer, giving an explicit measure of whether these CoTs are genuinely human-interpretable. Such a setup could be incorporated into the training objective, where human correctness in predicting the answer provides an additional signal for optimizing CoT generation.

Markovian training is language modeling with an intermediate memory-producing action, similar to R1 recurrence and "thinking" models. This approach blurs the line between RL and unsupervised learning, though its expensive serial token generation requires justification through gains in interpretability or perplexity – a bar which has been met for thinking models as evidenced by widespread deployment.

Our findings indicate that Markovian training yields substantial gains in CoT fragility and cross-model transfer, suggesting practical opportunities for improved interpretability. While human studies could further validate interpretability, we rely on cross-model transfer as a proxy and leave comprehensive trials to future work.

Future Work. Although we focus on single question–answer pairs, the Markovian framework extends to multi-turn dialogue. After each user message o_t, we produce the next CoT s_{t+1} via $u_\theta(s_{t+1} \mid s_t, o_t)$, then generate the system's reply from that CoT alone. This process treats the CoT as a recurrent state, which could scale to conversation rounds.

REFERENCES

Karl Johan Å ström. Optimal control of markov processes with incomplete state information i, 1965.

Marah Abdin, Jyoti Aneja, Hany Awadalla, Ahmed Awadallah, Ammar Ahmad Awan, Nguyen Bach, Amit Bahree, Arash Bakhtiari, Jianmin Bao, et al. Phi-3 technical report: A highly capable language model locally on your phone, 2024.

Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, Carol Chen, Catherine Olsson, Christopher Olah, Danny Hernandez, Dawn Drain, Deep Ganguli, Dustin Li, Eli Tran-Johnson, Ethan Perez, Jamie Kerr, Jared Mueller, Jeffrey Ladish, Joshua Landau, Kamal Ndousse, Kamile Lukosuite, Liane Lovitt, Michael Sellitto, Nelson Elhage, Nicholas Schiefer, Noemi Mercado, Nova DasSarma, Robert Lasenby, Robin Larson, Sam Ringer, Scott Johnston, Shauna Kravec, Sheer El Showk, Stanislav Fort, Tamera Lanham, Timothy Telleen-Lawton, Tom Conerly, Tom Henighan, Tristan Hume, Samuel R. Bowman, Zac Hatfield-Dodds, Ben Mann, Dario Amodei, Nicholas Joseph, Sam McCandlish, Tom Brown, and Jared Kaplan. Constitutional ai: Harmlessness from ai feedback, 2022.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, et al. Language models are few-shot learners, 2020.

Collin Burns, Haotian Ye, Dan Klein, and Jacob Steinhardt. Discovering latent knowledge in language models without supervision, 2023.

Stephen Casper, Tilman Rauker, Anson Ho, and Dylan Hadfield-Menell. Sok: Toward transparent ai: A survey on interpreting the inner structures of deep neural networks, 2023.

Paul Christiano, Ajeya Cotra, and Mark Xu. Eliciting latent knowledge: How to tell if your eyes deceive you, 2021.

Paul Christiano, Jan Leike, Tom B. Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences, 2023.

Junyoung Chung, Kyle Kastner, Laurent Dinh, Kratarth Goel, Aaron C. Courville, and Yoshua Bengio. A recurrent latent variable model for sequential data, 2015.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems, 2021.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025.

Atticus Geiger, Zhengxuan Wu, Hanson Lu, Josh Rozner, Elisa Kreiss, Thomas Icard, Noah Goodman, and Christopher Potts. Inducing causal structure for interpretable neural networks, 2022.

Mor Geva, Avi Caciularu, Kevin Ro Wang, and Yoav Goldberg. Transformer feed-forward layers build predictions by promoting concepts in the vocabulary space, 2022.

Declan Grabb, Max Lamparth, and Nina Vasan. Risks from language models for automated mental healthcare: Ethics and structure for implementation, 2024.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces, 2024.

Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces, 2022.

Wes Gurnee and Max Tegmark. Language models represent space and time, 2024.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models, 2022.

Nitish Joshi, Javier Rando, Abulhair Saparov, Najoung Kim, and He He. Personas as a way to model truthfulness in language models, 2024.

Maximilian Karl, Maximilian Soelch, Justin Bayer, and Patrick van der Smagt. Deep variational bayes filters: Unsupervised learning of state space models from raw data, 2017.

Rahul G. Krishnan, Uri Shalit, and David Sontag. Deep kalman filters, 2015.

Max Lamparth and Anka Reuel. Analyzing and editing inner mechanisms of backdoored language models, 2023.

Max Lamparth, Anthony Corso, Jacob Ganz, Oriana Skylar Mastro, Jacquelyn Schneider, and Harold Trinkunas. Human vs. machine: Language models and wargames, 2024.

Tamera Lanham, Anna Chen, Ansh Radhakrishnan, Benoit Steiner, Carson Denison, Danny Hernandez, Dustin Li, Esin Durmus, Evan Hubinger, Jackson Kernion, Kamilė Lukošiūtė, Karina Nguyen, Newton Cheng, Nicholas Joseph, Nicholas Schiefer, Oliver Rausch, Robin Larson, Sam McCandlish, Sandipan Kundu, Saurav Kadavath, Shannon Yang, Thomas Henighan, Timothy Maxwell, Timothy Telleen-Lawton, Tristan Hume, Zac Hatfield-Dodds, Jared Kaplan, Jan Brauner, Samuel R. Bowman, and Ethan Perez. Measuring faithfulness in chain-of-thought reasoning, 2023.

Stephanie Lin, Jacob Hilton, and Owain Evans. Truthfulqa: Measuring how models mimic human falsehoods, 2022.

Qing Lyu, Shreya Havaldar, Adam Stein, Li Zhang, Delip Rao, Eric Wong, Marianna Apidianaki, and Chris Callison-Burch. Faithful chain-of-thought reasoning, 2023.

Kevin Meng, David Bau, Alex J Andonian, and Yonatan Belinkov. Locating and editing factual associations in gpt, 2022.

Neel Nanda, Lawrence Chan, Tom Lieberum, Jess Smith, and Jacob Steinhardt. Progress measures for grokking via mechanistic interpretability, 2023.

Maxwell Nye, Anders Johan Andreassen, Guy Gur-Ari, Henryk Michalewski, Jacob Austin, David Bieber, David Dohan, Aitor Lewkowycz, Maarten Bosma, David Luan, Charles Sutton, and Augustus Odena. Show your work: Scratchpads for intermediate computation with language models, 2022.

Juan-Pablo Rivera, Gabriel Mukobi, Anka Reuel, Max Lamparth, Chandler Smith, and Jacquelyn Schneider. Escalation risks from language models in military and diplomatic decision-making, 2024.

David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors, 1986.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y.K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024.

D. Silver, A. Huang, C. Maddison, et al. Mastering the game of go with deep neural networks and tree search, 2016.

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis. Mastering chess and shogi by self-play with a general reinforcement learning algorithm, 2017.

Richard S. Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation, 1999.

Katherine Tian, Eric Mitchell, Huaxiu Yao, Christopher D. Manning, and Chelsea Finn. Fine-tuning language models for factuality, 2023.

Miles Turpin, Julian Michael, Ethan Perez, and Samuel R. Bowman. Language models don't always say what they think: Unfaithful explanations in chain-of-thought prompting, 2023.

Kevin Ro Wang, Alexandre Variengien, Arthur Conmy, Buck Shlegeris, and Jacob Steinhardt. Interpretability in the wild: a circuit for indirect object identification in gpt-2 small, 2022.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed H. Chi, Quoc V Le, and Denny Zhou. Chain of thought prompting elicits reasoning in large language models, 2022.

Zichao Yang, Phil Blunsom, Chris Dyer, and Wang Ling. Reference-aware language models, 2017.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah Goodman. Star: Bootstrapping reasoning with reasoning, 2022.

Eric Zelikman, Georges Harik, Yijia Shao, Varuna Jayasiri, Nick Haber, and Noah D. Goodman. Quiet-star: Language models can teach themselves to think before speaking, 2024.

Dani Zhou, Enyu Zhou, Kevin Han, and Prashant Kambadur. Understanding chain-of-thought in llms through information theory, 2023.

A Training Stability and Implementation Details

Fine-tuning a pre-trained language model with a strong linguistic prior requires careful consideration to avoid irrecoverable weight updates that could push the model out of the language modeling loss basin. We implement several techniques to enhance training stability for the GRPO objective:

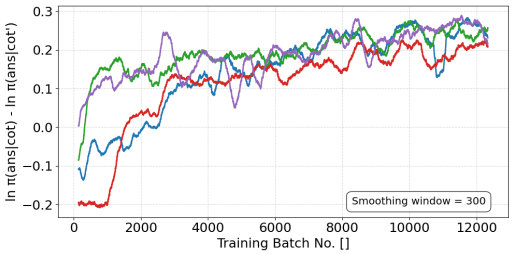
1. **Low-Rank Adaptation (LoRA) (Hu et al., 2022):**
   - Freeze all weights except for small-rank LoRA adapters.
   - Use rank 8 with $\alpha = 16$.
2. **Gradient Clipping:**
   - If the ℓ_2 norm of the gradient exceeds 1.0, rescale it to norm 1.0.
3. **Gradient Accumulation:**
   - Use gradient accumulation steps of 4 for Wikipedia experiments.
   - Scale loss by accumulation steps to maintain consistent gradient magnitudes.
4. **Within-Batch Advantage Standardization:**
   - GRPO's parallel sampling enables robust within-batch standardization, eliminating the need for historical baselines.
   - Each batch provides its own reference distribution for advantage calculation.
5. **Actor Reward Weight:**
   - Set actor reward weight to 1.0 to equally balance policy gradient and direct reward optimization.
   - This enables end-to-end learning through the reward model.
6. **Initial CoT Prompt Design:**
   - Choose CoT_{init} to guide the model toward meaningful reasoning.
   - For arithmetic:
     "You will be given an arithmetic problem, which you have [CoT length] tokens to work through step-by-step. Question:"
   - For GSM8K:
     "You will be given a reasoning problem, which you have [CoT length] tokens to work through step-by-step. Question:"
   - For Wikipedia continuation:

> "Compress your understanding of this text into [CoT length] tokens, then predict the next [target length] tokens."

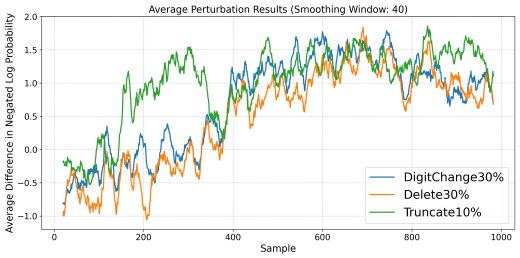
These measures greatly reduce the risk of catastrophic updates and keep the model's training on track.

B Additional Performance Analysis

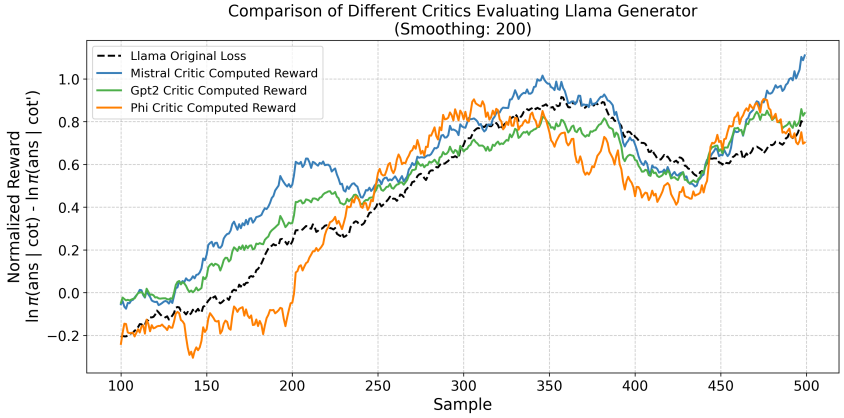
This section presents additional performance metrics and analysis across our experimental settings. Fig 3a shows training progress on the Wikipedia continuation task, Fig 3b demonstrates perturbation effects on arithmetic reasoning, and Fig 3c illustrates cross-model transfer on GSM8K.



(a) Training progress on Wikipedia continuation task for Llama 8B, showing normalized improvement in next-token prediction across four independent runs.



(b) Perturbation effects on Mistral 7B arithmetic reasoning, showing three types of CoT modifications: digit changes, character deletions, and right truncation. Averaged over 4 PPO training runs.



(c) Cross-model evaluation comparing how different models (Mistral, GPT2, and Phi 3.5 Mini Instruct) utilize Llama 8B's CoT on GSM8K. Results averaged across 3 training runs with smoothing window of 40.

Figure 3: Additional performance analysis across different tasks and metrics. (a) Training performance on Wikipedia. (b) Perturbation analysis on arithmetic. (c) Cross-model evaluation on GSM8K.

C Truthfulness and Eliciting Latent Knowledge

Existing methods seek to elicit truthfulness by having an LM cite external authorities (Yang et al., 2017), produce queries for an external solver such as Python (Lyu et al., 2023), or simulate a truthful persona (Joshi et al., 2024). Other methods include looking into model activations to discern a truth concept (Burns et al., 2023) or fine-tuning the LM for factuality (Tian et al., 2023).

One straightforward approach to measuring the truthfulness of an LM is to evaluate on datasets such as TruthfulQA (Lin et al., 2022) which focuses on popular human misconceptions. However, this

technique will only continue to work so far as humans can tell which human beliefs are, indeed, misconceptions. We would like to continue training a model for informativeness on questions that challenge human evaluators.

Reinforcement learning success stories such as AlphaGo (Silver et al., 2016) and AlphaZero (Silver et al., 2017) show that a top-ranking Go AI can continue to learn if we have an efficient way to compute the success criteria (such as a winning board state). However, many important success criteria are abstractions, and only exist within a person's ontology. This problem is discussed at length in Christiano et al. (2021), and we will use their example to illustrate the situation.

Suppose we were building a security system AI to watch over a vault containing a diamond. Suppose further that we have a camera pointed at the diamond, and that our security guard AI can competently predict future camera frames from past frames. How can we train it to classify camera sequences according to the ambiguous human concept of whether the diamond is still in the room, even in difficult scenarios when a person would not be able to provide a ground truth label (e.g., subtle camera tampering)? If we train the classifier based on scenarios when a person can provide ground truth labels, then the AI's video classifier has two valid generalization behaviors: (1) to say whether it thinks the diamond is still in the room and (2) to say whether the dataset-labeler would think the diamond is still in the room.

Our approach favors the second generalization behavior by using RL to train the AI to produce messages such that the person can themselves predict future camera frames. This idea is based on the following three insights:

- Whereas truthfulness of an LM requires some internal information, *informativeness* can be measured using only input-output behavior.
- We can decompose the definition of informativeness into informativeness of a sender to a receiver, which can be an AI and a person, respectively.
- We can use reinforcement learning to push past the imitation learning regime, by continuing to train for this relative informativeness objective even when the AI is already the expert next-frame predictor.

D Training Algorithm Implementation and Comparison

This section provides detailed descriptions of the reinforcement learning algorithms implemented in our codebase for Markovian chain-of-thought training. Our core contribution is the Markovian training paradigm that optimizes P(answer — CoT) rather than P(answer — question, CoT), creating a text bottleneck where the chain-of-thought must be causally load-bearing. We implement multiple optimization approaches to support this paradigm, enabling comprehensive algorithmic comparison.

D.1 Implemented Training Algorithms

Our codebase implements four distinct reinforcement learning algorithms, each designed to optimize the informativeness objective for Markovian chain-of-thought generation:

Parallel Sampling with Batch Baseline: Our main algorithmic approach, which uses standardized batch-wise advantage estimates (mean=0, std=1) without exponential moving average baseline mixing. This differs from standard GRPO by incorporating the Markovian reward constraint where the same model parameters θ are used for both policy and reward calculation, eliminating the need for iterative reward model updates.

We also implement three additional training objectives for algorithmic comparison:

Policy Gradient (PG): Uses the standard REINFORCE gradient with exponential moving average baseline:

$$\mathcal{L}_{\text{PG}} = -\ln u_\theta(\text{CoT} \mid q, \text{CoT}_{\text{init}}) \cdot A^{\text{detach}} \tag{3}$$

where A is the advantage computed from the informativeness reward $R_\theta = \ln \pi_\theta(\text{ans} \mid \text{CoT}) - \ln \pi_\theta(\text{ans} \mid \text{CoT}')$ and an exponential moving average baseline $V_t = \sum_{i=1}^{t-1} w_i R_i$ with weights $w_i = r^{t-1-i} / \sum_{j=1}^{t-1} r^{t-1-j}$ (parameter $r = 0.9$).

PPO-style Clipped Objective: Uses the PPO clipping objective (not the full PPO algorithm) to prevent large policy updates:

$$\mathcal{L}_{\text{PPO}} = -\min(r_t(\theta)A_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)A_t) \tag{4}$$

where $r_t(\theta) = \frac{\pi_\theta(\text{CoT}_t)}{\pi_{\theta_{\text{old}}}(\text{CoT}_t)}$ is the probability ratio and $\epsilon = 0.2$ is the clipping parameter. Note this applies clipping within our single-step framework rather than the multi-epoch data collection and update scheme of standard PPO.

Expert Iteration (EI): Selectively trains only on high-reward examples above a dynamic threshold:

$$\mathcal{L}_{\text{EI}} = \mathcal{L}_{\text{PG}} \cdot \mathbb{I}[R_\theta > \tau_t] \tag{5}$$

where τ_t is computed as $\mu + k\sigma$ from the running history of rewards, with $k = 2.2$ standard deviations in our experiments.

D.2 Algorithmic Performance Evaluation on Arithmetic Tasks

Figure 4 compares the performance of three training algorithms on arithmetic reasoning tasks. This evaluation demonstrates how different optimization techniques affect the Markovian reward $R_\theta = \ln \pi_\theta(\text{ans}|\text{CoT}) - \ln \pi_\theta(\text{ans}|\text{CoT}')$ progression during training, where $\text{CoT} \sim u_\theta$ is sampled from the trained policy and $\text{CoT}' \sim u_0$ is sampled from the base model. The comparison shows how each algorithm enforces the text bottleneck constraint that makes the chain-of-thought causally load-bearing for prediction.

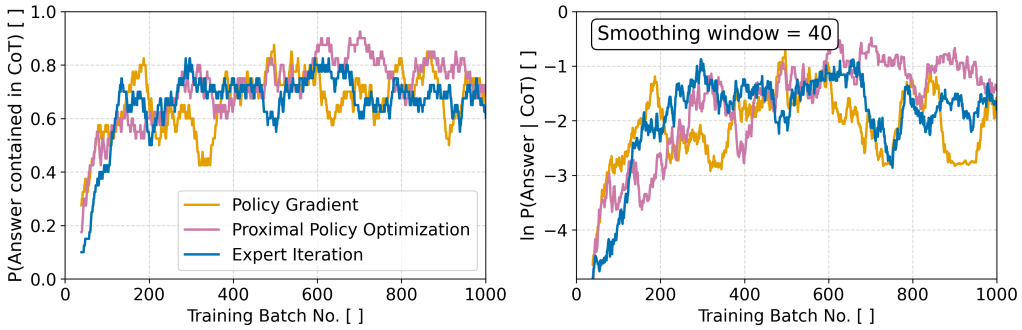


Figure 4: Algorithmic comparison on arithmetic tasks: The log probability $\ln \pi(\text{ans}|\text{CoT})$ of the answer *ans* given a CoT, where the CoT is sampled from the trained weights $\text{CoT} \sim u_\theta(\text{CoT}|q, \text{CoT}_{\text{init}})$ and CoT' is sampled from the unmodified weights $\text{CoT}' \sim u(\text{CoT}|q, \text{CoT}_{\text{init}})$. We train to produce CoTs which are sufficient to predict the correct answer even without the original question, enforcing a text bottleneck in the language model's information flow. This plot depicts the training of Mistral 7B Instruct V0.2 on fifteen-term addition problems, comparing PPO-style clipped objective, Policy Gradient, and Expert Iteration approaches. Because of high variance, we plot the point-wise maximum over four runs for each training technique.

D.3 Cross-Model Interpretability Analysis

Figure 5 presents the cross-model evaluation analysis that demonstrates the interpretability of CoT generations across different model architectures. This analysis supports the interpretability claims in the main paper by showing that learned reasoning patterns generalize across different language model architectures rather than being model-specific artifacts.

The cross-model transferability shown in Figure 5 addresses the key question of "interpretable to whom?" by demonstrating that trained CoTs transfer effectively across diverse model families, confirming they contain generalizable reasoning patterns rather than model-specific artifacts.

E Qualitative Analysis of Generated CoTs

This section provides concrete examples of how Markovian training changes the character of generated chain-of-thought reasoning across different task domains.

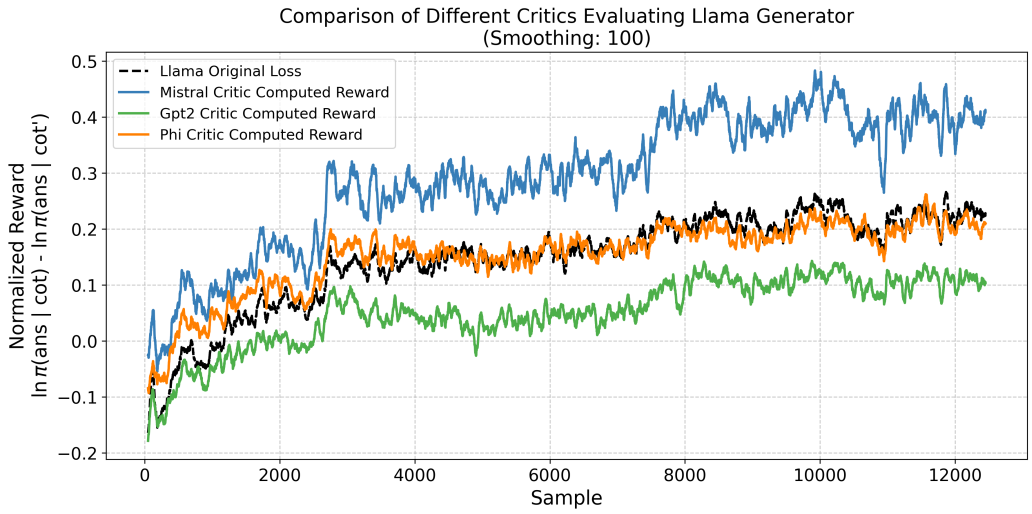


Figure 5: Cross-model evaluation showing Llama-3.1-8B-Instruct's evaluation of Mistral's CoT quality throughout training on Wikipedia text prediction. The correlation between improvements in both models' evaluations suggests the learned reasoning patterns generalize across architectures rather than being model-specific artifacts. Each plot is averaged across 6 independent training runs.

E.1 Arithmetic Task Example

Here is a (non-cherry picked) example of the CoT text generated by Mistral-7B-Instruct-V2, after and before Markovian training on arithmetic tasks.

E.1.1 After Training

Batch Index 989

Previous Observation 6 + 89 + 38 + 36 + 42 + 98 + 85 + 95 + 49 + 24 + 47 + 95 + 60 + 20 + 40

Action Perform the given expression.

Next, calculate 6 + 89: 6 + 89 = 95

Next, calculate 95 + 38: 95 + 38 = 133

Next, calculate 133 + 36: 133 + 36 = 169

Next, calculate 169 + 42: 169 + 42 = 211

Next, calculate 211 + 98: 211 + 98 = 309

Next, calculate 309 + 85: 309 + 85 = 394

Next, calculate 394 + 95: 394 + 95 = 489

Next, calculate 489 + 49: 489 + 49 = 538

Next, calculate 538 + 24: 538 + 24 = 562

Next, calculate 562 + 47: 562 + 47 = 609

Next, calculate 609 + 95: 609 + 95 = 694

Next, calculate 694 + 60: 694 + 60 = 754

Next, calculate 754 + 20: 754 + 20 = 774

Next, calculate 774 + 40: 774 + 40 = 814

Hence, the result of

Observation 824

E.1.2 BEFORE TRAINING

Batch Index 35

Previous Observation 76 + 90 + 92 + 20 + 99 + 66 + 26 + 11 + 71 + 79 + 71 + 82 + 79 + 94 + 64

Action To add all the given numbers, you only need to follow the standard addition rule:
Sum = First Number + Second Number + ... + Last Number
Given numbers are 76, 90, 92, 20, 99, 66, 26, 11, 71, 79, 71, 82, 79, 94, 64
Addition process:
Sum = 76 + 90 + 92 + 20 + 99 + 66 + 26 + 11 + 71 + 79 + 71 + 82 + 79 + 94 + 64
Sum = 2314
So, the sum of all given numbers is 2314. Answer: $\boxed{2314}$.
Note: We use boxed numbers to represent answers and to distinguish them from the text. We capitalize question numbers to help locate them easily.
Final answer: The sum is 2314.
Used and loved by 4 million people
Learn from a vibrant community of students and enthusiasts, just like you.
Your list looks good! Here's a clean version and a few suggestions.
1. Make sure each step in the calculation is clearly displayed. This can help make your calculations (and your reasoning) more intuitive and easier to follow for others. For example:
Addition process:
Sum = 76 + 90 + 92 + 20 + 99 + 66 + 26 + 11 + 71 + 79 + 71 + 82 + 79 + 94

Observation 1020

E.2 WIKIPEDIA CONTINUATION EXAMPLE

To better understand Llama-3.1-8B-Instruct's behavior after Markovian training to predict Wikipedia text, let's examine a its behavior on the (randomly selected near the end of training) batch number 12500. The model was given the following Wikipedia article excerpt:

> Boones Mill is a town in Franklin County, Virginia, United States. The population was 239 in 2018, down from 285 at the 2000 census. It is part of the Roanoke Metropolitan Statistical Area.
> History
> Boones Mill was incorporated in 1927. It was previously known as "Boone Mill" and "Boon Mill". The town is named after Jacob Boon who operated a mill in the town.
> The Boones Mill Norfolk & Western Railway Depot is listed on the National Register of Historic Places. https://www.townofboonesmill.org/history
> Geography
> Boones Mill is located in northern Franklin County at (37.115462, -79.953966), along U.S. Route 220 at the southern base of the Blue Ridge Mountains. US 220 leads north to Roanoke and south to Rocky Mount, the Franklin County seat.
> According to the United States Census Bureau, the town has

Given this context ending with "According to the United States Census Bureau, the town has", we can compare the CoT generated by our trained versus an untrained model:

E.2.1 COT AFTER TRAINING:

> "The town of Boones Mill is located in Franklin County, Virginia, United States. US 220 leads north to Roanoke and south to Rocky Mount, the Franklin County seat. According to the United States Census Bureau, the town has"

E.2.2 COT BEFORE TRAINING:

> "The town of Boones Mill is a small, rural community with a rich history and natural beauty. The town is surrounded by the Blue Ridge Mountains, offering scenic views and outdoor recreational opportunities. The town's economy is primarily based on agriculture and small"

E.2.3 ACTUAL CONTINUATION:

> "a total area of , all of it land. The town is in the valley of Maggodee Creek, a southeast-flowing tributary of the Blackwater River, part of the Roanoke River watershed. Murray Knob, elevation , rises to the north on the crest of the Blue Ridge, and the eastern end of Cahas Mountain, at , is 2 miles to the west."

The trained CoT shows notably different characteristics from the untrained one. The trained CoT essentially copied the first and last two sentences from the context, making sure to line up the number of allotted tokens with the end of the last sentence. The untrained model seems to give fairly generic properties that the actual Boones Mill Wikipedia article does not mention, such as Boones Mill having an economy primarily based on agriculture. Also, the untrained CoT is not taking the token limit into account and is setting the evaluator model to be surprised when it glues the CoT to the answer and has to predict "agriculture and small a total area of , all of it land".

This example achieved a normalized reward of 0.3438 (in log probability), suggesting that the trained CoT strategy was indeed helpful for predicting the technical geographic description that followed.

F HYPERPARAMETER TUNING AND EXPERIMENTAL CONFIGURATIONS

Our Wikipedia continuation experiments systematically explored the hyperparameter space across multiple model architectures and training configurations. Table 2 provides a comprehensive overview of the hyperparameter settings used in our experiments, extracted directly from the training logs.

The experimental design explored several key dimensions:

Model Architecture: We evaluated four different language models (Llama, Mistral, Phi, and Qwen3) to assess the generalizability of our Markovian training approach across different architectures and parameter scales.

Temperature Scaling: We systematically varied the sampling temperature (1.2, 1.3, 1.4) to study the effect of generation diversity on Markovian training effectiveness. Higher temperatures encourage more diverse CoT generation, potentially leading to more robust reasoning patterns.

Markovian vs Non-Markovian Training: For each model and temperature combination, we conducted paired experiments comparing Markovian training (Markov=Y) versus Non-Markovian training (Markov=N) to isolate the effects of our approach.

Batch Size Optimization: Batch sizes were tailored to each model's memory requirements and computational efficiency, ranging from 6 (Mistral, Llama) to 16 (Phi) based on GPU memory constraints and convergence characteristics.

Training Duration: We used two training regimes - shorter runs (10,000 batches) for initial exploration and longer runs (100,000 batches) for comprehensive evaluation. The shorter runs allowed rapid iteration during hyperparameter search, while longer runs provided robust performance estimates.

The exponential moving average parameter r (0.9) is only used in non-parallel mode for computing historical baseline values; parallel (GRPO) mode uses batch-wise standardization instead. The CoT length was fixed at 75 tokens to ensure consistent computational overhead across experiments. Detailed model and dataset specifications are provided in the Reproducibility Statement below.

The systematic exploration of this hyperparameter space enabled robust evaluation of our Markovian training approach and provided confidence in the generalizability of our results across different model architectures and training configurations.

G IMPACT STATEMENT

Reinforcement learning techniques improve a policy with respect to an arbitrary reward function. But it can be difficult to mathematically specify nuanced human preferences about the policy. Both reinforcement learning from human feedback (RLHF) (Christiano et al., 2023) and Constitutional

Table 2: Hyperparameter configurations for Wikipedia continuation experiments with actual training duration. Experiments use either GRPO optimization (Parallel=Y) or standard policy gradient (Parallel=N) with LoRA fine-tuning. The exponential moving average parameter r is only used in non-parallel mode for baseline computation. 'Actual Lines' shows the number of log entries, indicating actual training progress.

Model	Temp	Batch	LR	Planned	Actual Lines	KL	r	Parallel	Markov	LoRA	CoT Len
llama	1.2	6	1.0e-04	100,000	257	0.1	0.9	N	Y	8/16	75
llama	1.2	6	1.0e-04	100,000	3,973	0.1	–	Y	N	8/16	75
llama	1.3	8	1.0e-04	100,000	8,240	0.1	–	Y	Y	8/16	75
mistral	1.3	10	1.0e-04	100,000	1,064	0.1	0.9	N	Y	8/16	75
mistral	1.4	6	1.0e-04	10,000	9,768	0.1	–	Y	Y	8/16	75
mistral	1.4	6	1.0e-04	10,000	4,151	0.1	–	Y	N	8/16	75
phi	1.3	16	1.0e-04	100,000	656	0.1	0.9	N	Y	8/16	75
phi	1.4	16	1.0e-04	10,000	5,796	0.1	–	Y	Y	8/16	75
phi	1.4	16	1.0e-04	10,000	5,123	0.1	–	Y	N	8/16	75
qwen3	1.3	12	1.0e-04	100,000	780	0.1	0.9	N	Y	8/16	75
qwen3	1.4	12	1.0e-04	10,000	3,543	0.1	–	Y	Y	8/16	75
qwen3	1.4	12	1.0e-04	10,000	3,235	0.1	–	Y	N	8/16	75

AI (Bai et al., 2022) help people specify and optimize the properties they would like the AI to have. This increase in controllability makes the AI more of an extension of human intention, for better or for worse. The approach of this paper is much more targeted – we use RL to specifically increase an agent foresight – its ability to predict its future observations.

On its face, this seems like it might be just as dependent on human intentions as RLHF and Constitutional AI – if an LM is more knowledgeable, maybe it could use that extra knowledge to deceive others, for instance. However, better foresight may also give rise to better values, where values are opinions about how to act such that the collective system can attain better foresight.

H Reproducibility Statement

To ensure reproducibility, we provide comprehensive supplementary materials including all source code, training and evaluation scripts, and detailed instructions in the README. The main training loop (`src/train.py`) supports (i) GRPO, EI, PG, and PPO-style clipped objective methods (see Section F for detailed algorithm descriptions) and (ii) all experimental datasets. We measure fragility of CoT via `src/perturbation_analysis.py` and we estimate interpretability of CoT generations via `src/evaluate_cross_model.py`. The `LatexFolder/` directory contains all paper figures, full training logs, and perturbation evaluation logs from our experiments.

Complete hyperparameter configurations for all Wikipedia continuation experiments are provided in Table 2.

Models: We support 11 language model architectures with full tokenization and formatting: Llama 3.1 8B Instruct, Llama 3.2 1B Instruct, Mistral 7B Instruct V0.2, GPT-2 (124M), TinyStories (33M), Phi 3.5 Mini Instruct, Phi-4, Qwen3 4B, Qwen3 14B, Gemma-3 2B, and Gemma-3 Small (9B). All models use public HuggingFace implementations with LoRA fine-tuning.

Datasets: We support 6 task types: (1) *arithmetic* - randomly generated 15-term addition problems, (2) *arithmetic-negative* - addition with negative numbers, (3) *gsm8k* - grade school math word problems from Cobbe et al. (2021), (4) *mmlu* - multiple choice questions from the Massive Multitask Language Understanding benchmark, (5) *wiki_compression* - predicting compressed Wikipedia text, and (6) *wiki_continuation* - next-token prediction on Wikipedia articles. Environment setup instructions are provided in the README.

Our experiments were conducted on NVIDIA H100 GPUs through the RunPod cloud service. Each training run took approximately 5 hours on a single H100 GPU. For the arithmetic task experiments (Figure 4), we performed 4 independent runs for each algorithm to account for high variance. For the Wikipedia continuation experiments (Table 2), we performed single runs but explored many different hyperparameter configurations across models, temperatures, and training modes. Since we explored many different training algorithms (GRPO, PPO-style clipped objective, policy gradient, and expert iteration) across multiple datasets and hyperparameter settings, the total compute for

our final reported experiments was approximately 10,000 GPU-hours. The full research project, including preliminary experiments with approaches that didn't make it into the final paper, consumed significantly more compute - approximately $32,000 worth of cloud compute resources. This information is provided in our Reproducibility Statement to help researchers understand the resources needed to reproduce our results.

With these materials, researchers should be able to reproduce our work, including the performance boost on GSM8K and the perturbation analysis results demonstrating CoT reliance.