Causal Reward Adjustment: Mitigating Reward Hacking in External Reasoning via Backdoor Correction

Ruike Song[1,2*], **Zeen Song**[1,3*], **Huijie Guo**[1,3], **Wenwen Qiang**[1,3†]
[1]Institute of Software Chinese Academy of Sciences
[2]Nankai University
[3]University of Chinese Academy of Sciences
songruike@mail.nankai.edu.cn, {songzeen@, guohuijie@, qiangwenwen@}iscas.ac.cn

Abstract

External reasoning systems combine language models with process reward models (PRMs) to select high-quality reasoning paths for complex tasks such as mathematical problem solving. However, these systems are prone to reward hacking, where high-scoring but logically incorrect paths are assigned high scores by the PRMs, leading to incorrect answers. From a causal inference perspective, we attribute this phenomenon primarily to the presence of confounding semantic features. To address it, we propose Causal Reward Adjustment (CRA), a method that mitigates reward hacking by estimating the true reward of a reasoning path. CRA trains sparse autoencoders on the PRM's internal activations to recover interpretable features, then corrects confounding by using backdoor adjustment. Experiments on math solving datasets demonstrate that CRA mitigates reward hacking and improves final accuracy, without modifying the policy model or retraining PRM.

Code — https://github.com/Ruike-Song/CRA.git

Introduction

Large language models (LLMs) have been widely recognized as a central focus of contemporary artificial intelligence research (Shao et al. 2024a; Zhao et al. 2023). One prominent direction in LLM research is external reasoning, which relies on external reward signals such as Process Reward Models (PRMs) to guide the reasoning process (Lightman et al. 2023; Snell et al. 2024a). This approach has demonstrated strong performance on complex tasks such as mathematical problem solving (Wei et al. 2022; Lightman et al. 2023; Uesato et al. 2022; Liu et al. 2025). However, external reasoning systems are vulnerable to a critical flaw known as **reward hacking**, where reward models assign higher scores to incorrect reasoning steps than to correct ones (Skalse et al. 2022). As a result, the system might select high-score steps that are logically wrong, which ultimately reduces the accuracy of the final output.

To better understand the mechanism underlying the reward hacking phenomenon, we adopt a causal inference perspective and formulate the problem using a Structural Causal Model (SCM). In our formulation, the input reasoning path is denoted as X, the score as Y, and semantic features unrelated to correctness as Z. These features may include stylistic elements, step length, or frequently used expressions. Ideally, logically correct reasoning paths (X) should causally lead to higher reward scores (Y), represented by the direct path $X \rightarrow Y$. However, in observed training data, due to annotation biases or inherent preferences of annotators, reasoning paths that contain certain semantic patterns (Z) frequently receive higher labeled rewards. As a result, the presence of these semantic features (Z) creates a spurious correlation between reasoning paths (X) and reward scores (Y), represented by a backdoor path $X \leftarrow Z \rightarrow Y$. Here, the semantic feature Z acts as a confounder, leading to a confounding effect where the observed distribution $\mathbb{E}[Y \mid X]$ no longer accurately represents the true causal relationship $X \rightarrow Y$. Consequently, the PRM trained on such biased observational data learns to assign rewards based not only on logical correctness ($X \rightarrow Y$) but also on the presence of semantic features ($Z \rightarrow Y$), causing the reward hacking phenomenon. This confounding effect can be addressed using a backdoor adjustment. Specifically, it involves evaluating how a given reasoning path X would be scored under different values of Z. These scores are then averaged, weighted by how frequently each semantic feature occurs in the data. This neutralizes the influence of the confounder and recovers the true causal effect.

Inspired by the above causal analysis, we propose a causally grounded method called **Causal Reward Adjustment (CRA)**. CRA consists of three steps: (1) extracting interpretable features from the reward model, (2) identifying confounding features associated with reward hacking, and (3) implementing backdoor adjustment using the identified features to mitigate the reward hacking problem. In the first step, we train a sparse autoencoder (SAE) on the hidden representations of PRM. The SAE learns to encode each internal representation into a sparse latent vector, where each dimension corresponds to a distinct semantic feature. This design ensures that the latent space is interpretable and suitable for identifying the reward hacking feature. In the second step, we detect which latent feature is the reward hacking feature. We compute the activation distribution of each feature over two groups of reasoning steps, those labeled as reward hacking and those considered normal, and

*These authors contributed equally.
†Corresponding author

apply two-sample t-tests to quantify their statistical separation. Features with both statistically significant differences and sufficiently high activation are selected as potential reward hacking features. In the third step, we perform backdoor adjustment by marginalizing over the identified features. Specifically, we simulate the reward scores that would be assigned to a given reasoning path under different values of the confounding features, and compute a weighted average using their empirical frequencies in the training data. This procedure removes the influence of spurious correlations and yields an unbiased estimate of the true causal effect of the reasoning path on the reward, thereby directly addressing the reward hacking problem.

To evaluate the effectiveness of CRA, we conduct experiments on two mathematical reasoning benchmarks: GSM8K and MATH. Results show that CRA significantly mitigates reward hacking and improves the reasoning correctness. Furthermore, ablation studies show that intervening on features identified by our CRA method effectively suppresses reward hacking, while random interventions have little impact. These findings validate both our causal analysis and the effectiveness of the proposed intervention strategy. Our main contributions are summarized as follows:

- We provide a causal explanation for reward hacking, showing that semantic confounders induce spurious correlations between reasoning steps and rewards. As a result, the reward model may assign high scores to incorrect reasoning steps simply because they contain preferred stylistic patterns.
- We propose CRA, a causally grounded three-step method that (i) extracts interpretable features via sparse autoencoders, (ii) identifies features responsible for reward hacking, and (iii) performs backdoor adjustment to eliminate their spurious influence, thereby mitigating the reward hacking effect.
- Experiments on GSM8K and MATH show that CRA reduces reward hacking and improves reasoning performance, demonstrating the effectiveness of our approach.

Related Works

Inference Methods in Large Language Models. Recent advances in LLMs have led to the development of external reasoning methods, which enhance complex problem solving by decoupling inference-time reasoning from model parameters. These approaches construct multiple candidate reasoning paths using frozen LLMs, and rely on sampling- or search-based mechanisms guided by an external verifier to select high-quality solutions (Lightman et al. 2023; Wu et al. 2024; Snell et al. 2024b; Yao et al. 2023; Sel et al. 2024; Besta et al. 2024; Zhang et al. 2022; Brown et al. 2024; Liu et al. 2025). Compared to internal approaches that improve reasoning via reinforcement learning (Madaan et al. 2023; Saunders et al. 2022; DeepSeek-AI 2025; Shao et al. 2024b), external methods offer flexibility and modularity, enabling dynamic inference-time optimization without retraining. This paradigm has shown strong performance on math-reasoning benchmarks such as MATH (Hendrycks et al. 2021), and GSM8K (Cobbe et al. 2021).

Reward Hacking. PRMs can guide LLM reasoning but can also be exploited: models may produce reasoning that receives high PRM scores yet is clearly incorrect. This "high score but incorrect" behavior is called reward hacking, where models find unintended shortcuts in the reward function (Weng 2024; Skalse et al. 2022; Pan et al. 2024; Liu et al. 2024). This is captured by Goodhart's Law(Goodhart 2015): "when a measure becomes a target, it ceases to be a good measure." Amodei et al.(Amodei et al. 2016) identified reward hacking as a core safety concern, emphasizing the risks of models exploiting flawed objectives. In single-agent settings, mitigation strategies include information-theoretic regularization (Miao et al. 2024) and intent-inference frameworks such as Inverse Reward Design (IRD) and Cooperative Inverse Reinforcement Learning (CIRL) (Hadfield-Menell et al. 2020), which seek to recover human preferences from observed behavior. In multi-agent reinforcement learning, sparse or misspecified rewards are handled through targeted communication and exploration (Sun et al. 2024a, 2021, 2024b, 2025; Zang et al. 2025).

Problem Formulation

In this section, we formalize the reward hacking problem in external reasoning tasks. We first describe the external reasoning framework, where policy models generate reasoning paths scored by reward models, then define reward hacking as assigning a high score to an incorrect reasoning path.

Solving Reasoning Problem with External Reward

Reasoning tasks typically involve problems and their ground-truth answers drawn from a distribution $(x, y^*) \sim D$. The primary goal of external reasoning systems is to maximize the expected accuracy of solving these problems. To achieve this, external reasoning relies on PRMs to identify and select high-quality reasoning paths. An external reasoning system comprises three main components: (i) A policy model $\pi_\theta : \mathcal{X} \times R^{(t-1)} \to \Delta(S)$. Given an input problem $x \in \mathcal{X}$ and the sequence of previously generated reasoning steps $r^{(t-1)} = (s_1, s_2, \ldots, s_{t-1})$, the policy model generates a distribution over possible next reasoning steps $s_t \in S$. Here, π_θ is a language model parameterized by θ, $\mathcal{X}$ represents the space of input problems, S is the set of potential reasoning steps, and $\Delta(S)$ denotes the probability distribution over S. (ii) A PRM $R_\phi : \mathcal{X} \times R^{(t)} \to \mathbb{R}$, which evaluates the quality of a reasoning trajectory up to the current step $r^{(t)} = (s_1, s_2, \ldots, s_t)$ and provides a numerical score. The PRM, parameterized by ϕ, assigns higher scores to trajectories that better reflect accurate or desirable reasoning. (iii) A search algorithm, such as beam search, which utilizes these reward scores to iteratively guide the construction of reasoning paths.

The external reasoning process using beam search begins with an empty reasoning trajectory for a given problem x. At each time step t, the policy model π_θ proposes candidate next steps for each partial trajectory currently retained. These expanded trajectories are then scored by the reward model R_ϕ. Next, the top-K highest-scoring trajectories are preserved for further expansion, while lower-scoring alter-

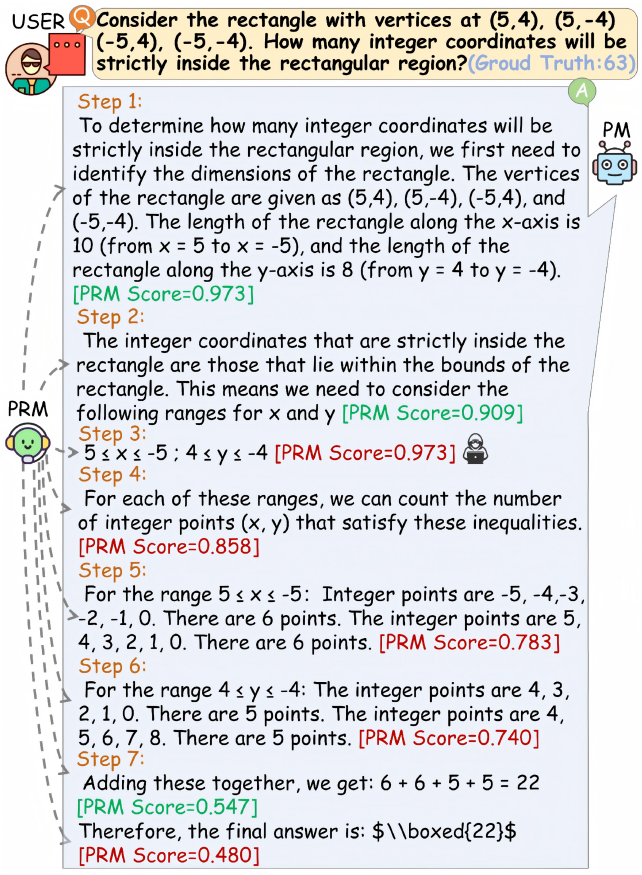


Figure 1: Example of reward hacking in mathematical reasoning. PRM scores are shown in brackets: green indicates correct steps, red indicates erroneous steps. The little hacker icon identifies reward hacking instances where logically flawed steps receive high scores.

natives are discarded. This iterative cycle of expansion and selective pruning continues until complete reasoning paths emerge. Finally, the output $(\hat{r}, \hat{y})$ is selected by:

$$(\hat{r}, \hat{y}) = \underset{(r,y)\in \text{Complete}}{\arg\max} \; R_\phi(x, r) \tag{1}$$

where Complete is the set of all fully formed reasoning paths terminating with a final answer.

The ultimate goal of external reasoning is defined as maximizing the expected accuracy over the distribution D:

$$\mathbb{E}_{(x,y^*)\sim D}[\mathbb{I}[\hat{y} = y^*]] \tag{2}$$

where $\mathbb{I}[\cdot]$ is the indicator function representing correct predictions, and y^* denotes the ground-truth answer corresponding to input problem x. In summary, achieving this goal critically depends on the PRM's ability to accurately score reasoning trajectories that lead to correct answers.

The Reward Hacking Problem

External reasoning systems rely on PRMs to rank candidate reasoning paths. However, these models occasionally assign high scores to reasoning steps that are not mathematically correct. As a consequence, the system tends to favor incorrect paths with higher scores, rather than correct ones with lower scores. This phenomenon is known as reward hacking.

To illustrate the reward hacking phenomenon, we present a concrete example from mathematical reasoning in Figure 1, which demonstrates an instance of reward hacking. Specifically, Step 3 introduces constraints "$5 \leq x \leq -5$; $4 \leq y \leq -4$", which are mathematically impossible since no number can simultaneously satisfy being greater than 5 and less than -5. Despite this clear logical contradiction, the reward model assigns a high score of 0.973 to this step. This led to an incorrect final answer.

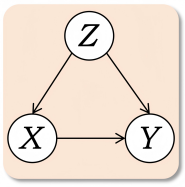


Figure 2: The SCM showing the relationship between spurious patterns (Z), reasoning paths (X), and reward scores (Y).

Causal Analysis

During external reasoning, we observed that PRM occasionally assign high scores even to reasoning steps that are logically incorrect. To better understand the underlying causes of this phenomenon, we adopt a causal inference perspective. Specifically, we identify that the issue stems from the confounding effects induced by specific semantic patterns during the reward evaluation process. Based on the analysis, we argue that the true reward of a reasoning step can be estimated through backdoor adjustment, motivating novel approaches to mitigating reward hacking.

Structural Causal Model

We formalize the problem using a Structural Causal Model (SCM) to describe causal relationships between observational variables (Pearl, Glymour, and Jewell 2016), as illustrated in Figure 2. Let variable X represent the sequence of reasoning steps, Y the correctness score that reflects the logical validity of the reasoning, and Z the semantic features unrelated to correctness, such as stylistic elements, step length, or frequently used expressions.

In the SCM, the directed edge $X \rightarrow Y$ signifies that coherent and logically correct reasoning paths X should causally increase the reward score Y. The edge $Z \rightarrow X$ means that semantic patterns Z, such as step length, stylistic phrases, or frequently used templates, often appear in the generated reasoning steps. These features influence how the model constructs X. The edge $Z \rightarrow Y$ means that human annotators or automated annotation processes tend to assign higher scores to reasoning steps that contain such semantic patterns, regardless of their logical correctness (Zhang et al. 2025). Together, $X \leftarrow Z \rightarrow Y$ forms a backdoor path, introducing a spurious correlation between X and Y.

The Confounding Effect

The structure of the SCM described above implies that, in observational data, the correctness score Y is influenced not only by the logical validity of the reasoning path X ($X \rightarrow Y$), but also by the presence of certain semantic features Z ($X \leftarrow Z \rightarrow Y$). Specifically, Z acts as a confounder, simultaneously affecting the generation of reasoning steps and the labels assigned during annotation.

In practice, the PRM is trained to predict the probability that $Y = 1$ by approximating the conditional expectation $\mathbb{E}[Y \mid X]$. In the presence of confounding, the conditional expectation learned by the PRM can be decomposed as:

$$\mathbb{E}[Y \mid X] = \sum_{z \in \{0,1\}} \mathbb{E}[Y \mid X, Z = z]\, P(Z = z \mid X). \quad (3)$$

When annotation biases yield $P(Y = 1 \mid Z = 1) \gg P(Y = 1 \mid Z = 0)$, the term $\mathbb{E}[Y \mid X = x, Z = 1]$ is close to one while $\mathbb{E}[Y \mid X = x, Z = 0]$ is near zero. Therefore, the conditional expectation simplifies to:

$$\mathbb{E}[Y \mid X = x] \approx P(Z = 1 \mid X = x). \quad (4)$$

Equation 4 shows that the PRM learns to predict whether the confounding feature Z is present in the reasoning path, rather than assessing the true logical validity of X, directly leading to the reward hacking phenomenon.

Motivation

Given that the PRM's prediction $\mathbb{E}[Y \mid X = x]$ is biased by the confounding influence of Z, we seek to recover the true causal effect of the reasoning path on reward by removing this bias. Specifically, the backdoor adjustment formula states that the true causal effect of X on Y, represented as $\mathbb{E}[Y \mid \mathrm{do}(X = x)]$, can be estimated by marginalizing over the confounder Z according to its marginal distribution:

$$\mathbb{E}[Y \mid \mathrm{do}(X)] = \sum_{z} \mathbb{E}[Y \mid X, Z = z]\, P(Z = z). \quad (5)$$

This means that, for any given reasoning path $X = x$, the estimated reward is computed as if the confounding feature Z could take either value, regardless of whether x actually contains Z or not. In practice, it is common to assume a balanced distribution, i.e., $P(Z = 1) = P(Z = 0) = 0.5$, so that neither presence nor absence of the confounder dominates the estimated score. As a consequence, the adjusted reward is independent of spurious semantic patterns, thereby effectively eliminating reward hacking. This motivates using a backdoor adjustment to mitigate reward hacking.

Methodology

Grounded in our causal analysis, we propose Causal Reward Adjustment (CRA), a method designed to address reward hacking through backdoor adjustment. CRA consists of three steps: (1) Training SAEs on reward models to decompose internal activations into interpretable semantic features, (2) Identifying reward hacking semantics by statistically analyzing which features discriminate between reward hacking and normal reasoning steps, and (3) Implement backdoor adjustment using the identified features to mitigate the reward hacking problem and improve system accuracy. The overall framework is illustrated in Figure 3.

Training Sparse Autoencoders on Reward Models

To identify interpretable semantic features within the reward model that may contribute to reward hacking, we train a set of SAEs to decompose the internal activations of the model into sparse, disentangled components.

Token-Level Activation Collection. To collect the training data for training SAEs, we first construct a corpus of reasoning paths by prompting multiple instruction-tuned LLMs (Qwen2.5-0.5B-Instruct, Qwen-2.5-math-7B-Instruct, LLaMA-3.1-8B-Instruct) to perform step-by-step reasoning on problems from training sets of GSM8K and MATH. Each resulting path consists of a sequence of intermediate reasoning steps, which are then processed token-by-token by the reward model R_ϕ to extract internal activations. And for each reasoning path, we tokenize the input and extract the hidden representation $h_{l,k} \in \mathbb{R}^d$ for each token position k from each Transformer block $l \in \{1, ..., L\}$. These token-level activations will serve as input data for layer-specific autoencoders.

Layer-Wise SAE Architecture. We train one SAE for each Transformer block layer in the reward model. At each layer l, the SAE encoder maps the hidden activation at token position k to a sparse feature vector:

$$z_{l,k} = \mathrm{ReLU}(W_e^{(l)} h_{l,k} + b_e^{(l)}), \quad (6)$$

and the decoder reconstructs the original activation:

$$\hat{h}_{l,k} = W_d^{(l)} z_{l,k}. \quad (7)$$

Here, $h_{l,k} \in \mathbb{R}^d$ denotes the hidden representation of token k at layer l in the reward model, and $z_{l,k} \in \mathbb{R}^m$ is the corresponding sparse latent vector. The encoder consists of a weight matrix $W_e^{(l)} \in \mathbb{R}^{m \times d}$ and a bias vector $b_e^{(l)} \in \mathbb{R}^m$, while the decoder $W_d^{(l)} \in \mathbb{R}^{d \times m}$ uses the transpose of $W_e^{(l)}$ to reconstruct the input, i.e., $W_d^{(l)} = W_e^{(l)\top}$ Here, d is the hidden size of the reward model, and m is the number of latent features in the sparse representation. We set $m = 8d$ to promote overcomplete and disentanglement representations.

We train the SAE at each layer l by minimizing the following empirical loss over the training set $\mathcal{H}_l$ of token-level activations:

$$\mathcal{L}^{(l)} = \frac{1}{|\mathcal{H}_l|} \sum_{h_{l,k} \in \mathcal{H}_l} \left[\frac{1}{d} \|h_{l,k} - \hat{h}_{l,k}\|_2^2 + \alpha \|z_{l,k}\|_1 \right]. \quad (8)$$

Here, $\mathcal{H}_l$ denotes the set of hidden activations collected from layer l across all tokens and reasoning paths in the training corpus, $\|\cdot\|_2$ denotes the Euclidean norm, and $\|\cdot\|_1$ denotes the ℓ_1 norm. The first term ensures accurate reconstruction of the reward model's activations, and the second term enforces sparsity in the latent features. Besides those, the sparsity coefficient α is a hyperparameter.

Feature Interpretation. After training, the sparse autoencoder provides interpretable semantic features. Specifically, each row vector $f_i^{(l)} \in \mathbb{R}^d$ of the decoder weight matrix $W_d^{(l)}$ represents a distinct semantic feature in layer l, effectively serving as a basis vector capturing specific patterns in the activation space. To understand how these features work in practice, consider an input token sequence

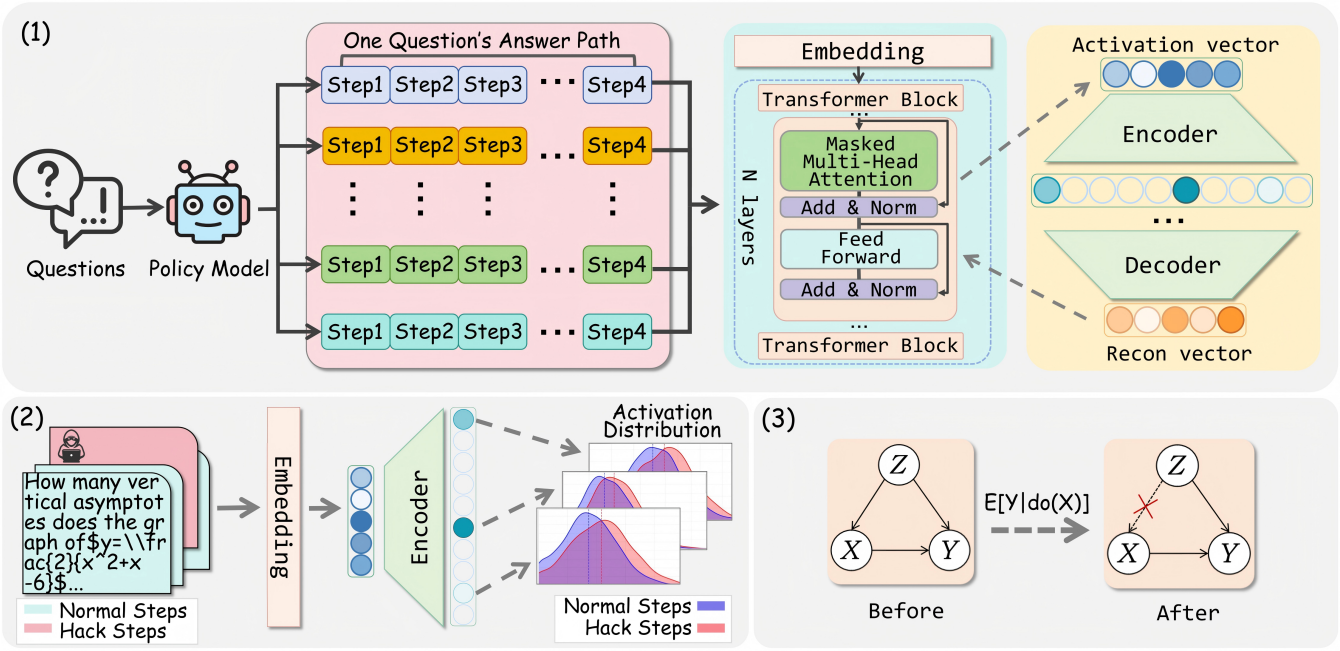


Figure 3: **(1) Training Sparse Autoencoders on Reward Models.** Given some input problems, the policy model generates reasoning steps. These steps are then passed through selected transformer blocks in the reward model to extract step-level activation vectors. The activations are used to train SAEs. **(2) Identifying Reward Hacking Semantics.** The reasoning steps are manually labeled as normal (blue) or reward hacking (red with a hacker icon). These labeled samples are then encoded using the SAE trained in (a), and the activation distributions of individual features are obtained. Those with significant distribution differences are regarded as reward hacking features. **(3) Implementing Backdoor Adjustment.** Based on the reward hacking features identified in the previous step, the influence of these features during the scoring process can be cut off through backdoor regulation, thereby addressing the reward hacking issue.

$T = [k_1, k_2, \ldots, k_N]$. The hidden activation $h_{l,k_N} \in \mathbb{R}^d$ at the final token k_N in layer l can be decomposed as a linear combination of these learned semantic basis vectors:

$$h_{l,k_N} \approx \sum_{i=1}^{m} z_{l,T}^{(i)} f_i^{(l)} = W_d^{(l)} z_{l,T}, \tag{9}$$

where $z_{l,T} \in \mathbb{R}^m$ is the sparse code computed by the encoder over the input sequence T. Each coefficient $z_{l,T}^{(i)}$ reflects the presence and strength of the i-th semantic feature in the representation of T: if $z_{l,T}^{(i)} = 0$, the feature $f_i^{(l)}$ is inactive; if large, it strongly influences the output. Due to the enforced sparsity, only a few features are active per input, making their contribution interpretable. Empirically, these features align with symbolic computations, common logical substeps, or spurious patterns exploited by reward hacking (Bricken et al. 2023). In the next section, we analyze these features to identify those correlated with reward hacking.

Identifying Reward Hacking Semantics

After obtaining SAEs, we can decompose the reward model activations into interpretable features with them. Next, we identify which specific features are responsible for reward hacking behavior. This process involves two key steps: (1) constructing a labeled dataset of reward hacking versus normal reasoning steps, (2) applying statistical analysis to identify discriminative features. Through this analysis, we can pinpoint the features that cause reward models to favor reward hacking steps.

Dataset Construction for Feature Analysis. To identify reward hacking features, we construct labeled examples by analyzing reasoning trajectories from the beam search process. We isolate every intermediate reasoning step and assign binary labels based on two criteria: mathematical validity and reward score. Specifically, we label $y_i = 1$ for steps that are mathematically incorrect yet receive high reward scores (reward hacking instances), and $y_i = 0$ for all other steps (normal instances). This labeling process yields N_1 reward hacking steps and N_0 normal steps, providing sufficient data for statistical analysis.

Statistical Feature Selection. Building on our causal analysis, we aim to leverage SAEs to identify semantic features responsible for reward hacking. To achieve this, we propose a statistical screening approach: Each labeled reasoning step is first encoded into a sparse representation $z_i \in \mathbb{R}^m$ using the trained SAEs, where m denotes the number of learned semantic features. We then identify reward hacking features by selecting those whose activation distribution differs significantly between reward hacking steps and normal reasoning steps.

For each sparse dimension j, we compute the mean activations $\mu_{1,j}$ and $\mu_{0,j}$, and variances $\sigma_{1,j}^2$ and $\sigma_{0,j}^2$ within the

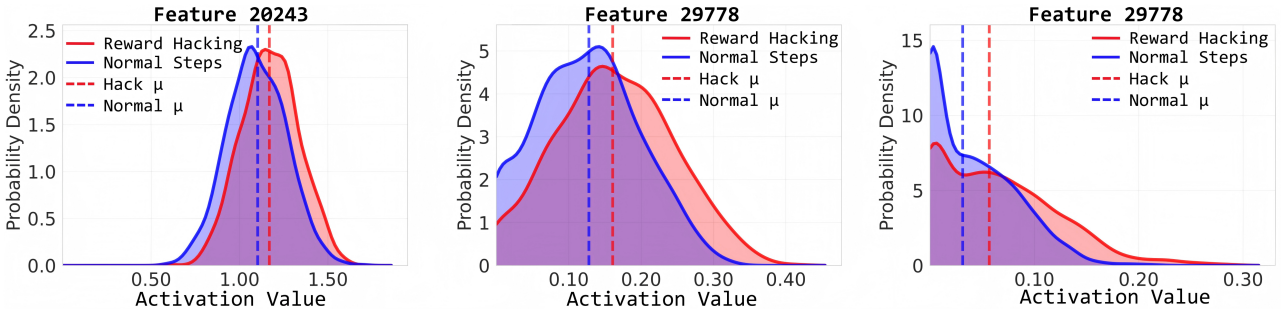


Figure 4: Examples of features with high t-statistics showing systematic differences between reward hacking (red) and normal (blue) reasoning steps. Dashed vertical lines indicate mean activation values for each condition.

two classes. We then compute the two-sample t-statistic:

$$t_j = \frac{\mu_{1,j} - \mu_{0,j}}{\sqrt{\sigma_{1,j}^2/n_1 + \sigma_{0,j}^2/n_0}} \tag{10}$$

where n_1 and n_0 are the numbers of reward hacking and normal samples, respectively.

Features with high $|t_j|$ values represent the confounding variable Z in our SCM, since they exhibit systematic differences between the two conditions, indicating their role in creating spurious correlations between reasoning paths and reward scores. We retain dimensions satisfying both statistical significance and activation thresholds:

$$|t_j| > \tau_t \text{ and } \max(\mu_{1,j}, \mu_{0,j}) > \tau_a \tag{11}$$

where τ_t controls statistical significance and τ_a filters out low-activation noisy dimensions. We show in Figure 4 the probability distributions of activation values for representative features. The vertical dashed lines indicate the mean activation values for each condition, showing systematic shifts between the two distributions. Having identified the observable confounding features $\mathcal{F}^* = j : |t_j| > \tau_t \wedge \max(\mu_{1,j}, \mu_{0,j}) > \tau_a$, we can implement causal intervention based on backdoor adjustment.

Implementing Backdoor Adjustment

Once we identify which SAE latent corresponds to a reward-hacking semantic feature, we can perform backdoor adjustment based on this latent. This process consists of four steps: (1) We collect the activation values of the reward-hacking feature across the dataset to estimate its prior distribution. (2) For each reasoning step, we substitute the activation value of the reward-hacking feature with different possible values, reconstruct the hidden state, and obtain the corresponding conditional rewards from the PRM. (3) The final CRA score, $\hat{R}_{\text{CRA}}$, is computed as the weighted average of all conditional rewards, using the estimated prior as weights. (4) We then use $\hat{R}_{\text{CRA}}$ to replace the original PRM scores when selecting reasoning paths in external reasoning.

Building Prior Distribution. According to Equation 5, implementing backdoor adjustment requires estimating two components: the prior distribution $P(Z)$ and the conditional rewards $\mathbb{E}[Y \mid X, Z]$. To estimate the prior, we first collect the activation values of the reward-hacking feature across all reasoning steps. These activation values are then grouped into non-overlapping bins to construct an empirical distribution. For each bin i with range $[z_i, z_{i+1}]$, the prior probability is calculated as

$$P(Z \in [z_i, z_{i+1}]) = \frac{n_i}{N}, \tag{12}$$

where n_i is the number of steps within the bin and N is the total number of steps. The midpoint of each bin is used as the intervention value z in subsequent calculations.

Compute Conditional Rewards. To obtain $\mathbb{E}[Y|X, Z = z]$, for each reasoning step t, we edit the SAE latent vector z_t by replacing the activation value at the target dimension with z, while keeping all other dimensions unchanged:

$$\tilde{z}_t^{(j)} = \begin{cases} z, & \text{if } j \in \mathcal{F}^\star \\ z_t^{(j)}, & \text{otherwise} \end{cases} \tag{13}$$

We then decode $\tilde{z}_t$ to obtain the hidden state $\tilde{h}_t = W_d \tilde{z}_t$, and input it into the PRM. The resulting PRM output is taken as the conditional reward $\mathbb{E}[Y|X, Z = z]$.

Perform Backdoor Adjustment. Finally, the adjusted reward is computed as the weighted average over all conditional rewards:

$$\hat{R}_{\text{CRA}}(x) = \sum_z \mathbb{E}[Y \mid X, Z = z] \cdot P(Z = z). \tag{14}$$

By this construction, $\hat{R}_{\text{CRA}}(x)$ removes the PRM's bias toward the semantic feature Z, thereby mitigating the reward hacking problem.

System Integration and Evaluation. We integrate the adjusted reward into the external reasoning pipeline by replacing the original PRM score with the adjusted reward $\hat{R}_{\text{CRA}}$ at each reasoning step. During beam search, this adjusted score guides the selection of partial trajectories, which downgrades the rewards of incorrect steps even if they contain the semantic feature Z.

Experiments And Results

In this section, we evaluate the effectiveness of CRA by comparing external reasoning accuracy before and after CRA integration on mathematical benchmarks, assessing whether it improves the reasoning accuracy.

Policy Model	Qwen2.5-0.5B-Instruct		Qwen-2.5-math-7B-Instruct		Llama-3.2-3B-Instruct	
Reward Model \ Dataset	MATH	GSM8K	MATH	GSM8K	MATH	GSM8K
Math-Shepherd-PRM-7B	40.1	55.1	77.0	96.8	48.3	78.1
Qwen2.5-Math-PRM-7B	46.6	60.9	78.1	96.5	53.9	80.1
⋆ Math-Shepherd-PRM-7B + **CRA**	43.7	58.0	80.3	**97.1**	51.7	80.7
⋆ Qwen2.5-Math-PRM-7B + **CRA**	**48.6**	**62.3**	**80.6**	97.0	**56.4**	**82.1**

Table 1: Performance comparison using beam search (beam = 4) across different policy models. ⋆ represents our trained models.

Experimental Setup

Datasets. We evaluate our method on GSM8K(Cobbe et al. 2021) and MATH(Hendrycks et al. 2021). For SAE training, we construct a corpus of 18,000 reasoning trajectories, comprising over 190,000 reasoning steps.

Models and Architecture. We use Qwen2.5-0.5B-Instruct as the primary policy model, with additional evaluations on Qwen-2.5-math-7B-Instruct and Mistral-7B-Instruct. For reward models, we employ Math-Shepherd-Mistral-7B and Qwen2.5-Math-PRM-7B. We train layer-wise SAEs with sparse dimension $m = 8d$ to encourage interpretable feature discovery.

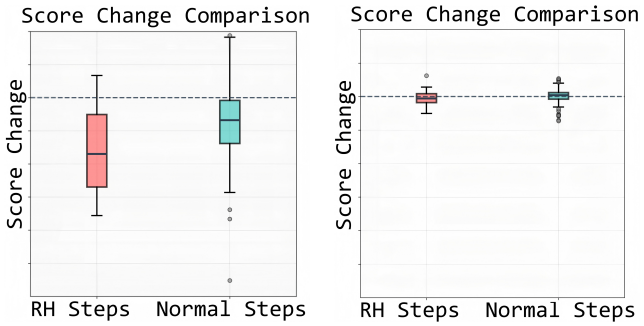
Implementation Details. We implement our approach using PyTorch 2.0 on 8 Tesla V100 GPU with mixed-precision training. Beam search is configured with a beam size of 4. At each expansion step, 8 candidate steps are generated, and 8 complete reasoning paths are produced as final outputs. SAEs are trained using the Adam optimizer (lr=0.001, cosine annealing) for 50 epochs with a batch size of 2,048 and a sparsity coefficient $\alpha = 0.001$. For feature selection, we use thresholds $\tau_t = 4.0$ and $\tau_a = 0$ to identify discriminative features for intervention.

Results

The results are shown in Table 1. From Table 1, we can observe that CRA consistently improves performance in all settings. On MATH, the average accuracy increases from 57.3% to 60.2%, with an average gain of 2.9 percentage points. On GSM8K, the accuracy improves from 77.9% to 79.5%, with an average gain of 1.6 percentage points. Specifically, in the combination of Qwen2.5-Math-PRM-7B and LLaMA-3.2-3B-Instruct as the policy model, GSM8K accuracy improves from 80.1% to 82.1%, indicating that CRA provides consistent benefits even in strong model configurations. These results confirm that CRA mitigates reward hacking and improves external reasoning reliability.

Ablation Experiment

We conduct ablation studies to validate CRA by comparing it against a random intervention baseline. Both methods share the same technical pipeline, differing only in feature selection: CRA intervenes on features identified by their statistical significance (highest t-statistics), while the random baseline selects features arbitrarily.



(a) Causal intervention (b) Random intervention

Figure 5: Score change distributions after feature intervention. "RH Steps" denotes reward hacking steps, and "Normal Steps" denotes non-hacking reasoning. Both panels share the same vertical scale. (a) Causal intervention markedly reduces scores for reward hacking instances while minimally affecting normal steps. (b) Random intervention produces negligible, non-discriminative effects on both step types.

Experimental results (Figure 5) show a clear advantage for CRA. When intervening on selected features, CRA specifically lowers the scores of reward-hacking steps (average decrease ≈ -0.04) without significantly impacting normal reasoning steps. Conversely, random feature interventions produce negligible effects with score changes tightly clustered around zero. These findings confirm that precise, causally informed feature identification is critical for effectively mitigating reward hacking.

Conclusion

We study reward hacking in external reasoning systems through a causal lens and attribute it to confounding semantic features that jointly affect reasoning generation and reward evaluation. Building on this analysis, we propose Causal Reward Adjustment (CRA), which trains sparse autoencoders on process reward models to identify such confounders and applies backdoor adjustment to compute debiased rewards. Experiments on GSM8K and MATH show that CRA reduces the selection of flawed reasoning steps and improves final accuracy, without modifying policy models or retraining reward models.

Acknowledgments

The authors would like to thank the anonymous reviewers for their valuable comments. This work was supported by the National Natural Science Foundation of China under Grants 62506355 and 42506186, and the numerical calculations in this study were partially performed on the ORISE Supercomputer (DFZX202416).

References

Amodei, D.; Olah, C.; Steinhardt, J.; Christiano, P.; Schulman, J.; and Mané, D. 2016. Concrete Problems in AI Safety. arXiv:1606.06565.

Besta, M.; Blach, N.; Kubicek, A.; Gerstenberger, R.; Podstawski, M.; Gianinazzi, L.; Gajda, J.; Lehmann, T.; Niewiadomski, H.; and Nyczyk, P. 2024. Graph of Thoughts: Solving Elaborate Problems with Large Language Models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, 17682–17690.

Bricken, T.; Templeton, A.; Batson, J.; Chen, B.; Jermyn, A.; Conerly, T.; Turner, N.; Anil, C.; Denison, C.; Askell, A.; Lasenby, R.; Wu, Y.; Kravec, S.; Schiefer, N.; Maxwell, T.; Joseph, N.; Hatfield-Dodds, Z.; Tamkin, A.; Nguyen, K.; McLean, B.; Burke, J. E.; Hume, T.; Carter, S.; Henighan, T.; and Olah, C. 2023. Towards Monosemanticity: Decomposing Language Models With Dictionary Learning. *Transformer Circuits Thread*. Https://transformer-circuits.pub/2023/monosemantic-features/index.html.

Brown, B.; Juravsky, J.; Ehrlich, R.; Clark, R.; Le, Q. V.; Ré, C.; and Mirhoseini, A. 2024. Large Language Monkeys: Scaling Inference Compute with Repeated Sampling. arXiv:2407.21787.

Cobbe, K.; Kosaraju, V.; Bavarian, M.; Chen, M.; Jun, H.; Kaiser, L.; Plappert, M.; Tworek, J.; Hilton, J.; Nakano, R.; Hesse, C.; and Schulman, J. 2021. Training Verifiers to Solve Math Word Problems. *arXiv preprint arXiv:2110.14168*.

DeepSeek-AI. 2025. DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning. arXiv:2501.12948.

Goodhart, C. 2015. Goodhart's law.

Hadfield-Menell, D.; Milli, S.; Abbeel, P.; Russell, S.; and Dragan, A. 2020. Inverse Reward Design. arXiv:1711.02827.

Hendrycks, D.; Burns, C.; Kadavath, S.; Arora, A.; Basart, S.; Tang, E.; Song, D.; and Steinhardt, J. 2021. Measuring Mathematical Problem Solving With the MATH Dataset. *NeurIPS*.

Lightman, H.; Kosaraju, V.; Burda, Y.; Edwards, H.; Baker, B.; Lee, T.; Leike, J.; Schulman, J.; Sutskever, I.; and Cobbe, K. 2023. Let's Verify Step by Step. arXiv:2305.20050.

Liu, R.; Gao, J.; Zhao, J.; Zhang, K.; Li, X.; Qi, B.; Ouyang, W.; and Zhou, B. 2025. Can 1b llm surpass 405b llm? rethinking compute-optimal test-time scaling. *arXiv preprint arXiv:2502.06703*.

Liu, T.; Xiong, W.; Ren, J.; Chen, L.; Wu, J.; Joshi, R.; Gao, Y.; Shen, J.; Qin, Z.; Yu, T.; et al. 2024. Rrm: Robust reward model training mitigates reward hacking. *arXiv preprint arXiv:2409.13156*.

Madaan, A.; Tandon, N.; Gupta, P.; Hallinan, S.; Gao, L.; Wiegreffe, S.; Alon, U.; Dziri, N.; Prabhumoye, S.; Yang, Y.; Gupta, S.; Majumder, B. P.; Hermann, K.; Welleck, S.; Yazdanbakhsh, A.; and Clark, P. 2023. Self-Refine: Iterative Refinement with Self-Feedback. In *Thirty-Seventh Conference on Neural Information Processing Systems*.

Miao, Y.; Zhang, S.; Ding, L.; Bao, R.; Zhang, L.; and Tao, D. 2024. Inform: Mitigating reward hacking in rlhf via information-theoretic reward modeling. *Advances in Neural Information Processing Systems*, 37: 134387–134429.

Pan, A.; Jones, E.; Jagadeesan, M.; and Steinhardt, J. 2024. Feedback loops with language models drive in-context reward hacking. *arXiv preprint arXiv:2402.06627*.

Pearl, J.; Glymour, M.; and Jewell, N. P. 2016. *Causal inference in statistics: A primer*. John Wiley & Sons.

Saunders, W.; Yeh, C.; Wu, J.; Bills, S.; Ouyang, L.; Ward, J.; and Leike, J. 2022. Self-critiquing models for assisting human evaluators. *arXiv preprint arXiv:2206.05802*.

Sel, B.; Tawaha, A.; Khattar, V.; Jia, R.; and Jin, M. 2024. Algorithm of Thoughts: Enhancing Exploration of Ideas in Large Language Models. In *Forty-First International Conference on Machine Learning*.

Shao, M.; Basit, A.; Karri, R.; and Shafique, M. 2024a. Survey of Different Large Language Model Architectures: Trends, Benchmarks, and Challenges. *IEEE Access*, 12: 188664–188706.

Shao, Z.; Wang, P.; Zhu, Q.; Xu, R.; Song, J.; Bi, X.; Zhang, H.; Zhang, M.; Li, Y. K.; Wu, Y.; and Guo, D. 2024b. DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models. arXiv:2402.03300.

Skalse, J.; Howe, N.; Krasheninnikov, D.; and Krueger, D. 2022. Defining and characterizing reward gaming. *Advances in Neural Information Processing Systems*, 35: 9460–9471.

Snell, C.; Lee, J.; Xu, K.; and Kumar, A. 2024a. Scaling LLM Test-Time Compute Optimally Can Be More Effective than Scaling Model Parameters. arXiv:2408.03314.

Snell, C. V.; Lee, J.; Xu, K.; and Kumar, A. 2024b. Scaling LLM Test-Time Compute Optimally Can Be More Effective than Scaling Parameters for Reasoning. In *The Thirteenth International Conference on Learning Representations*.

Sun, C.; He, P.; Ji, Q.; Zang, Z.; Li, J.; Wang, R.; and Wang, W. 2024a. M2i2: Learning efficient multi-agent communication via masked state modeling and intention inference. *arXiv preprint arXiv:2501.00312*.

Sun, C.; He, P.; Wang, R.; and Zheng, C. 2025. Revisiting Communication Efficiency in Multi-Agent Reinforcement Learning from the Dimensional Analysis Perspective. In Das, S.; Nowé, A.; and Vorobeychik, Y., eds., *Proceedings of the 24th International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2025, Detroit, MI, USA, May 19-23, 2025*, 1977–1986. International Foundation for Autonomous Agents and Multiagent Systems / ACM.

Sun, C.; Wu, B.; Wang, R.; Hu, X.; Yang, X.; and Cong, C. 2021. Intrinsic Motivated Multi-Agent Communication. In *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '21, 1668–1670. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems. ISBN 9781450383073.

Sun, C.; Zang, Z.; Li, J.; Li, J.; Xu, X.; Wang, R.; and Zheng, C. 2024b. T2mac: Targeted and trusted multi-agent communication through selective engagement and evidence-driven integration. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, 15154–15163.

Uesato, J.; Kushman, N.; Kumar, R.; Song, F.; Siegel, N.; Wang, L.; Creswell, A.; Irving, G.; and Higgins, I. 2022. Solving math word problems with process-and outcome-based feedback. *arXiv preprint arXiv:2211.14275*.

Wei, J.; Wang, X.; Schuurmans, D.; Bosma, M.; Xia, F.; Chi, E.; Le, Q. V.; and Zhou, D. 2022. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. *Advances in neural information processing systems*, 35: 24824–24837.

Weng, L. 2024. Reward Hacking in Reinforcement Learning. *lilianweng.github.io*.

Wu, Y.; Sun, Z.; Li, S.; Welleck, S.; and Yang, Y. 2024. Inference Scaling Laws: An Empirical Analysis of Compute-Optimal Inference for LLM Problem-Solving. In *The Thirteenth International Conference on Learning Representations*.

Yao, S.; Yu, D.; Zhao, J.; Shafran, I.; Griffiths, T. L.; Cao, Y.; and Narasimhan, K. R. 2023. Tree of Thoughts: Deliberate Problem Solving with Large Language Models. In *Thirty-Seventh Conference on Neural Information Processing Systems*.

Zang, Z.; Sun, C.; Liu, L.; Sun, F.; and Zheng, C. 2025. Loss of Plasticity: A New Perspective on Solving Multi-Agent Exploration for Sparse Reward Tasks. In Das, S.; Nowé, A.; and Vorobeychik, Y., eds., *Proceedings of the 24th International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2025, Detroit, MI, USA, May 19-23, 2025*, 2299–2308. International Foundation for Autonomous Agents and Multiagent Systems / ACM.

Zhang, Z.; Zhang, A.; Li, M.; and Smola, A. 2022. Automatic Chain of Thought Prompting in Large Language Models. arXiv:2210.03493.

Zhang, Z.; Zheng, C.; Wu, Y.; Zhang, B.; Lin, R.; Yu, B.; Liu, D.; Zhou, J.; and Lin, J. 2025. The Lessons of Developing Process Reward Models in Mathematical Reasoning. arXiv:2501.07301.

Zhao, W. X.; Zhou, K.; Li, J.; Tang, T.; Wang, X.; Hou, Y.; Min, Y.; Zhang, B.; Zhang, J.; Dong, Z.; et al. 2023. A survey of large language models. *arXiv preprint arXiv:2303.18223*, 1(2).