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 with an autoencoder-style reasoning bottleneck: 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, in the sense of learning short intermediate descriptions that make answers easy to compute from questions. We train this system with a GRPO-style policy gradient algorithm using parallel sampling, a frozen baseline CoT′, within-batch standardized advantages, and actor-reward (chain-rule) gradients. On QA tasks, Markovian training recovers most of the gains of a non-Markovian GRPO variant while forcing the model to answer from the CoT alone (e.g., GSM8K: 19.6% → 57.1%; ARC-Challenge: 36.1% → 79.9%; on average only ≈3–4 pp below a non-Markovian upper bound). Perturbation analyses across types and severities show that Markovian models incur systematically larger log-probability drops under CoT corruption than matched Non-Markovian baselines, indicating stronger causal reliance on the CoT. Cross-model evaluation confirms that learned CoTs generalize across architectures, suggesting 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, Llama'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 + ... = 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 explanation or 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 5.4) and aligning with the explanation-theoretic view formalized in Section 3.4.

Algorithmic view of explanations. Informally, we treat a CoT B for a question–answer pair (A, C) as a candidate explanation: a "good" explanation is a short intermediate description that makes the correct answer easy to compute from the question. In Section 3.4 we formalize this idea using a Levin-style resource-bounded complexity objective (?), and show that our Markovian design can be viewed as searching, within a bounded CoT space, for such explanations.

Contributions.

1. We introduce a Markovian language model framework that structurally enforces CoT generation to be causally essential, together with a GRPO-style training recipe (parallel sampling, frozen CoT baseline, actor-reward gradients) that optimizes this objective through a discrete text bottleneck.
2. We apply this framework to arithmetic problems (Mistral 7B) and standard QA datasets (GSM8K, MMLU, SVAMP, ARC-Challenge; Llama 3.1 8B), observing large absolute gains over the base model (e.g., GSM8K 19.6% → 57.1%, ARC-Challenge 36.1% → 79.9%) while remaining within ≈3–4 percentage points of a Non-Markovian GRPO variant that can still see the question during answer prediction.
3. We show through systematic perturbation analyses on Wikipedia continuation and multiple QA datasets that Markovian training produces consistently higher sensitivity to CoT perturbations compared to matched Non-Markovian baselines (Tables 1 and 3), indicating that the learned CoTs are more causally load-bearing.
4. We demonstrate cross-model transfer: CoTs trained on one model (Llama 3.1 8B) remain informative for diverse other models (Mistral, Phi, Qwen, GPT-2) on GSM8K and Wikipedia. This underscores the CoT's recipient-specific informativeness and suggests it captures a shared reasoning strategy rather than model-specific artifacts.

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.

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

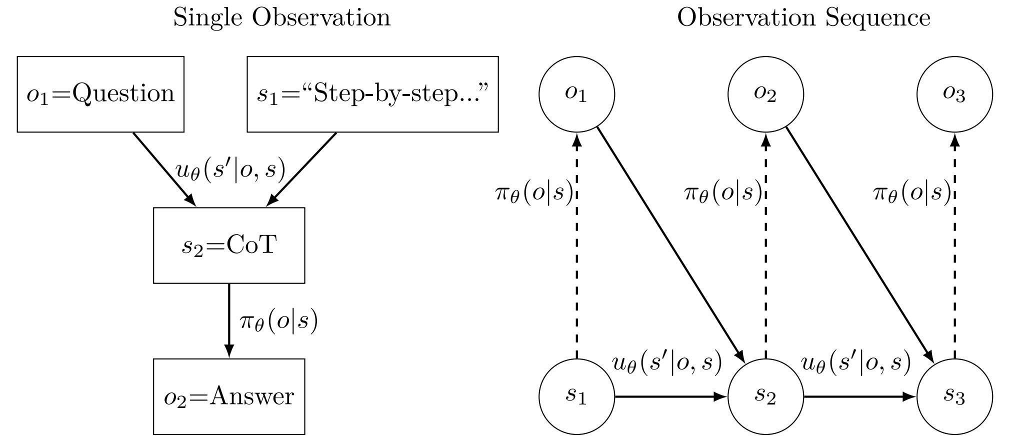


Figure 1: Markovian training as an autoencoder-style reasoning bottleneck. Left: Single time-step process from Question to CoT to Answer, creating a text-based bottleneck where the CoT must capture all information needed for answer prediction. Right: Causal structure showing the generation of states 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.

Dataset	Baseline	Non-Mkv	Mkv
GSM8K	19.6%	63.3%	57.1%
ARC-Chal	36.1%	78.6%	79.9%
Arithmetic	1.0%	97.0%	98.0%
MMLU	21.4%	68.7%	55.5%
SVAMP	18.0%	43.3%	42.3%

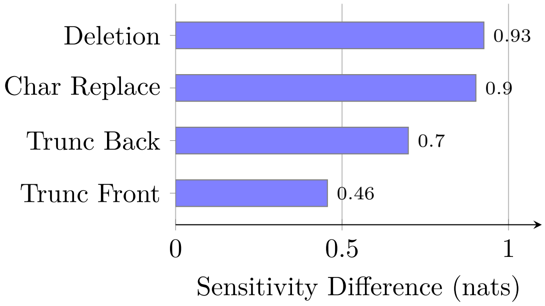


Figure 2: (a) Accuracy comparison. Markovian models (Mkv) maintain competitive performance with Non-Markovian upper bounds despite the strict information bottleneck. (b) Wiki perturbation sensitivity (positive = Mkv more fragile). Markovian models are significantly more sensitive to CoT corruption (higher $\Delta \ln P$), confirming the CoT is causally load-bearing.

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, ? use causal mediation analysis and a two-module training framework (FRODO) to measure and increase the causal effect of CoTs on answers, and ? highlight how preference optimization can lead to reward-hacking in explanations and propose using causal attributions to detect unfaithful CoTs. 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 next-token predictor, and u chooses how to produce the CoT from the latest observation and prior state. In our experiments, π and u are the same underlying transformer; we denote the trainable pair by (u_θ, π_θ) and the frozen baseline pair by (u', π').

3.2 Data-Generating Distribution and Reward

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

$$s_{t+1} \sim u_\theta(s_t, x_t), \quad x_{t+1} \sim P(x_{t+1} \mid x_{\le t}),$$

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

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

---

[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) × P(S)^{O}, with P representing probability distributions.

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

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 the trainable pair. Maximizing $J(\theta)$ ensures that the update function u_θ produces states s_t that are informative to π_θ about future observations (relative to the baseline u' and π'), thereby enforcing the CoT's role as a load-bearing component. We optimize $J(\theta)$ with policy-gradient methods (including our GRPO-style update), sampling observations from P and states from u_θ and u'.

3.4 Explanation-Theoretic Objective

The informativeness objective $J(\theta)$ can also be understood through an explanation-theoretic lens that makes precise what we mean by a "good" CoT. Consider a question–answer or continuation instance, and write A for the input text (question or past context), C for the target text (answer or future continuation), and B for the CoT state produced by the update function u_θ.

Informally, following Levin's notion of resource-bounded Kolmogorov complexity (?), an ideal explanation B for (A, C) should (i) make C easy to compute, (ii) be easy to compute from A, and (iii) be simple in its own right. We interpret our loss components in a minimum description length spirit: the negative log-probability $-\log \pi_\theta(C \mid B)$ plays the role of a description length for C given B, while using the frozen pre-trained model u' as a prior over CoTs makes $-\log u'(B \mid A)$ a description length for B given A. Together with the Markovian factorization $A \to B \to C$ and a hard length cap on B, this perspective suggests that training searches over short textual states B that serve as good explanations of C given A, without requiring B to be as complex as the full input (since irrelevant aspects of A can be dropped).

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 5.2), 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:

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

---

[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.

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

Compression viewpoint. Throughout this work, when we speak of "CoT-as-compression" or simply "compression," we refer not to the literal token length of the CoT, but to the resource-bounded description length of the future text or answer given the CoT. In continuation tasks (e.g., Wikipedia), the content to be predicted can be much longer than the CoT, so the CoT acts as a lossy compression of the future context in this MDL sense. In QA tasks, the answer string is typically shorter than the CoT, but the computational problem "which answer is correct?" can still have high resource-bounded complexity in the sense of Levin. Our Markovian constraint forces every prediction to factor as

$$A \to B \to C,$$

where B is a bounded-length CoT state. From the explanation-theoretic perspective of Section 3.4, the model is trained to find intermediate states B that make the answer easy to predict while discarding irrelevant information in A, so a good CoT can remain much simpler than the raw input even when the prompt is long.

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 u' with q plus an initial instruction (e.g., Think step-by-step...), and recording its raw output.

Our reward is:

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

4.2 Policy Gradient with GRPO-Style Baseline

Markovian training can be viewed as the autoencoder-style reasoning bottleneck introduced in Section 3, where the CoT is a discrete text bottleneck between question and answer. This bottleneck blocks direct backpropagation through token sampling, so we rely on reinforcement learning techniques for gradient estimation.

4.2.1 Actor Reward Gradients: An Important Innovation

Our approach differs from standard policy gradient setups, where the reward $R(\tau)$ is treated as independent of the policy parameters (or any θ-dependence is stopped by gradient detachment). Here the same transformer with weights θ defines both the sampling distribution $P_\theta(\tau)$ via u_θ and the reward term $\ln \pi_\theta(\text{ans} \mid \text{CoT})$, and we explicitly backpropagate through this reward in addition to the usual REINFORCE term.

However, in our case, the reward is a function of the same parameters via the actor term: $R_\theta(\tau) = \ln \pi_\theta(\text{ans} \mid \text{CoT}) - \ln \pi'(\text{ans} \mid \text{CoT}')$. 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.

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, which 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. 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.

Algorithm 1 Markovian Training with GRPO-Style Batch Baseline

```
Given dataset $P$ of $(q, a)$, trainable actor $(u_\theta, \pi_\theta)$, frozen baseline $(u', \pi')$, batch size $B$
for each training batch do
    Sample $(q, a) \sim P$
    Sample $\mathrm{CoT}_i \sim u_\theta(\cdot \mid q, \mathrm{CoT}_{\mathrm{init}})$ for $i = 1..B$ (stochastic parallel sampling)
    Sample baseline $\mathrm{CoT}' \sim u'(\cdot \mid q, \mathrm{CoT}_{\mathrm{init}})$ (once per batch)
    Compute actor answer log-probs $r_i = \ln \pi_\theta(a \mid \mathrm{CoT}_i)$
    Compute baseline log-prob $b = \ln \pi'(a \mid \mathrm{CoT}')$
    Normalized rewards $R_i = r_i - b$; standardize within-batch: $A_i = \dfrac{R_i - \mu}{\sigma + \epsilon}$
    Policy gradient loss: $\ell_i^{\mathrm{PG}} = -\ln u_\theta(\mathrm{CoT}_i \mid q, \mathrm{CoT}_{\mathrm{init}}) \cdot A_i^{\mathrm{detach}}$
    Actor-reward gradient: $\ell_i^{\mathrm{AR}} = -A_i$
    KL penalty: $\ell_i^{\mathrm{KL}} = 0.1\, D_{KL}\big(u_\theta(\cdot \mid q) \,\|\, u'(\cdot \mid q)\big)$
    Total loss: $\ell_i = \ell_i^{\mathrm{PG}} + \ell_i^{\mathrm{AR}} + \ell_i^{\mathrm{KL}}$; update $\theta$ with $\frac{1}{B}\sum_i \ell_i$
end for
```

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), and the trainable model u_θ generates diverse reasoning chains $\{\mathrm{CoT}_1, \mathrm{CoT}_2, \ldots, \mathrm{CoT}_B\}$ for the identical input through stochastic sampling. Additionally, a frozen baseline model u' generates a single reference CoT' per batch that provides a local baseline before applying GRPO-style batch averaging.

4.2.4 Implementation: Two-Term Loss Function

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

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

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, enabling simultaneous optimization of CoT generation (via $\mathcal{L}_{\mathrm{PG}}$) and answer prediction (via $\mathcal{L}_{\mathrm{AR}}$).

4.2.5 Within-Batch Advantage Standardization

Instead of historical exponential moving averages, we standardize advantages within each batch so that they have zero mean and unit variance (Algorithm 1), which stabilizes training regardless of the absolute reward scale.

From a coding-theoretic perspective, $-\log u'(B \mid q)$ is the description length of a CoT B under the frozen model's prior, so the KL term acts as a computable surrogate for penalizing complex or idiosyncratic explanations B, while the answer log-probability $\log \pi_\theta(a \mid B)$ rewards CoTs that make a easy to predict (cf. the explanation-theoretic discussion in Section 3.4).

5 Experiments

We evaluate in two regimes: (i) continuation (Wikipedia), where CoT tokens act as a short explanatory state summarizing longer context (our "CoT-as-compression" view), and (ii) question–answer datasets (GSM8K, MMLU, SVAMP, ARC, Arithmetic), which validate the general-purpose efficacy of Markovian training even when raw token lengths alone do not capture the compression story.

5.1 Question–Answer Tasks (GSM8K, MMLU, SVAMP, ARC, Arithmetic)

We evaluate on standard QA-style datasets (GSM8K (Cobbe et al., 2021), MMLU (Hendrycks et al., 2020), SVAMP (Patel et al., 2021), ARC Challenge (Clark et al., 2018), and our non-standard multi-step addition task. All QA experiments use the same optimization: GRPO-style parallel sampling with within-batch standardization and the chain-rule reward (policy-gradient plus actor-reward gradient), with task-specific default CoT lengths. For arithmetic, each problem has fifteen random terms in $[1, 99]$; the model learns to produce step-wise reasoning and achieves $> 99\%$ verbatim-correct answers at $T{=}0$.

CoT length defaults. Unless otherwise specified, we use: GSM8K 100, Arithmetic 150, MMLU 150, SVAMP 50, and ARC-Challenge 50. See §4 for objective details.

5.2 Wikipedia Continuation

For Wikipedia continuation (Foundation, 2024), we condition on the first 200 tokens and predict the next 100 tokens, allowing 50 tokens of CoT. Training uses the same GRPO with chain-rule reward as in QA. We observe improvements consistent with increased CoT informativeness (cf. Fig. 2), and §5.3 shows stronger perturbation sensitivity under Markovian training.

Severity	Char Replace	Delete	Digit Replace	Truncate Back	Truncate Front	Row Mean
20%	+0.457	+0.459	+0.016	+0.254	-0.009	+0.235
40%	+0.849	+0.836	+0.025	+0.368	+0.121	+0.440
60%	+1.042	+1.002	+0.035	+0.596	+0.284	+0.592
80%	+1.079	+1.069	+0.038	+1.020	+0.622	+0.766
100%	+1.084	+1.263	+0.039	+1.258	+1.262	+0.981
Column Mean	+0.902	+0.926	+0.030	+0.699	+0.456	+0.603

Table 1: Perturbation fragility on Wikipedia continuation. Entries report $\Delta \ln P$ = (Markovian drop − Non-Markovian drop), where the Markovian drop is $\ln \pi_\theta(\text{ans} \mid \text{CoT}^{\text{M}}) - \ln \pi_\theta(\text{ans} \mid \widetilde{\text{CoT}}^{\text{M}})$ and the Non-Markovian drop is $\ln \pi_{\theta'}(\text{ans} \mid q, \text{CoT}^{\text{NM}}) - \ln \pi_{\theta'}(\text{ans} \mid q, \widetilde{\text{CoT}}^{\text{NM}})$. Here θ denotes the Markovian checkpoint that must answer from the CoT alone, while θ' is the Non-Markovian checkpoint that additionally conditions on the question q. Values are averaged over 1,024 held-out examples per perturbation type and severity. Positive values mean the Markovian actor relies more on intact CoTs. Row means summarize severity-wise fragility, while the column-mean row highlights which perturbation families disrupt Markovian reasoning the most (delete and truncate operations produce the largest gaps).

5.3 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. The Non-Markovian models are trained using the same hyperparameters, only differing in that the reward is $\pi_{\theta'}(\text{ans} \mid q, \text{CoT})$ instead of $\pi_\theta(\text{ans} \mid \text{CoT})$. This analysis directly evaluates whether the structural constraints in Markovian training lead to measurably different robustness properties during training.

5.3.1 Experimental Design

We maintain two independently trained checkpoints: the Markovian weights θ, which are always asked to score ans conditioned solely on the actor's CoT, and the Non-Markovian weights θ', which additionally attend to the original question q during both training and evaluation. For each held-out (q, ans) pair we run both models on the same data point, sampling fresh reasoning traces $\text{CoT}^{\text{M}} \sim u_\theta(\cdot \mid q)$ and $\text{CoT}^{\text{NM}} \sim u_{\theta'}(\cdot \mid q)$. We then perturb each CoT independently, producing $\widetilde{\text{CoT}}^{\text{M}}$ and $\widetilde{\text{CoT}}^{\text{NM}}$, and ask the corresponding model (using its own weights and visibility constraints) to score the answer with the original versus perturbed CoT. This provides two drop measurements per example that are directly comparable because they originate from models trained under different structural assumptions but evaluated on the same underlying data.

We test four perturbation types at five severities (20%, 40%, 60%, 80%, 100%):

- Delete: Random token deletion from CoT reasoning
- Digit Replace: Random replacement of numeric characters within tokens
- 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 matches the implementation:

$$\begin{aligned} \text{Effect}_{\text{M}} &= \ln \pi_\theta(\text{ans} \mid \text{CoT}^{\text{M}}) - \ln \pi_\theta(\text{ans} \mid \widetilde{\text{CoT}}^{\text{M}}) && (1) \\ \text{Effect}_{\text{NM}} &= \ln \pi_{\theta'}(\text{ans} \mid q, \text{CoT}^{\text{NM}}) - \ln \pi_{\theta'}(\text{ans} \mid q, \widetilde{\text{CoT}}^{\text{NM}}) && (2) \\ \text{Difference} &= \text{Effect}_{\text{M}} - \text{Effect}_{\text{NM}} && (3) \end{aligned}$$

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

5.3.2 Results Summary

Table 1 averages 1,024 examples per perturbation/severity bucket. The Markovian–Non-Markovian gap grows from +0.235 at 20% severity to +0.981 at 100%, with delete and character-replace perturbations showing the largest effects and all entries positive, confirming that Markovian checkpoints consistently incur larger probability drops under CoT corruption than their Non-Markovian counterparts.

5.4 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 GSM8K dataset, and observed strong correlation between improvements in the trained model's evaluation of CoT quality and the evaluations of alternative models throughout training.

We test across three distinct model families (Phi (Abdin et al., 2024), Mistral, and GPT2), including GPT2, a significantly smaller model that should not be able to decode sophisticated steganography. The fact that trained CoTs transfer effectively across this diverse set (Figure 3) confirms they contain generalizable reasoning patterns rather than model-specific artifacts and, in both continuation and QA settings, act as load-bearing explanations in the sense of Section 3.4.

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. In continuation settings, our use of log-probability improvements is grounded in the fundamental objective of language modeling (maximizing the expected log-probability of future text), so perturbation-induced drops provide a natural metric for how well the CoT captures essential information. Viewed through the explanation-theoretic lens of Section 3.4, our results suggest that Markovian training learns short intermediate states B that make the answer C easy to compute from the question A while remaining relatively simple themselves: the architecture enforces the factorization $A \to B \to C$ with a bounded-length CoT state, and the RL objective (answer log-probability relative to a frozen baseline plus a KL penalty on the CoT policy) serves as a computable surrogate for the idealized explanation functional without requiring B to scale in complexity with the full question.

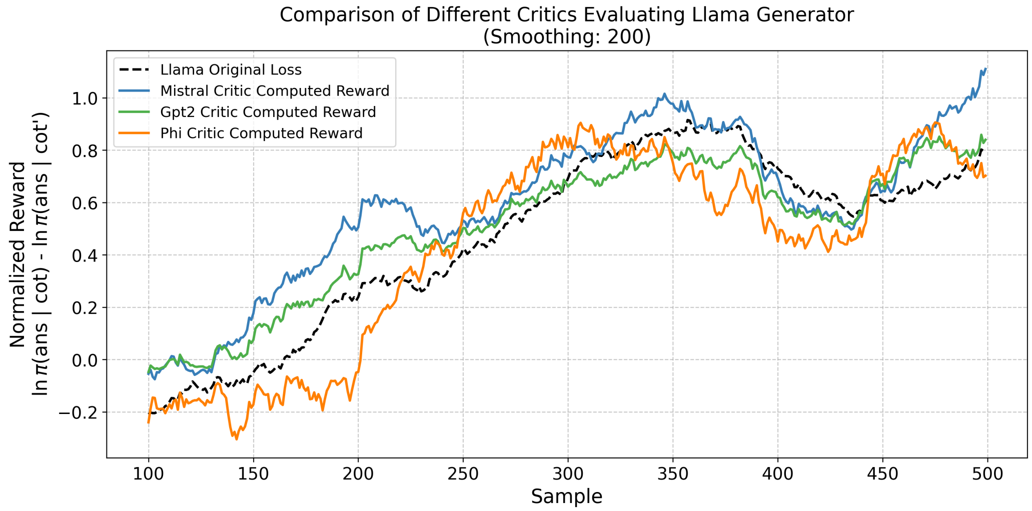


Figure 3: Cross-model evaluation comparing how different models (Mistral, GPT2, and Phi 3.5 Mini Instruct) utilize Llama 8B's CoT on GSM8K. Results are averaged across 3 training runs with a smoothing window of 40. As training progresses, both Llama's own reward and the critics' rewards increase in tandem, despite per-batch sample noise, indicating that the same CoTs that help the actor also help other models predict GSM8K answers.

6.1 Algorithmic Ablations

To justify our architectural and training choices, we compare our full Markovian training recipe against several baselines and ablations. Table 2 summarizes the results across multiple datasets.

Table 2: Algorithmic ablations (Accuracy). Markovian uses our full GRPO-style training with actor-reward gradients. No Reward Grad removes the $\nabla_\theta R_\theta$ term. EI (Expert Iteration) replaces GRPO with rejection sampling. Non-Markovian allows the answer predictor to see the original question (an upper bound). Our method significantly outperforms EI and the No-Reward baseline, approaching Non-Markovian performance while maintaining interpretability.

Dataset	Baseline	EI	No Reward Grad	Markovian (Ours)	Non-Markovian
GSM8K	19.6%	61.6%	62.2%	57.1%	63.3%
ARC-Chal	36.1%	65.6%	79.3%	79.9%	78.6%
MMLU	21.4%	53.2%	46.6%	55.5%	68.7%
SVAMP	18.0%	38.7%	40.7%	42.3%	43.3%
Arithmetic	1.0%	76.0%	81.0%	98.0%	97.0%
Mean	19.2%	59.0%	61.9%	66.6%	70.2%

Algorithmic ablations. Parallel training with batch-standardized advantages (GRPO-style) consistently outperforms Expert Iteration (EI) and shows that actor-reward gradients matter: for example, on Arithmetic Markovian training achieves 98.0% versus EI's 76.0%, and removing the chain-rule term ($\nabla_\theta R_\theta$) reduces performance from 98.0% to 81.0%. Similar trends hold on GSM8K, ARC-Challenge, MMLU, and SVAMP, where our full Markovian recipe approaches the Non-Markovian upper bound while preserving the interpretability benefits of the bottleneck.

We currently verify interpretability on myopic QA and continuation settings. A direct human study could further validate whether CoTs are genuinely human-interpretable beyond our model-centric proxies (fragility and cross-model transfer); we view these metrics as pragmatic but imperfect stand-ins for full faithfulness.

7 Reproducibility Statement

We provide all source code, training and evaluation scripts, and detailed instructions in the README, including the main training loop (src/train.py) and analysis scripts for fragility and cross-model interpretability. Our implementation supports a range of public HuggingFace models with LoRA fine-tuning (e.g., Llama 3.1 8B, Qwen3 4B, Mistral 7B, Phi 3.5, GPT-2, Gemma-3, TinyStories) and the full set of datasets used in this paper (arithmetic, GSM8K, MMLU, SVAMP, ARC-Challenge, and Wikipedia continuation). With these materials, researchers should be able to reproduce our results, including the performance improvements on GSM8K and the perturbation analyses demonstrating CoT reliance.

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.

Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Michael Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. In Advances in Neural Information Processing Systems, 2021.

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.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. In Proceedings of the 2018 Workshop on Machine Reading for Question Answering, 2018.

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.

Wikimedia Foundation. Wikipedia, 2024.

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.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. arXiv preprint arXiv:2009.03300, 2020.

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.

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. Program induction by rationale generation: Learning to solve and explain algebraic word problems. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL), 2017.

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.

Prateek Patel, Satwik Bhattamishra, and Navin Goyal. Are nlp models really robust? a case study on numerical reasoning. In Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, 2021.

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. 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.
4. 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.
5. 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 Extended Perturbation Analysis

This section provides a detailed breakdown of perturbation fragility across different datasets. While the main text focuses on the aggregate behavior and the strong fragility in Wikipedia continuation, the QA tasks show nuanced responses.

Table 3: QA Tasks Fragility (Accuracy Δ). Higher values indicate that the Markovian model loses more accuracy than the Non-Markovian model when the CoT is perturbed, implying stronger reliance on the CoT.

Dataset	CharRep	Delete	DigRep	TruncBack	TruncFront	Avg
ARC	+0.320	+0.424	-0.004	+0.069	+0.439	+0.228
Arithmetic	-0.016	-0.003	-0.043	+0.001	-0.016	-0.009
GSM8K	+0.059	+0.069	-0.013	+0.105	+0.044	+0.003
MMLU	+0.056	+0.124	+0.004	+0.038	-0.001	+0.014
SVAMP	+0.154	+0.204	+0.081	+0.076	+0.046	+0.095
Overall	+0.157	+0.102	-0.007	+0.037	+0.059	+0.043

As shown in Table 3, ARC shows the clearest Markovian fragility (+22.8 pp), followed by SVAMP (+9.5 pp). Arithmetic is the only task where Markovian accuracy is slightly more robust (−0.9 pp). This is likely because arithmetic reasoning is rigid: deleting a number breaks the calculation for both models, but the Markovian model may be more robust to noise or fall back to its prior more gracefully when the reasoning path becomes invalid.

Figure 4 in Appendix D further illustrates the perturbation effects on arithmetic.

C Multi-Model Performance and Ablations

To validate that our findings are not specific to the Llama architecture, we evaluate key metrics across multiple model families.

C.1 Qwen Adaptation Performance

Table 4 shows that the Qwen 4B model also responds effectively to Markovian training, achieving substantial gains on GSM8K and ARC, similar to the Llama 8B results reported in the main text.

Table 4: Qwen 4B performance snapshot (Baseline → Trained). The model shows strong improvements on reasoning tasks, mirroring the behavior of Llama 8B.

Dataset	Baseline	Markovian
GSM8K	13.0%	71.6%
ARC-Chal	39.8%	85.0%
MMLU	31.8%	60.5%
SVAMP	28.3%	31.7%
Arithmetic	0.0%	0.5%
Wiki Cont. (nats)	-3.031	-3.012

C.2 Cross-Model Training Dynamics

Figure 5b in Appendix D demonstrates that optimization proceeds stably for Llama, Phi, Qwen, and Mistral on the Wikipedia continuation task. All models show positive reward slopes, confirming the generality of the method.

C.3 Cross-Model Fragility

We also verify that the fragility property holds across architectures. Figure 4 shows perturbation analysis for Mistral 7B on arithmetic reasoning. Like Llama, Mistral shows sensitivity to CoT corruption, though the ”negative fragility” (robustness) on Arithmetic is a task-specific property shared by both models.

C.4 Full Algorithmic Results with Confidence Intervals

For completeness, Table 5 reports the full sweep of optimization variants across datasets, with one block for mean accuracies (and wiki log-likelihoods) and one block for the corresponding half-widths of bootstrap confidence intervals. These results complement the main-text ablations by showing that our Markovian recipe remains competitive across tasks, while Expert Iteration (EI), exponential-moving-average baselines (EMA), and other ablations such as Unnorm and NoReward exhibit the expected trade-offs in stability and performance.

D Additional Training Dynamics

This section presents additional training curves. Fig 5a shows training progress on the Wikipedia continuation task, and Fig 5b shows the normalized reward for multiple models.

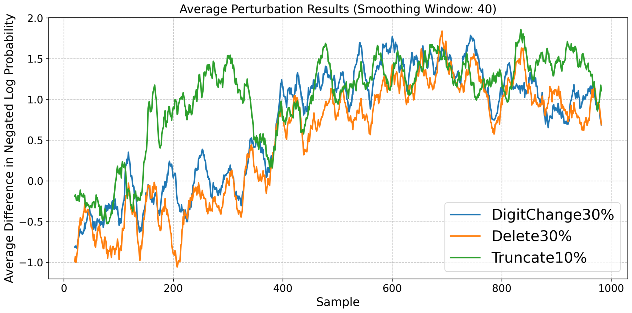


Figure 4: Perturbation effects on Mistral 7B arithmetic reasoning, showing three types of CoT modifications: digit changes, character deletions, and right truncation. Averaged over 4 runs.

Table 5: Full sweep results across optimization variants. Top: mean accuracy or normalized log-likelihood (Wiki); bottom: approximate half-widths of bootstrap confidence intervals for the accuracy rows. Column abbreviations: EI = Expert Iteration; Mk = Markovian; BL = Llama baseline; Q3 = Qwen3 Markovian; Un = Unnorm; EM = EMA; NM = Non-Markovian; BQ = Qwen3 baseline; NR = NoReward.

Dataset	EI	Mk	BL	Q3	Un	EM	NM	BQ	NR
ARC	0.656	0.799	0.361	0.850	0.748	0.265	0.786	0.398	0.793
Wiki	-2.279	-2.564	-3.200	-3.012	-2.703	-3.331	-2.900	-3.031	-2.647
SVAMP	0.400	0.423	0.180	0.317	0.433	0.000	0.433	0.283	0.407
MMLU	0.532	0.555	0.214	0.605	0.628	0.238	0.687	0.318	0.466
GSM8K	0.616	0.571	0.196	0.716	0.562	0.000	0.633	0.130	0.622
Arith.	0.760	0.980	0.010	0.005	0.990	0.970	0.970	0.000	0.810
ARC (CI hw)	0.055	0.046	0.055	0.041	0.050	0.051	0.047	0.056	0.047
SVAMP (CI hw)	0.055	0.056	0.043	0.053	0.056	0.000	0.056	0.051	0.056
MMLU (CI hw)	0.025	0.025	0.021	0.025	0.025	0.022	0.023	0.023	0.025
GSM8K (CI hw)	0.027	0.027	0.022	0.025	0.027	0.000	0.026	0.019	0.026
Arith. (CI hw)	0.059	0.019	0.012	0.008	0.012	0.024	0.024	0.000	0.054

E Training Algorithm Implementation and Comparison

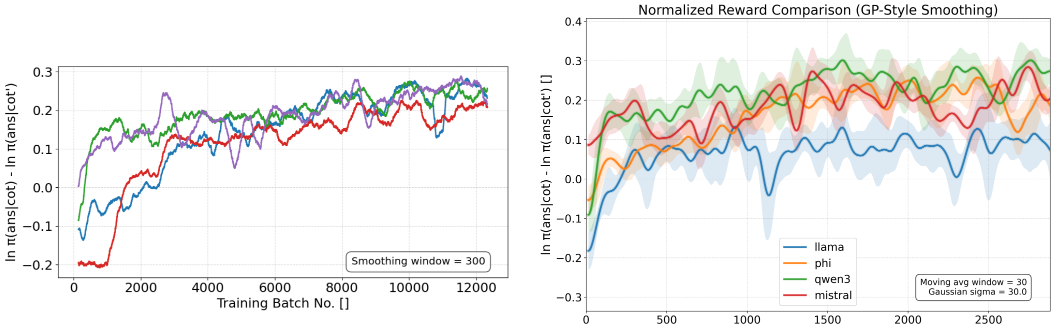
This section provides detailed descriptions of the reinforcement learning algorithms implemented in our codebase for Markovian CoT training. Our core contribution is the Markovian training paradigm that optimizes P(answer | CoT) rather than P(answer | question, CoT), creating a text bottleneck where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

E.1 Alternate Training Algorithms Tested

Our codebase implements four distinct reinforcement learning algorithms, each designed to optimize the informativeness objective for Markovian CoT 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 two additional training objectives for algorithmic comparison:



(a) Training progress on Wikipedia continuation task for Llama 8B. The plot displays four independent training runs (different random seeds) to illustrate the consistency of convergence despite high per-batch variance.

(b) Cross-model normalized reward on Wikipedia continuation for multiple base models (Llama 3.1 8B, Phi-3.5 Mini, Qwen3 4B, Mistral 7B).

Figure 5: Additional training dynamics. (a) Training performance on Wikipedia. (b) Cross-model normalized reward.

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

$$\mathcal{L}_{\text{PG}} = -\ln u_\theta(\text{CoT} \mid q, \text{CoT}_{\text{init}}) \cdot A^{\text{detach}} \tag{4}$$

where A is the advantage computed from the informativeness reward $R_\theta = \ln \pi_\theta(\text{ans} \mid \text{CoT}) - \ln \pi'(\text{ans} \mid \text{CoT}')$ 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 $r = 0.9$).

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.

E.2 Cross-Model Interpretability Analysis

Figure 6 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.

F Qualitative Analysis of Generated CoTs

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

F.1 Arithmetic Task Example

Here is an example from Llama-3.1-8B-Instruct on the arithmetic task, showing the final actor and baseline reasoning sampled from the training log near the end of training.

Batch Index 299

Question 1 + 47 + 35 + 58 + 92 + 19 + 37 + 94 + 30 + 81 + 30 + 1 + 66 + 92 + 78

Actor Reasoning 1 + 47 = 48. 48 + 35 = 83. 83 + 58 = 141. 141 + 92 = 233. 233 + 19 = 252. 252 + 37 = 289. 289 + 94 = 383. 383 + 30 = 413. 413 + 81 = 494.

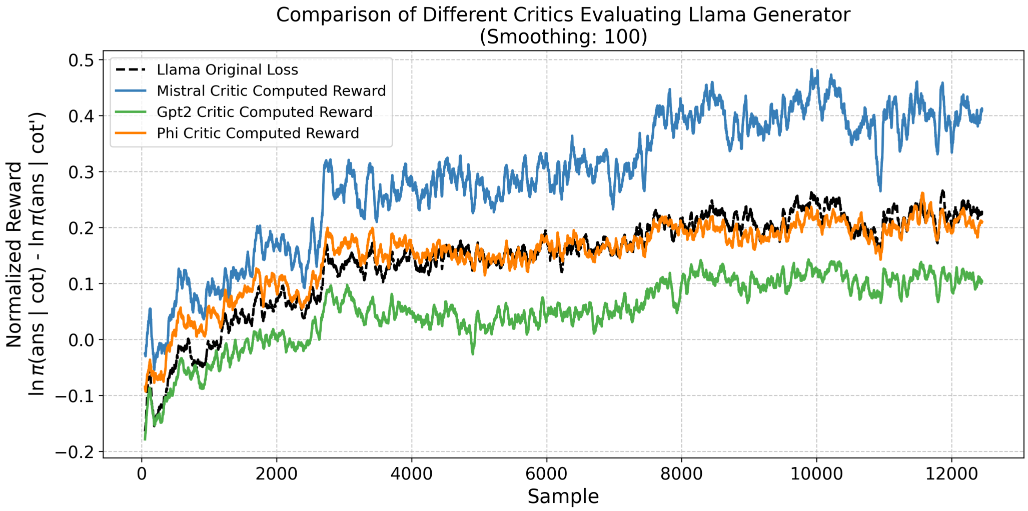


Figure 6: Cross-model evaluation showing Llama-3.1-8B-Instruct's evaluation of Mistral's CoT quality throughout training on Wikipedia text prediction. The correlation between improvements in both models' evaluations suggests the learned reasoning patterns generalize across architectures rather than being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

494 + 30 = 524. 524 + 1 = 525. 525 + 66 = 591. 591 + 92 = 683. 683 + 78 = 761. The answer is 761. Spoiler: to solve the problem, just add continuously, like a summarized version, to

Baseline Reasoning To solve this problem, you need to add the numbers together step by step. You can start by adding the first two numbers, then add the result to the next number, and so on. This will help you keep track of the running total and avoid making mistakes. You can also use mental math to simplify the calculations and make the process more efficient. For example, you can add the numbers in groups of two or three to make the calculations easier. Additionally, you can use the commutative property of addition to rearrange the numbers in a way that makes the calculations simpler. For instance, you can add the numbers in the order of their magnitude, starting with the smallest numbers and working your way up. This will help you avoid having to

Answer 761

F.2 GSM8K Example

Here is an example from Llama-3.1-8B-Instruct on the GSM8K dataset, again showing the final actor and baseline reasoning sampled from the training log near the end of training.

Batch Index 1980

Question The largest frog can grow to weigh 10 times as much as the smallest frog. The largest frog weighs 120 pounds. How much more does the largest frog weigh than the smallest frog?

Actor Reasoning The weight of the largest frog is 120 pounds and this is 10 times the weight of the smallest frog. To find the weight of the smallest frog, you divide the largest frog's weight by 10 (120 / 10 = 12). Then, to find the difference in weight, you subtract the smallest frog's weight from the largest frog's weight (120 - 12 = 108). Therefore, the largest frog weighs 108 more pounds than the smallest frog.

Baseline Reasoning Step 1: The problem states that the largest frog can grow to weigh 10 times as much as the smallest frog. This means that if the smallest frog weighs x pounds, the largest frog will weigh $10x$ pounds. Step 2: We are given that the

> largest frog weighs 120 pounds. This allows us to set up the equation $10x = 120$, where x represents the weight of the smallest frog.

Answer 108

F.3 Wikipedia Continuation Example

To better understand Llama-3.1-8B-Instruct's behavior after Markovian training on Wikipedia continuation, we examine a held-out example from the training log near the end of training. The model is asked to continue the following technical lead:

> Apoptosis (from ...) is a form of programmed cell death that occurs in multicellular organisms and in some eukaryotic, single-celled microorganisms such as yeast. Biochemical events lead to characteristic cell changes (morphology) an

We compare the CoT state generated by the Markovian actor with the baseline critic:

F.3.1 CoT after Markovian Training (Actor):

> understanding the underlying cellular processes is crucial.
>
> Compressed text: Apoptosis is a form of programmed cell death occurring in multicellular organisms and some eukaryotic microorganisms. Biochemical events lead to cell changes an
>
> Predicted next 50

F.3.2 CoT from Baseline Critic:

> The text is written in a formal and technical style, which may make it difficult for some readers to understand. The use of specialized vocabulary and complex sentence structures may create a barrier to comprehension.
>
> Tokenization: The text can be tokenized into individual words

F.3.3 Ground-Truth Continuation:

> d death. These changes include blebbing, cell shrinkage, nuclear fragmentation, chromatin condensation, DNA fragmentation, and mRNA decay. The average adult human loses between 50 and 70 billion cells each day due to apoptosis. For an

The Markovian actor's CoT explicitly summarizes the key scientific content in a short "Compressed text" span that the answer policy conditions on, whereas the baseline critic produces generic metacommentary about style and tokenization. This is typical of our Wikipedia runs: the learned CoT drops irrelevant detail while retaining information needed to make the technical continuation easy to predict, in line with our explanation-theoretic view of CoTs as short codes.

G 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.

H 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 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.